

VR-Forces

Users Guide

VT MAK
A company of VT Systems



VR-Forces

Users Guide

Version 4.7

VT MAK
A company of VT Systems

Copyright © 2019 VT MAK
All rights Reserved. Printed in the United States.

Under copyright laws, no part of this document may be copied or reproduced in any form without prior written consent of VT MAK.

VR-Exchange™, VR-TheWorld™, VR-Vantage™, DI-Guy™, and DI-Guy Scenario™ are trademarks of VT MAK. MÄK Technologies®, VR-Forces®, RTIspy®, B-HAVE®, and VR-Link® are registered trademarks of VT MAK.

Terrain Profiles are based in part on the work of the Qwt project (http://qwt.sourceforge.net).

All other trademarks are owned by their respective companies.

For third-party license information, please see "Third Party Licenses," on page lxiii.

VT MAK
150 Cambridge Park Drive, 3rd Floor
Cambridge, MA 02140 USA

Voice: 617-876-8085
Fax: 617-876-9208

info@mak.com

www.mak.com

Revision VRF-4.7-1-190422



Contents

Preface

How This Manual is Organized li
VR-Forces Documentation lvii
MAK Products lviii
How to Contact Us lxi
Document Conventions lxii
Mouse Button Naming Conventions lxiii
Third Party Licenses lxiii
Boost License lxv
libXML and libICONV lxv
Lua lxvi
Freefont OpenType Font Set lxvi
NVIDIA lxvi
Third-Party Licenses for VR-Vantage Applications lxvii

Section I Introduction, Installation, and Startup

Chapter 1. Introduction to VR-Forces

1.1. Overview 1-3
1.1.1. Entity-Level and Aggregate-Level Simulation 1-5
1.1.2. Realistic Display of Vehicles and Terrain (3D View) 1-6
1.1.3. Create Complex Scenarios 1-7
1.1.4. Simulation Object Types Supported 1-8
1.1.5. Mission Planning 1-10
1.1.6. Simulation Object Tasks 1-10
1.1.7. Scripted Tasks and Set Data Requests 1-11
1.1.8. Tactical Graphics 1-11
1.1.9. Terrain Agility and Composability 1-12
1.1.10. Dynamic Terrain 1-12
1.1.11. Crowd Behaviors and Pattern of Life 1-12
1.1.12. Flexible, Intuitive Graphical User Interface 1-13

1.1.13. Overlays 1-13
1.1.14. Behaviors 1-13
1.1.15. Observer Attach Modes 1-14
1.1.16. Special Effects and Simulation Object Information Visualization 1-14
1.1.17. Dynamic Ocean 1-14
1.1.18. Lighting Effects 1-14
1.1.19. Accurate Vehicle Positioning 1-15
1.1.20. Batch Mode Operation 1-15
1.1.21. Remote Control 1-15
1.2. The VR-Forces Toolkit 1-16
1.2.1. Plug-in Architecture 1-16
1.3. DI-Guy 1-17
1.4. Helpful Utilities 1-18
1.5. Third-Party Software and Content 1-19
1.5.1. SilverLining 1-19
1.5.2. Triton Ocean SDK 1-20
1.5.3. GL Studio 1-20
1.5.4. SpeedTree 1-21
1.5.5. 3D Models, Terrain, and Graphical Content 1-21
1.5.6. OpenSceneGraph 1-22
1.5.7. osgEarth 1-22
1.6. Distributed Simulation Standards Supported 1-22

Chapter 2. Installing VR-Forces

2.1. Installing VR-Forces 2-2
2.1.1. Installing VR-Forces on Windows 2-2
2.1.2. Installing VR-Forces on a Linux System 2-3
2.1.3. Uninstalling VR-Forces on Windows 2-3
2.2. The VR-Forces Directory Structure 2-4
2.3. Installing and Setting Up the MAK License Manager 2-6
2.3.1. Specifying the License Server 2-7
2.4. Installing an RTI 2-9
2.4.1. Installing the MAK RTI 2-10
2.5. Localizing the Graphical User Interface 2-11
2.5.1. Translating Other Interface Files 2-14
2.5.2. Translating VR-Forces Scripts and Console Messages ... 2-15
2.5.3. Applying the Language Files 2-16

Chapter 3. VR-Forces Application Concepts

3.1. The VR-Forces Program 3-2
3.2. Front-end and Back-end Concepts 3-2
3.2.1. How Front-ends and Back-ends Work Together 3-3
3.2.2. How VR-Forces Back-ends are Identified 3-4
3.2.3. VR-Forces Sessions 3-4
3.2.4. Coordinating Multiple Front-ends 3-6
3.2.5. Working with Multiple Back-ends 3-7

3.3. Objects 3-8
3.3.1. The Object Parameter Database 3-8
3.3.2. Local Objects and Remote Objects 3-9
3.4. Representing and Managing Time in VR-Forces 3-10
3.4.1. Simulation Time 3-10
3.4.2. Time of Day 3-10
3.4.3. Exercise Clock Modes 3-12
3.5. Interactions 3-13
3.6. Advanced Terrain Navigation 3-13
3.7. Managing VR-Forces Settings 3-13
3.7.1. Synchronizing Settings Among VR-Forces Installations 3-16
3.7.2. Global Settings and Observer-Specific Settings 3-16

Chapter 4. Starting VR-Forces

4.1. Starting VR-Forces 4-3
4.1.1. Starting VR-Forces from the VR-Forces Launcher 4-4
4.1.2. Starting Independent VR-Forces Executables 4-9
4.1.3. Specifying a Session ID 4-9
4.1.4. VR-Forces Startup Tutorial 4-9
4.2. The VR-Forces Window 4-11
4.2.1. Opening New Windows 4-13
4.2.2. Printing the VR-Forces Display 4-14
4.2.3. Taking Screen Captures 4-14
4.3. Managing a Front-end's Session Connection 4-15
4.3.1. Joining a Session 4-16
4.3.2. Resigning from a Session 4-17
4.3.3. Configuring Session Messages and Join at Startup 4-17
4.4. Using HLA Time Management 4-19
4.4.1. VR-Forces Simulation Time Versus Federation Time 4-19
4.4.2. Configuring Time Management for HLA Exercises 4-19
4.5. Opening a Terrain Database 4-21
4.5.1. Loading a Terrain Database at Startup 4-22
4.5.2. Displaying Terrain Correlation Warnings 4-23
4.5.3. Closing a Terrain 4-23
4.6. Exiting VR-Forces 4-23
4.6.1. Disabling the Quit Prompt 4-24
4.7. Configuring Simulation Connections 4-24
4.7.1. Opening the Simulation Connections Configuration Dialog Box 4-25
4.7.2. Adding a Simulation Connection 4-26
4.7.3. Editing a Simulation Connection 4-27
4.7.4. Simulation Connection Parameters 4-28
4.7.5. Deleting a Simulation Connection 4-30
4.7.6. Configuring Auto Connect 4-31
4.7.7. Displaying Connection Information 4-31
4.8. Managing Plug-ins 4-33
4.8.1. Loading Plug-ins 4-33
4.8.2. Adding a Plug-in 4-36
4.8.3. Specifying the DLLs for a Plug-in 4-37
4.8.4. Adding a Plug-in Configuration 4-38

4.8.5. Deleting a Plug-in Configuration 4-38
4.8.6. Deleting a Plug-in 4-39
4.8.7. Viewing a List of Loaded Plug-ins 4-39
4.9. The VR-Forces Log Files 4-40

Chapter 5. Command-Line Options

5.1. Command-Line Options for vrfGui 5-2
5.2. Command-Line Options for vrfSim 5-9
5.3. Command-Line Options for vrfLauncher 5-16
5.3.1. Running in Combined Mode from the Command Line ... 5-17
5.3.2. Using the -- Command-line Argument 5-17
5.4. Protocol-Independent Command-line Options 5-17
5.4.1. Specifying the Site ID and Application Number 5-18
5.4.2. Specifying the Language to Use in the GUI 5-18
5.4.3. Setting the Notification Level 5-18
5.4.4. Loading Plug-ins 5-19
5.5. Command-Line Options for HLA Federations 5-19
5.5.1. Specifying the Federation Execution 5-19
5.5.2. Specifying the FED File 5-19
5.5.3. Specifying the RPR FOM Version 5-20
5.5.4. Specifying a FOM Mapper 5-20
5.5.5. Specifying the RPR FOM Revision 5-20
5.5.6. Specifying FOM Mapper Initialization Data 5-20
5.5.7. Specifying a FED File That is Appropriate for the FOM Mapper 5-21
5.5.8. Enabling Time Management 5-21
5.6. Command-Line Options for DIS Exercises 5-21
5.6.1. Specifying the Port Number 5-21
5.6.2. Specifying Point-to-Point or Multicast Operation 5-22
5.6.3. Specifying the Multicast Time-to-Live 5-22
5.6.4. Using Asynchronous I/O 5-22
5.6.5. Specifying the Exercise ID 5-22
5.6.6. Specifying the DIS Version 5-22

Chapter 6. Optimizing Performance

6.1. Introduction 6-3
6.1.1. Optimizing Simulation Engine Performance 6-3
6.1.2. Optimizing Front-End Performance 6-4
6.2. Monitoring the Back-end Frame Rate 6-6
6.3. Monitoring Script Frame Rates 6-6
6.4. Displaying Performance Statistics 6-7
6.4.1. Displaying VR-Forces Performance Statistics 6-9
6.4.2. Displaying OSG Statistics 6-9
6.4.3. Displaying the VR-Forces Function Profiler 6-10
6.5. Filtering Simulation Objects Using Interest Management 6-10
6.5.1. Enabling Interest Management 6-10
6.5.2. Configuring Interest Management 6-11
6.6. Clearing the Model Instancing Cache 6-13

6.7. Miscellaneous Performance Configuration Options 6-14
6.7.1. Limiting Use of Spot Reports 6-14
6.7.2. Using Asynchronous I/O for DIS Exercises 6-14
6.7.3. Using Incremental Compiling with Streamed Data 6-14
6.8. Setting the Tick Rate 6-15
6.8.1. Specifying the Tick Rate for Components 6-15
6.8.2. Tuning the Network Interface 6-16
6.8.3. Tuning the State Repository Tick Rate 6-16
6.9. Configuring Graphics Quality 6-16
6.9.1. Alpha to Coverage 6-17
6.10. Balancing Visual Quality Against Network Performance 6-19
6.10.1. Enabling Configuration of Performance Settings 6-20
6.11. Trajectory Smoothing 6-21
6.11.1. Configuring Trajectory Smoothing 6-22
6.12. Tuning the Target Frame Rate 6-22
6.13. Coloring Draw Calls 6-23
6.14. Configuring VSync 6-24
6.15. Preloading Models 6-24
6.15.1. Command-Line Options for Preloading Models 6-25
6.16. DI-Guy Performance Settings 6-26
6.17. Using the Developer Debug Overlay 6-27

Section II Scenarios

Chapter 7. Creating and Running Scenarios

7.1. Creating a Scenario 7-3
7.1.1. Specifying Multiple Simulation Model Sets 7-8
7.1.2. Creating Simulation Model Set Configurations 7-9
7.1.3. Setting the Scenario Starting Date and Time 7-12
7.2. Importing Objects into Scenarios 7-14
7.2.1. Importing (Merging) Scenarios 7-14
7.2.2. Importing and Exporting MSDL 7-16
7.2.3. Importing Scenario Objects from CSV Files 7-18
7.2.4. Importing Airspace Control Orders 7-20
7.3. Saving a Scenario 7-21
7.3.1. Saving a Previously Saved Scenario 7-22
7.3.2. Saving a Scenario to a New Name or a Different Format 7-24
7.4. Loading a Scenario 7-24
7.4.1. Loading a Recently Loaded Scenario 7-26
7.4.2. Loading a Scenario from the Command Line 7-26
7.4.3. Editing Scenario Properties 7-26
7.4.4. Editing the Scenario Description 7-29
7.4.5. Sample Scenarios 7-29
7.5. Load Balancing Scenarios 7-30
7.5.1. Load Balancing an Existing Scenario 7-30
7.5.2. Loading a Load-Balanced Scenario 7-32

7.6. Running a Scenario 7-34
7.6.1. Changing the Simulation Speed 7-34
7.6.2. Pausing a Scenario 7-36
7.6.3. Rewinding a Scenario 7-36
7.6.4. Closing a Scenario 7-37
7.7. Reinitializing a Scenario 7-38
7.8. Checkpoints and Snapshots 7-38
7.8.1. Saving Checkpoints 7-39
7.8.2. Deleting Checkpoints 7-42
7.8.3. Creating Snapshots 7-42
7.8.4. Clearing the Snapshot List 7-43
7.9. Pausing a Scenario Automatically 7-44
7.9.1. Setting the Scenario End Time for a New Scenario 7-44
7.9.2. Setting the Scenario End Time for an Open Scenario 7-44
7.10. Running VR-Forces in Batch Mode 7-45
7.10.1. The Batch File 7-45
7.10.2. Creating a Batch File 7-46
7.10.3. Editing a Batch File 7-46
7.10.4. Running VR-Forces in Batch Mode 7-48
7.10.5. Recording Batch Scenarios 7-49
7.11. Sending Run and Pause Messages to Simulation Participants 7-50
7.12. Recording VR-Forces Simulations with the MAK Data Logger 7-50

Chapter 8. Scenario Events

8.1. Introduction to Scenario Events 8-2
8.2. Creating a Scenario Event 8-3
8.2.1. Adding Content to a Scenario Event 8-5
8.2.2. Adding Linked Events 8-6
8.3. Adding Intelligence Objects 8-8
8.4. Starting a Scenario Event 8-10
8.4.1. Controlling Scenario Events from a Plan 8-12
8.5. Ending a Scenario Event 8-13
8.6. Enabling and Disabling Scenario Events 8-14
8.7. Sorting the Event List 8-15
8.8. Deleting a Scenario Event 8-15
8.9. Showing and Hiding Scenario Events 8-16

Chapter 9. Target Detection and Combat Features

9.1. Displaying Spot Reports 9-2
9.1.1. Enabling or Disabling Spot Reports 9-3
9.1.2. Configuring the Spot Reports Viewpoint 9-4
9.1.3. Configuring the Spot Reports Certainty Level 9-7
9.1.4. Applying Spot Reports to Tactical Graphics 9-8
9.1.5. Displaying Labels for Spot Reports 9-8
9.1.6. Using Spot Reports in Tasks 9-8
9.1.7. Displaying Track Histories for Spot Reports 9-9

9.2. Displaying Tactical Data Link Messages 9-10
9.2.1. Configuring the Display of TDL Messages 9-11
9.2.2. Displaying Track Histories for TDL Messages 9-12
9.3. Managing Force Hostility Relationships 9-13
9.3.1. Changing a Force's Hostility in a Plan 9-15
9.4. Detecting Targets 9-16
9.4.1. Target Detection and Spot Reports 9-18
9.4.2. Lasing Targets 9-18
9.5. Engagement Zones 9-19
9.6. Electronic Warfare 9-20
9.7. Radar Coverage 9-20
9.7.1. Enabling the Display of Radar Coverage 9-22
9.7.2. Displaying Radar Coverage 9-22
9.7.3. Configuring Radar Coverage Displays 9-23
9.8. Using Sonar 9-24
9.8.1. Propulsion Noise and Sonar 9-25
9.9. Launching Counter Measures (Chaff and Flare) 9-26
9.10. Modeling Artillery Munitions 9-28

Chapter 10. Indirect Fire, Ballistic Missiles, and Munition Detonations

10.1. Introduction to Indirect Fire 10-2
10.1.1. Creating an Indirect Fire Event 10-3
10.1.2. Editing Indirect Fire Events 10-6
10.1.3. Deleting Indirect Fire Events 10-6
10.1.4. Configuring Indirect Fire Event Default Values 10-7
10.2. Ballistic Missiles 10-8
10.2.1. Firing Ballistic Missiles 10-9
10.2.2. Editing Missile Target Events 10-11
10.2.3. Deleting Missile Target Events 10-11
10.3. Creating Munition Detonations 10-12

Chapter 11. Setting Environment Conditions

11.1. Introduction to Environment Conditions 11-3
11.2. Setting the Date and Time of Day 11-3
11.2.1. Setting the Date 11-4
11.2.2. Setting the Time of Day 11-5
11.2.3. Choosing the Illumination Model 11-5
11.3. Specifying the Environment Conditions 11-6
11.3.1. Adding an Environment Condition 11-7
11.3.2. Editing an Environment Condition 11-8
11.3.3. Deleting an Environment Condition 11-8
11.4. Setting Weather Conditions 11-9
11.4.1. Setting the Wind Speed and Direction 11-9
11.4.2. Setting the Ambient Air Temperature 11-10
11.4.3. Setting Visibility (Fog) 11-11
11.4.4. Specifying Precipitation Type and Intensity 11-12
11.4.5. Specifying Cloud Cover 11-12

11.5. Creating Local Weather Zones 11-13
11.5.1. Creating a Weather Zone as a Tactical Graphic 11-15
11.5.2. Editing a Weather Zone's Environment 11-15
11.6. Creating and Editing Cloud Covers 11-17
11.6.1. Editing Cloud Covers 11-20
11.6.2. Enabling and Disabling Cloud Layers 11-20
11.6.3. Removing a Cloud Cover 11-21
11.6.4. Cloud Layer Parameters 11-21
11.7. Configuring Marine Conditions 11-22
11.7.1. Enabling Marine Effects 11-23
11.7.2. Configuring Marine Conditions 11-24
11.7.3. Enabling Tidal Stream Wakes 11-26
11.7.4. Setting Ocean Quality 11-27
11.7.5. Configuring Breaking Waves 11-27
11.8. Setting the Thermocline 11-29

Chapter 12. Scenario Files

12.1. Introduction 12-2
12.2. The Scenario File 12-3
12.2.1. Scenario Parameters 12-4
12.2.2. Specifying Pathnames for Scenario Files 12-7
12.2.3. Changing the Terrain Database for a Scenario 12-8
12.3. The Plan File 12-9
12.4. The Object Map File 12-9
12.5. Editing the Scenario Objects File 12-10
12.5.1. Saving Default Parameters to the Scenario Objects File 12-10
12.6. The Scenario Extras File 12-10
12.7. Temporary Scenario Directories 12-11

Section III Simulation Objects

Chapter 13. Introduction to Simulation Objects

13.1. Simulation Objects 13-2
13.2. How Simulation Objects are Identified 13-2
13.2.1. UUIDs 13-3
13.2.2. Simulation Object Names 13-3
13.2.3. Echelon IDs 13-4
13.2.4. Object IDs 13-4
13.2.5. Labels 13-4
13.3. How Simulation Objects are Organized 13-5
13.3.1. How Echelon IDs are Assigned 13-6
13.4. How Simulation Objects Communicate 13-6
13.4.1. Custom Communication Models 13-6
13.5. VR-Forces Uses a Component Architecture 13-7
13.5.1. Sensors 13-7
13.5.2. Controllers 13-8
13.5.3. Actuators 13-8
13.6. Simulation Object Behaviors and Tasks 13-8

13.7. Entity-Level Modeling and Aggregate-Level Modeling 13-9
13.8. Collision, Obstacle, and Feature Avoidance 13-11
13.9. Entity Movement and Soil Type 13-12

Chapter 14. Creating and Placing Objects

14.1. Creating Objects 14-3
14.2. Selecting the Object to Create 14-4
14.2.1. Selecting the Object to Create on an Object Palette 14-4
14.2.2. Selecting the Object to Create on the Create Menu 14-7
14.2.3. Filtering Object Creation Palettes 14-8
14.2.4. Draw Mode 14-11
14.3. Placing Objects 14-13
14.3.1. Placing an Object Using Default Values (Click to Create) 14-13
14.3.2. Specifying an Object's Properties Before You Create It (Click to Locate) 14-15
14.3.3. Default Placement of New Entities 14-17
14.3.4. Placing Entities Inside Buildings 14-17
14.3.5. Placing Simulation Objects on Other Simulation Objects 14-17
14.4. Specifying an Object's Altitude 14-18
14.4.1. Setting Altitude Dynamically 14-18
14.4.2. Setting Altitude in the Create Object or Edit Object Dialog Box 14-18
14.4.3. Specifying the Altitude for All of the Vertices in a Route 14-19
14.5. Specifying an Object's Heading Dynamically 14-20
14.6. Locking the Mouse to the Object Being Created 14-21
14.7. Simulation Object Resources 14-22
14.8. Moving Objects 14-23
14.8.1. Dragging an Object to a New Location 14-23
14.9. Editing Simulation Objects 14-25
14.9.1. Editing an Object's Name in an Object List 14-26
14.10. Copying and Pasting Objects 14-28
14.10.1. Copying Objects 14-28
14.10.2. Pasting Objects 14-29
14.10.3. Pasting Specific Entity Characteristics 14-29
14.11. Deleting Simulation Objects 14-32
14.12. Simulation Object Groups 14-32
14.13. Creating Random Versions of Similar Entity Types 14-33
14.14. Creating Cultural Features 14-33
14.14.1. Embedding Cultural Features in the Terrain 14-34
14.15. Creating Props 14-35
14.16. Adding an Object to the Favorites List 14-36
14.16.1. Removing an Object from the Favorites List 14-36

Chapter 15. Selecting Objects

15.1. The Objects List Panel 15-2
15.2. Selecting Simulation Objects, Tactical Graphics, and Props .. 15-4
15.2.1. Selecting Objects in the Window 15-5
15.2.2. Selecting Objects in the Objects List Panel 15-6
15.2.3. Selecting a Simulation Object in the Object Console Summary Panel 15-6
15.2.4. Unselecting Objects 15-7
15.3. Using Selection Groups 15-8
15.3.1. Creating a Selection Group 15-8
15.3.2. Selecting a Selection Group 15-9
15.3.3. Adding Objects to a Selection Group 15-10
15.3.4. Removing Objects from Selection Groups 15-10
15.3.5. Renaming a Selection Group 15-10
15.3.6. Deleting a Selection Group 15-10

Chapter 16. Viewing Information about Objects

16.1. Displaying Information About an Object 16-3
16.1.1. Information Dialog Boxes for Entity-Level Scenarios .. 16-3
16.1.2. Information Dialog Boxes for Aggregate-Level Scenarios 16-5
16.1.3. Information Dialog Boxes for Tactical Graphics 16-7
16.1.4. Viewing Information for Multiple Objects 16-8
16.2. Displaying Entity Labels 16-9
16.2.1. Displaying Text Labels for 3D Models 16-12
16.2.2. Turning Entity Labels On and Off 16-13
16.2.3. Pinning 3D Text Labels to the Window 16-13
16.2.4. Displaying Labels for 2D Icons 16-14
16.2.5. Configuring Shadowing and Background Color for 2D Labels 16-16
16.3. Using Extended Labels 16-17
16.3.1. Adding an Extended Label 16-18
16.3.2. Changing an Extended Label's Index 16-19
16.3.3. Deleting an Extended Label 16-19
16.4. Changing Force Colors 16-19
16.5. Displaying HAT Lines for Entities (3D Only) 16-20
16.6. Displaying Entity Resources 16-21
16.6.1. Viewing Numerical Resource Data 16-21
16.7. Displaying Bounding Volumes 16-22
16.7.1. Enabling and Disabling Bounding Volumes for Entities 16-22
16.7.2. Showing Bounding Volumes Only for Selected Entities 16-23
16.8. Configuring Object Console Messages 16-23
16.8.1. Setting the Notification Level for Console Messages .. 16-24
16.8.2. Viewing All Console Messages 16-25
16.8.3. Answering Questions from Scripted Tasks 16-26
16.8.4. Saving Console Messages 16-26
16.8.5. Displaying an Object Console Warning Icon 16-27

16.8.6. Clearing the Object Console 16-27
16.9. Viewing Object Counts 16-28

Chapter 17. Embarked and Embedded Entities

17.1. Embarking and Disembarking Simulation Objects 17-2
17.1.1. Embarking and Disembarking Simulation Objects Instantly 17-4
17.1.2. Disembarking Simulation Objects Using the Disembark Command 17-7
17.2. Embedded Entities 17-8
17.2.1. Configuring an Entity to Support Embedded Entities . 17-9
17.2.2. Assigning Tasks and Plans to Embedded Entities 17-11
17.2.3. Restoring an Entity that has Embedded Entities 17-11

Chapter 18. Generating Traffic (Pattern of Life)

18.1. Generating Traffic 18-2
18.2. Creating Spawn Points 18-3
18.2.1. Activating and Deactivating Spawn Points 18-4
18.3. Creating Sink Points 18-5
18.4. Spawn Patterns 18-6
18.4.1. How Spawn Events are Scheduled 18-7
18.4.2. Spawning Entities in Bursts 18-9
18.4.3. Using a Custom Plan 18-10
18.4.4. Default Spawn Patterns and Scenario-Specific Patterns 18-10
18.5. Creating a Spawn Pattern 18-11
18.5.1. Copying a Spawn Pattern 18-14
18.6. Editing a Spawn Pattern 18-15
18.7. Deleting a Spawn Pattern 18-16

Chapter 19. Crowds and Multiple Simulation Objects

19.1. Creating Multiple Simulation Objects and Crowds 19-2
19.2. Creating Crowds Using a Pedestrian Area 19-2
19.2.1. Adding Entities to a Pedestrian Area 19-3
19.3. Creating Crowd Units 19-3
19.3.1. Crowd Tasks 19-4
19.4. Creating Multiple Simulation Objects 19-5
19.4.1. Adding an Area for Multiple Simulation Object Creation 19-8
19.4.2. Specifying the Simulation Objects to Create 19-9
19.5. Pedestrian-Oriented Tactical Graphics 19-10
19.5.1. Using Civilian Visit Points 19-10

Chapter 20. Creating and Using an Order of Battle

20.1. Introduction to the Order of Battle 20-2
20.1.1. The Order of Battle Panel 20-3
20.2. Creating an Order of Battle 20-3
20.2.1. Adding Objects Directly to the Order of Battle 20-4
20.2.2. Adding Existing Objects to the Order of Battle 20-5

20.3. Creating Units in the Order of Battle Panel 20-5
20.4. Instantiating Members of an Order of Battle in a Scenario 20-6
20.4.1. The Order of Battle Palette 20-6
20.5. Removing Objects from an Order of Battle 20-6
20.6. Editing the Objects in an Order of Battle 20-7
20.7. Changing the Order of Objects in an Order of Battle 20-7
20.8. Exporting an Order of Battle 20-7
20.9. Importing an Order of Battle 20-8

Section IV Modeling Units

Chapter 21. Entity-Level Unit Modeling

21.1. Units in Entity-Level Scenarios 21-2
21.1.1. How Units Are Organized 21-3
21.1.2. How Units Move 21-4
21.1.3. How Units Carry Out Tasks 21-5

Chapter 22. How Aggregate-Level Modeling Works

22.1. The Aggregate Warfare Model 22-3
22.1.1. Unit Combat 22-4
22.1.2. Unit Footprints 22-5
22.1.3. Unit Posture 22-6
22.1.4. Unit Movement 22-8
22.1.5. Limited Munition Attack 22-9
22.1.6. Contamination Areas 22-9
22.1.7. Unit Sensors 22-10
22.2. Combat Engineering Objects 22-11
22.2.1. Concealed Obstacles 22-12
22.2.2. Creating Combat Engineering Objects 22-13
22.2.3. How Engineering Units Create Engineering Objects . 22-13
22.3. Logistics 22-15
22.3.1. Receiving Supplies 22-17
22.3.2. Providing Supplies 22-17
22.4. Electronic Warfare 22-18
22.4.1. Jamming Communications 22-18
22.4.2. Jamming Radar 22-19
22.4.3. Sensing Electronic Emissions 22-20
22.5. Air Combat 22-20
22.5.1. Loadouts 22-21
22.6. Air Missions 22-22
22.7. Fuel Use and Aerial Refueling 22-23
22.7.1. Aerial Refueling Systems 22-23
22.7.2. Refueling Missions 22-24
22.8. Air Bases and Air Missions 22-25
22.8.1. Preparing Missions 22-26
22.8.2. Landing Air Units 22-27
22.8.3. Repairing Aircraft 22-27
22.8.4. Quick Reaction Alert Status 22-28

Chapter 23. Creating and Controlling Units

23.1. Creating Units 23-2
23.1.1. Creating a Unit by Combining Existing Simulation Objects 23-3
23.1.2. Creating a Preconfigured Unit 23-5
23.1.3. Configuring the Unit Creation State 23-6
23.2. Selecting a Unit 23-6
23.3. Adding Simulation Objects to a Unit 23-7
23.4. Removing a Simulation Object from a Unit 23-8
23.5. Units and Unit State in Entity-Level Scenarios 23-8
23.5.1. How a Unit's State is Shown 23-9
23.5.2. How Changing Aggregation State Affects a Unit 23-10
23.6. Triggering Unit State Transitions 23-11
23.6.1. Aggregating and Disaggregating Units Manually 23-12
23.6.2. Configuring Automatic Aggregation and Disaggregation 23-12
23.6.3. Using Disaggregation Areas 23-12
23.7. Deleting a Unit 23-13

Chapter 24. Displaying Units

24.1. Selecting Units 24-2
24.2. Expanding and Collapsing Units 24-2
24.3. Displaying Ghosted Icons 24-4
24.4. Displaying Unit Icons and Bounding Volumes 24-8
24.4.1. Specifying the Color Scheme for Unit Bounding Volumes 24-10

Section V Tasks

V.1. Introduction to Tasks V-1

Chapter 25. Assigning Tasks

25.1. Assigning Tasks to Simulation Objects 25-3
25.1.1. Task Procedures 25-3
25.1.2. C++ Tasks and Scripted Tasks 25-3
25.1.3. Concurrent Task Execution 25-4
25.1.4. How Do I Know Which Simulation Objects Can Execute a Task? 25-6
25.1.5. Escaping the Task Assignment Process 25-6
25.1.6. Specifying Parameters for Tasks 25-6
25.1.7. Viewing Task Status 25-8
25.1.8. Filtering the Object Selection Lists 25-9
25.1.9. Skipping (Stopping) a Task 25-9
25.2. Assigning Tasks to Units 25-10
25.2.1. Convoy Tasks 25-10
25.2.2. Independently Tasking Unit Members 25-11
25.3. Reactive Tasks 25-11
25.3.1. Enabling Reactive Tasks 25-13
25.3.2. Disabling Reactive Tasks 25-14

25.3.3. Setting the Priority of a Reactive Task 25-15
25.3.4. Managing Reactive Tasks 25-16
25.3.5. Canceling a Reactive Task 25-17
25.4. Using Behavior Sets to Manage Scripts 25-18
25.4.1. Creating Behavior Sets 25-19
25.4.2. Editing Behavior Sets 25-20
25.4.3. Assigning a Behavior Set to a Force 25-20
25.5. Entity Movement On Roads and Off Roads 25-21
25.5.1. Road Driving Behavior 25-22
25.5.2. Specifying the Side of the Road to Drive On 25-22
25.6. Fixed-Wing Entity Tasks and Behaviors 25-23
25.6.1. Placement of Newly Created Fixed-Wing Entities 25-23
25.6.2. Fly Task Behavior is Different from Move Task Behavior 25-24
25.6.3. Specifying and Maintaining Altitude for Fixed-Wing Entities 25-24
25.6.4. Fixed-Wing Entity Movement on the Ground and in the Air 25-25
25.6.5. How Fixed-Wing Entities Take Off 25-26
25.6.6. How Fixed-Wing Entities Land 25-27
25.7. Rotary-Wing Entity Tasks and Behaviors 25-28
25.8. Suppressive Fire Tasks 25-30
25.8.1. Configuring Suppression Effects 25-32
25.9. Configuring Task Execution Rules 25-33
25.9.1. Configuring Action Categories 25-34

Chapter 26. Movement Tasks for Entity-Level Scenarios

26.1. Air Drop 26-3
26.2. Animated Movement 26-4
26.3. Come to Stop 26-4
26.4. Convoy Along 26-5
26.5. Convoy To 26-5
26.6. Fast Rope Insertion 26-6
26.7. Fixed-Wing Land 26-7
26.8. Fixed Wing Takeoff 26-7
26.9. Fly Altitude 26-8
26.10. Fly Heading 26-8
26.11. Fly Heading and Altitude 26-9
26.12. Follow Entity 26-10
26.12.1. How Fixed-Wing Entities Follow Entities 26-10
26.13. Generate Air Traffic From Flight Plans 26-11
26.14. Land at Airport Near Location 26-11
26.15. Move Along Route 26-12
26.16. Move Into Formation 26-13
26.17. Move to Altitude 26-14
26.18. Move to Depth 26-14
26.19. Move to Location 26-15
26.19.1. Adding Multiple Move to Location Tasks to a Plan 26-16
26.20. Move to Location (Plan Along Roads) 26-16
26.21. Move to Location (Plan Path) 26-17

26.22. Move to Waypoint 26-18
26.23. Move to Waypoint (Plan Along Roads) 26-19
26.24. Move to Waypoint (Plan Path) 26-20
26.25. Orbit 26-21
26.26. Orbit Object 26-21
26.27. Parallel Pattern 26-22
26.28. Patrol Route 26-23
26.29. Patrol Between 26-23
26.30. Pattern Hold (Location) 26-24
26.31. Pattern Hold (Waypoint) 26-24
26.32. Rotary Wing Airlift Cargo to Location 26-26
26.33. Rotary Wing Land 26-26
26.34. Rotary Wing Retrieve Cargo 26-27
26.35. Rotary Wing Unload Cargo 26-27
26.36. Sail Heading 26-28
26.37. Sweep Naval Mines 26-28
26.38. Turn to Heading 26-29
26.39. Zig Zag to Heading 26-29

Chapter 27. Engagement Tasks for Entity-Level Scenarios

27.1. Arm Mine at Depth 27-3
27.2. Attack Aircraft with Guns 27-3
27.3. Drop Naval Depth Charge 27-4
27.4. Drop Naval Depth Charge at Location 27-4
27.5. Drop Naval Mine 27-5
27.6. Drop Naval Mines Along Route 27-6
27.7. Execute Close Air Support 27-7
27.8. Explode Charge at Depth 27-9
27.9. Fire at Target 27-9
27.10. Fire Cruise Missile 27-10
27.11. Fire for Effect Tasks 27-11
    27.11.1. Firing Naval Guns 27-12
27.12. Lase Location 27-12
27.13. Lase Target 27-13
27.14. Launch Anti-Submarine Missile (Vertical) 27-13
27.15. Launch Ballistic Missile at Location 27-14
27.16. Launch Ballistic Missile at Target Object 27-14
27.17. Launch Counter Measures 27-15
27.18. Launch Smoke 27-15
27.19. Launch Torpedo 27-16
    27.19.1. Anti-ship (Fixed Depth) 27-16
    27.19.2. Anti-Submarine 27-17
27.20. Place IED 27-18
27.21. Provide Suppressive Fire 27-19
27.22. Provide Suppressive Fire at Location 27-20
27.23. Release Bomb Tasks 27-20
27.24. Strafe Ground Target 27-22
27.25. Throw Grenade Tasks 27-24
    27.25.1. Throw Grenade at Location 27-24
    27.25.2. Throw Grenade at Target 27-24

Chapter 28. Human and Crowd Tasks

28.1. Circle Around Location 28-3
28.2. Civilian Idle 28-3
28.3. Close Door 28-3
28.4. Close Window 28-4
28.5. Create Pedestrians 28-4
28.6. Crowd Along Line 28-4
28.7. Crowd Around Location 28-5
28.8. Crowd Around Object 28-5
28.9. Crowd in Front of Entity 28-6
28.10. DI-Guy Animation 28-7
28.11. DI-Guy Gesture 28-7
28.12. Disperse Crowd 28-8
28.13. Find Cover 28-8
28.14. Find Firing Position 28-9
28.15. Flee from Entities 28-11
28.16. Move Between Cover to Location 28-11
28.17. Open Door 28-12
28.18. Open Window 28-12
28.19. Parachute to Target 28-13
28.20. Protest Around Location 28-14
28.21. Protest Along Line 28-14
28.22. Wait in Queue 28-15
28.23. Wander 28-15
28.24. Wander In Area 28-16

Chapter 29. Embarkation, Wait, Radio, and Other Tasks

29.1. Embarkation Tasks 29-3
29.1.1. Disembark 29-3
29.1.2. Disembark All 29-3
29.1.3. Embark 29-4
29.1.4. Embarking Humans in Cars 29-10
29.2. Radio Tasks 29-11
29.2.1. Send Radio Set 29-11
29.2.2. Send Radio Task 29-13
29.2.3. Send Text Message 29-14
29.3. Wait Tasks 29-15
29.3.1. Wait 29-15
29.3.2. Wait Duration 29-15
29.3.3. Wait Elapsed 29-15
29.4. Tasks for Moving Articulated Parts 29-16
29.4.1. Close Car Door 29-16
29.4.2. Close Cargo Door 29-16
29.4.3. Deploy Refueling Boom 29-16
29.4.4. Lower Periscope 29-17
29.4.5. Open Car Door 29-17
29.4.6. Open Cargo Door 29-18
29.4.7. Raise Periscope 29-18
29.4.8. Stow Refueling Boom 29-18

29.5. Embedded Entity Tasks 29-19
29.5.1. Deploying Embedded Entities 29-19
29.5.2. Recovering Deployed Entities 29-19
29.5.3. Sector Search Operation 29-20
29.5.4. Deploying an Entity by Assigning a Task 29-21
29.6. Deploy Sonobuoy 29-22
29.7. Deploy Sonobuoys Along Route 29-22
29.8. Sonar Dip 29-23
29.9. Release Next Entity From Queue 29-23
29.10. User Task 29-24
29.11. Reactive Tasks 29-25
29.12. Background Processes 29-26

Chapter 30. Tasks for Aggregate-Level Scenarios

30.1. Introduction to Aggregate-Level Tasks 30-4
30.2. Activate Jammer 30-4
30.3. Air-to-Air Attack 30-4
30.4. Air-to-Ground Air Base Attack 30-4
30.5. Air-to-Ground Attack 30-5
30.6. Attack by Fire 30-5
30.7. Attack with Anti-Air Missile 30-6
30.8. Attack with Anti-Ship Missile 30-6
30.9. Attack with Land-Attack Missile 30-6
30.10. Attack with Torpedo 30-7
30.11. Automatic Air Defense 30-7
30.12. Biological Attack 30-7
30.13. Bomb Location 30-8
30.14. Breach Obstacles 30-9
30.15. Change MOPP Level 30-10
30.16. Change Posture 30-10
30.17. Chemical Attack 30-11
30.18. Construct Abatis 30-12
30.19. Construct Anti-Tank Ditch 30-12
30.20. Construct Barbed Wire 30-13
30.21. Construct Barricade 30-13
30.22. Construct Berm 30-14
30.23. Construct Booby Trap 30-14
30.24. Construct Bridge 30-15
30.25. Construct Dry Gap 30-16
30.26. Construct Fortified Area 30-17
30.27. Construct Fortified Line 30-18
30.28. Construct Minefield 30-18
30.29. Construct Strong Point 30-19
30.30. Deactivate Jammer 30-19
30.31. Destroy Bridge 30-20
30.32. Destroy Ditch 30-20
30.33. Destroy Explosive 30-20
30.34. Destroy Fortification 30-21
30.35. Destroy Obstacle 30-21
30.36. Disembark 30-21

30.37. Disembark All 30-21
30.38. Drop Fuel Tanks 30-21
30.39. Embark 30-22
30.40. Escort Air Unit 30-22
30.41. FASCAM 30-22
30.42. Halt Movement 30-22
30.43. Improve Booby Trap 30-23
30.44. Improve Breach 30-23
30.45. Improve Bridge 30-23
30.46. Improve Ditch 30-24
30.47. Improve Fortification 30-24
30.48. Improve Obstacle 30-24
30.49. Indirect Fire 30-25
30.50. Land at Air Base 30-25
30.51. Launch Flight 30-26
30.52. Launch QRA Aircraft 30-27
30.53. Move Along Route 30-28
30.54. Move Along Route Retrograde 30-28
30.55. Move to Location 30-28
30.56. Move to Location Retrograde 30-28
30.57. Move To Waypoint 30-28
30.58. Move to Waypoint Retrograde 30-28
30.59. Nuclear Attack 30-29
30.60. Patrol Route 30-29
30.61. Patrol Between 30-29
30.62. Prepare QRA 30-30
30.63. Refuel from Tanker 30-30
30.64. Refueling Mission 30-31
30.65. Send NBC Report 30-31
30.66. Send Obstacle Report 30-32
30.67. Send Radio Set 30-32
30.68. Send Radio Task 30-32
30.69. Send Text Message 30-32
30.70. User Task 30-33
30.71. Wait for Escort 30-33
30.72. Wait Tasks 30-33
30.73. Background Process List 30-34

Chapter 31. Creating Scripted Tasks and Sets

31.1. Introduction to Scripts 31-3
31.1.1. Scripted Tasks 31-4
31.1.2. Scripted Sets 31-5
31.1.3. Reactive Task Scripts 31-5
31.1.4. Background Process Scripts 31-5
31.1.5. Script Metadata 31-5
31.1.6. The Lua Scripting Language 31-5
31.2. Creating a New Script 31-6
31.2.1. Specifying the Script ID 31-11
31.2.2. Specifying a Menu Icon 31-12
31.2.3. Specifying an Extended Name 31-12

31.2.4. Specifying a Short Description 31-12
31.2.5. Specifying a Description 31-13
31.2.6. Specifying Script Parameters 31-14
31.2.7. Specifying a Menu Location 31-23
31.2.8. Specifying Action Categories 31-24
31.2.9. Configuring a Script's Availability to a Simulation Object Type 31-25
31.2.10. Specifying Equivalent Scripts for a Toolbar Button . 31-29
31.2.11. Specifying the Programming Language for a Script . 31-30
31.2.12. Configuring Scripted Tasks to Run Concurrently 31-31
31.2.13. Creating Scripted Sets 31-31
31.2.14. Creating Reactive Tasks 31-33
31.2.15. Creating Background Processes 31-34
31.3. Saving Scripts 31-36
31.4. Editing a Script 31-37
31.5. Filtering the List of Scripts 31-37
31.6. Organizing Scripts into Folders 31-38
31.6.1. Adding a Folder 31-38
31.6.2. Renaming a Folder 31-38
31.6.3. Deleting a Folder 31-38
31.6.4. Adding Scripts to a Folder 31-38
31.6.5. Removing a Script from a Folder 31-38
31.7. Exporting and Importing Scripts 31-39
31.7.1. Importing a Script Package 31-40
31.8. Copying a Script 31-41
31.9. Deleting a Script 31-41
31.10. Creating a System Script 31-42
31.10.1. Including System Scripts on the Task and Set Menus 31-42
31.11. Specifying a Script Editor 31-44
31.12. Editing Lua Files 31-44

Chapter 32. Writing Scripts

32.1. The VR-Forces Lua Interface 32-2
32.1.1. Lua Classes 32-2
32.1.2. Lua Modules 32-5
32.2. Script Loading and Execution 32-5
32.2.1. Script Entry Points 32-6
32.2.2. The Script Execution Sequence 32-8
32.2.3. Limitations for Checkpointing Scripts 32-11
32.2.4. Editing Scripts While a Scenario is Running 32-13
32.3. Tasks and Subtasks 32-14
32.3.1. Monitoring the Status of Tasks and Subtasks 32-15
32.3.2. Stopping Tasks 32-15
32.3.3. Task Parameters 32-16
32.4. Geometry 32-19
32.4.1. *Location3D* 32-19
32.4.2. *Vector3D* 32-19
32.4.3. *VectorOffset3D* 32-22
32.4.4. *VectorGeoc3D* 32-22

32.5. Coding Messages for Translation 32-23
32.6. Error Detection 32-23
32.7. Interactive Scripts 32-25
32.8. A Basic Scripted Task 32-25
32.8.1. Create and Move to Waypoint Metadata 32-26
32.8.2. The Create and Move to Waypoint Lua Script 32-28
32.9. A Simple Scripted Set 32-30
32.10. A Simple Reactive Task 32-31
32.11. Background Processes 32-32

Chapter 33. Using Qt Designer for Script Dialog Boxes

33.1. Introduction to Using Qt Designer for Scripts 33-2
33.2. Selecting Qt Designer for Dialog Box Layout 33-2
33.3. Adding Parameters When Using Qt Designer 33-2
33.3.1. Specifying Choices for the Choice Parameters in Qt Designer 33-6
33.3.2. Specifying Categories for a Simulation Object in Qt Designer 33-7
33.4. Adding Design Elements 33-8

Section VI Set Data Requests

VI.1. Setting Simulation Object State and Attributes VI-1

Chapter 34. Setting the State of Entity-Level Simulation Objects

34.1. Active Sonar Mode 34-4
34.2. Add Jamming Target 34-4
34.3. AI Enabled 34-5
34.4. Aiming 34-6
34.5. Altitude 34-7
34.6. Appearance 34-8
34.7. Armed 34-9
34.8. Assume Unknown is Hostile 34-9
34.9. Attention Grabbing 34-9
34.10. Automatically Close Doors 34-10
34.11. Capabilities 34-10
34.12. Collision Avoidance Types 34-11
34.13. Concealed 34-12
34.14. Counter Measures Auto Launch 34-12
34.15. Current Speed 34-13
34.16. Destroyed 34-13
34.17. Detonation Fuse Type 34-14
34.18. DI-Guy Characteristics (Appearance, Head, Body Weight) 34-15
34.19. DI-Guy Hand Item 34-15
34.20. Disembarked 34-16
34.21. Embarked 34-16
34.22. Emitter 34-17
34.23. Engagement Zones 34-18
34.24. Force 34-18
34.25. Formation 34-19

34.26. Heading 34-20
34.27. Hide Occupants 34-20
34.28. IFF 34-21
34.29. Invisible 34-22
34.30. Invulnerable 34-22
34.31. Jammer State 34-22
34.32. Jamming Mode 34-23
34.33. Lase Autonomous 34-24
34.34. Laser Code 34-24
34.35. Laser Power and Wavelength 34-25
34.36. Location 34-26
34.37. Look At 34-26
34.38. Markings 34-27
34.39. Navigation Enabled 34-27
34.40. Navigation Preferences 34-28
34.41. Notify Level 34-28
34.42. Orbit 34-29
34.43. Ordered Speed 34-30
34.44. Posture (Human) 34-31
34.45. Preferred Trajectory Height 34-32
34.46. Radar Mode 34-32
34.47. Remote Sensor 34-33
34.48. Remove Jamming Target 34-33
34.49. Reorganize 34-33
34.50. Resources 34-34
34.51. Restore 34-34
34.52. Rules of Engagement 34-35
34.52.1. How Fire-When-Fired-Upon Works 34-35
34.53. Sector Blanking 34-36
34.54. Sector of Responsibility 34-37
34.55. Sensor Aim 34-38
34.56. Sensor Enabled 34-38
34.57. Sensor Field of View Color 34-39
34.58. Sensor Zoom 34-40
34.59. Sonar Depth 34-40
34.60. Spot Reports 34-41
34.61. Stow Hand Item 34-42
34.62. Superior 34-42
34.63. Surrendered 34-42
34.64. Synchronize Laser Code 34-43
34.65. Target 34-43
34.66. Tasked by Superior 34-44
34.67. TDL Messaging 34-44
34.68. Tracer Use 34-44
34.69. Unit State 34-45

Chapter 35. Setting the State of Aggregate-Level Simulation Objects

35.1. Activity 35-3
35.2. Ammunition 35-3
35.3. Ammunition Pacing/Tracking 35-4

35.4. Assign Aircraft 35-4
35.4.1. Editing Aircraft Assignments 35-5
35.4.2. Deleting Aircraft Assignments 35-6
35.5. Assume Unknown is Hostile 35-6
35.6. Capabilities 35-6
35.7. Concealed 35-6
35.8. Contamination 35-6
35.9. Current Speed 35-7
35.10. Destroyed 35-7
35.11. Disembarked 35-7
35.12. Embarked 35-7
35.13. Emitter 35-7
35.14. Engagement Zones 35-7
35.15. Engagement Results 35-7
35.16. Equipment Pacing/Tracking 35-8
35.17. Facilities 35-8
35.18. Food, Water, Fuel, Oil and Lubricant 35-9
35.19. Food, Water Fuel, Oil and Lubricant Pacing/Tracking 35-10
35.20. Force 35-10
35.21. Has Drag Chute 35-10
35.22. Heading 35-10
35.23. Health 35-11
35.24. IFF 35-11
35.25. Loadout 35-11
35.26. Location 35-11
35.27. MOPP Level 35-12
35.28. Morale 35-12
35.29. Notify Level 35-12
35.30. Ordered Speed 35-12
35.31. Percent Complete 35-12
35.32. Personnel and Supplies 35-13
35.33. Posture (Unit) 35-13
35.34. Preferred Targets 35-14
35.35. Radar Mode 35-14
35.36. Reorganize 35-14
35.37. Resources 35-14
35.38. Resources Pacing/Tracking 35-14
35.39. Restore 35-15
35.40. Resupply Mode 35-15
35.41. Role 35-15
35.42. Rules of Engagement 35-15
35.43. Sector of Responsibility 35-15
35.44. Sensor Enabled 35-16
35.45. Spot Reports 35-16
35.46. Superior 35-16
35.47. Supplying 35-16
35.48. Tasked by Superior 35-16
35.49. TDL Messaging 35-16
35.50. Weapons Pacing/Tracking 35-17

Section VII Plans

VII.1. Introduction to Plans VII-1

Chapter 36. Plan Concepts

36.1. Introduction to Individual Plans and Global Plans 36-2
36.2. Conditional Statements 36-3
36.2.1. The If Statement 36-5
36.2.2. Triggers 36-6
36.2.3. While Statements 36-8
36.2.4. Wait Until Statements 36-8
36.2.5. Conditional Tests 36-9
36.3. Plan Variables 36-14
36.4. Viewing Plans 36-15
36.5. Considerations for Creating Plans 36-16
36.5.1. Considerations for Using Triggers 36-16
36.5.2. Using Concurrent Tasks in Plans 36-17
36.5.3. Moving In Formation 36-17
36.5.4. Using the Tasked-By-Superior Request in a Plan 36-17
36.5.5. Following Simulation Objects 36-17
36.5.6. Using Non-VR-Forces Simulation Objects in Plans 36-18

Chapter 37. Writing Plans

37.1. Writing Plans for Simulation Objects 37-3
37.1.1. Adding a Task or Set Data Request to a Plan 37-4
37.1.2. Adding an If Block to a Plan 37-5
37.1.3. Adding a While Block to a Plan 37-6
37.1.4. Adding a Wait Until Statement to a Plan 37-6
37.1.5. Adding a Trigger to a Plan 37-7
37.1.6. Registering Triggers 37-9
37.1.7. Cleaning Up Unused Triggers 37-10
37.1.8. Editing a Statement 37-11
37.1.9. Deleting Statements from a Plan 37-12
37.2. Printing Plans 37-12
37.3. Saving Changes to a Plan 37-12
37.4. Building Condition Statements 37-13
37.4.1. Building a Condition Statement that Has One Condition 37-13
37.4.2. Building a Complex Condition 37-15
37.4.3. Editing Condition Statements 37-22
37.4.4. Specifying Names or Patterns in Condition Statements 37-23
37.5. Using Plan Variables 37-26
37.6. Creating Global Plans 37-27
37.6.1. Controlling When Global Plans Run (Quick Launch) . 37-29
37.6.2. Opening an Information Dialog Box for a Global Plan 37-29
37.7. Adding Commands to a Plan 37-30
37.7.1. Adding Tasks and Sets from the Commands Menu 37-31
37.7.2. Sending Text Messages from the Commands Menu ... 37-34
37.7.3. Sending Console Messages 37-35

37.7.4. Creating Objects from Within a Plan 37-35
37.7.5. Deleting Objects from Within a Plan 37-35
37.7.6. Issuing a Plan 37-36
37.7.7. Sending View Control Messages 37-37
37.8. Writing Plans for Units 37-41
37.9. Writing a Plan for Multiple Simulation Objects 37-41
37.10. Writing Plans for Remote Entities 37-41
37.11. Copying Plans and Plan Statements 37-42
37.12. Restarting a Plan 37-43
37.13. Abandoning a Plan 37-44

Section VIII Tactical Graphics

Chapter 38. Introduction to Tactical Graphics

38.1. Introduction 38-2
38.2. Points 38-4
38.3. Phase Lines 38-4
38.4. Routes 38-5
38.5. Areas 38-5
38.6. Obstacles 38-5
38.7. Displaying Tactical Graphics 38-6
38.7.1. Showing and Hiding Tactical Graphics 38-7
38.7.2. Showing and Hiding Tactical Graphics Labels 38-8
38.7.3. Displaying Height-Above-Terrain Lines for Vertices ... 38-8

Chapter 39. Overlays

39.1. Creating and Editing Overlays 39-2
39.1.1. Creating an Overlay 39-3
39.1.2. Selecting Overlays 39-3
39.1.3. Locking an Overlay 39-4
39.1.4. Hiding the Objects on an Overlay 39-4
39.1.5. Changing an Overlay's Name 39-5
39.1.6. Deleting an Overlay 39-5
39.2. Saving and Loading Tactical Graphics and Overlays 39-6
39.2.1. Loading Tactical Graphics and Overlays 39-6
39.3. Exporting Tactical Graphics as Shapefiles 39-7
39.3.1. Importing Shapefiles into Overlays 39-7

Chapter 40. Creating and Editing Tactical Graphics

40.1. Creating Tactical Graphics 40-2
40.1.1. How Areas Are Drawn 40-2
40.1.2. Creating Freehand Lines 40-3
40.1.3. Drawing Circles and Ellipses 40-4
40.1.4. Drawing Boxes 40-4
40.1.5. Creating Volumetric (Extruded) Tactical Graphics 40-5
40.1.6. Adding Text to an Overlay 40-5
40.1.7. Assigning Tactical Graphics to an Overlay 40-6
40.1.8. Displaying a Terrain Profile When You Create a Line . 40-7
40.1.9. Publishing Tactical Graphics 40-8

40.2. Creating a Computed Route 40-9
40.3. Creating an Intel Collection Area 40-11
40.4. Creating Nav Edges 40-12
40.5. Creating Cloud Layers 40-14
40.6. Attaching Tactical Graphics to a Simulation Object 40-16
40.6.1. Attaching a Tactical Graphic Automatically 40-16
40.7. Deleting Tactical Graphics 40-17
40.8. Editing Tactical Graphics 40-17
40.8.1. Editing Vertices 40-18
40.8.2. Resizing Boxes and Text Objects 40-20
40.8.3. Rotating an Ellipse 40-20
40.8.4. Changing the Direction of a Line 40-21
40.8.5. Changing the Color of a Tactical Graphic 40-21
40.8.6. Using the Force Color for Vertices 40-21
40.8.7. Changing Linear and Areal Style Properties 40-23
40.9. Active and Inactive Tactical Graphics 40-24

Chapter 41. Setting the State of Tactical Graphics

41.1. Set Data Requests for Tactical Graphics 41-2
41.2. Activation 41-2
41.3. Allow Crossing 41-3
41.4. Arrowhead Style 41-3
41.5. Color 41-4
41.6. Fill Style 41-4
41.7. Force 41-4
41.8. Line Width 41-4
41.9. Notify Level 41-5
41.10. Pen Style 41-5

Chapter 42. Remote Graphics

42.1. Displaying Remote Graphics 42-2
42.1.1. Viewing a List of Remote Graphics 42-4

Section IX Visual and Audio Effects

Chapter 43. Displaying Tactical Information and Effects

43.1. Displaying Trailing and Decal Effects (3D Only) 43-3
43.1.1. Specifying Models for Trailing Effects and Decal Effects 43-4
43.2. Displaying Track Histories 43-5
43.2.1. Configuring Track Histories 43-6
43.2.2. Displaying a Terrain Profile for a Track History 43-7
43.3. Viewing Fire and Detonation Effects 43-8
43.4. Viewing Fire and Detonation Lines 43-9
43.5. Viewing Unit Combat Graphics 43-10
43.6. Visualizing Tasks 43-12
43.6.1. Displaying Task Visualizations for an Entity 43-13
43.6.2. Pinning Task Visualizations for an Entity 43-13
43.7. Displaying Sensor Contact Lines 43-14

43.7.1. Enabling the Display of Sensor Contact Lines 43-14
43.7.2. Specifying which Simulation Objects Display Sensor Contact Lines 43-15
43.7.3. Configuring Sensor Contact Lines 43-16
43.8. Displaying Threat and Sensor Range Rings 43-16
43.8.1. Enabling and Disabling the Display of Range Rings .. 43-18
43.8.2. Pinning Range Rings for a Simulation Object 43-19
43.8.3. Configuring the Display of Range Rings 43-19
43.9. Displaying Laser Designators 43-22
43.10. Displaying Supply Lines 43-23
43.11. Displaying Tactical Smoke 43-23
43.12. Displaying Electromagnetic Emissions 43-24
43.13. Displaying Radio Communication Lines 43-25
43.13.1. Enabling Radio Communications Graphics 43-26
43.13.2. Configuring the Lifetime and Display Mode 43-27
43.13.3. Configuring the Color of Radio Communication Lines 43-28

Chapter 44. Visualizing 2D Icons and 3D Models

44.1. Simulation Object Icons 44-2
44.1.1. Changing the Size of 2D Icons 44-3
44.1.2. Rotating 2D Icons to a Simulation Object's Heading 44-4
44.2. Exaggerating the Scale of 3D Entity Models 44-5
44.3. Hiding 3D Models 44-8
44.4. Displaying DI-Guys in Vehicles 44-8

Chapter 45. Lighting Effects

45.1. Displaying Lighting Effects 45-2
45.2. Enabling and Disabling Lighting 45-3
45.2.1. Configuring Ambient and Diffuse Lighting 45-4
45.2.2. Specifying the Emissive Lighting Scale 45-5
45.3. Displaying Dynamic Lighting 45-6
45.3.1. Daylight Control of Lights 45-7
45.4. Displaying Ocean Planar Reflections 45-8
45.4.1. Configuring Ocean Planar Reflections 45-10
45.5. Lens Flare 45-11
45.6. Displaying Crepuscular (Sun) Rays 45-13
45.7. Configuring the Skybox Cube Map 45-15
45.8. Flashlight Mode 45-16
45.9. Using High Dynamic Range Lighting 45-17
45.10. Disabling Lighting and Clouds 45-19
45.11. Displaying Shadows (3D Only) 45-20
45.11.1. Enabling Shadows on Trees 45-22
45.11.2. Configuring Shadows 45-23

Chapter 46. Marine Effects

46.1. Displaying Water Droplets and Splash Effects 46-2
46.1.1. Enabling Sea Spray 46-4
46.1.2. Setting the Size of Water Droplets 46-5

46.1.3. Enabling and Disabling Rotor Wash 46-6
46.1.4. Enabling High Quality Rotor Wash 46-6
46.2. Displaying Wakes and Spray Effects 46-7
46.3. Enabling Buoyancy for Surface Entities 46-8

Chapter 47. Using CameraFX Sensor Views

47.1. Using Sensors 47-2
47.2. Enabling Sensors 47-3
47.3. Configuring Sensors 47-4
47.4. Adding New Sensors 47-5

Chapter 48. Intervisibility

48.1. Intervisibility (Line-of-Sight) 48-2
48.2. Types of Intervisibility 48-3
48.2.1. Point-to-Point Intervisibility 48-3
48.2.2. Intervisibility Fan 48-4
48.2.3. Simulation Object Intervisibility 48-4
48.3. Intervisibility Objects can be Transient or Permanent 48-5
48.3.1. Creating a Transient Intervisibility Object 48-6
48.3.2. Creating a Permanent Intervisibility Object 48-8
48.4. Editing an Intervisibility Object 48-9
48.5. Configuring Intervisibility Lines and Fans 48-10
48.6. Pinning an Intervisibility Object to a Simulation Object 48-12
48.7. Showing and Hiding Intervisibility Lines 48-13
48.8. Displaying Intervisibility Line Information 48-14
48.9. Displaying an Intervisibility Line's Terrain Profile 48-15
48.10. Deleting Intervisibility Objects 48-16

Chapter 49. Sound Effects

49.1. Introduction to Sound Effects 49-2
49.2. Enabling Sound Effects 49-2
49.2.1. Configuring Sound Effects 49-3
49.3. Associating Sound Files with Entities and Events 49-4

Section X The Observer and Viewing Scenarios

Chapter 50. The Observer and Observer Modes

50.1. Switching Between the 2D and 3D Projections 50-2
50.2. Observer Modes 50-2
50.3. Changing the Observer Mode 50-3
50.4. Creating and Editing Observer Modes 50-4
50.4.1. Creating an Observer Mode 50-4
50.4.2. Deleting an Observer Mode 50-5
50.4.3. Editing an Observer Mode 50-5
50.4.4. Renaming an Observer Mode 50-6
50.5. Adding an Observer 50-6
50.5.1. Adding an Observer on the Observer Settings Page 50-7

50.5.2. Adding an Observer in the Display Engine Configuration Editor 50-7
50.6. Removing an Observer 50-9
50.7. Introduction to XR Mode 50-10
50.8. Model Sets 50-11
50.8.1. Changing the Model Set 50-11
50.9. Communicating Information About the Observer 50-11
50.9.1. Making the Observer Visible to Other Applications ... 50-11
50.9.2. Viewing and Attaching to Other Observers 50-12
50.10. Displaying Models for Remote Observers 50-12

Chapter 51. Moving the Observer

51.1. Moving the Observer 51-3
51.1.1. Using the Compass to View the Observer's Heading 51-4
51.1.2. Editing the Compass's Attributes 51-5
51.2. Observer Movement Functions and Keyboard Mappings 51-6
51.2.1. 3D Observer Movement Coordinate Systems 51-7
51.2.2. 2D Observer Coordinate System 51-7
51.2.3. The Attached Context and Observer Movement (3D Only) 51-8
51.3. Moving the Observer from the Keyboard in the 3D View 51-10
51.3.1. Navigation Options 51-12
51.3.2. Orbiting 51-13
51.3.3. Moving the Camera 51-13
51.4. Moving the Observer Using the Mouse (3D View) 51-14
51.4.1. Dragging the Terrain 51-14
51.4.2. Changing the Observer's Orientation with the Mouse 51-16
51.4.3. Dragging the View 51-16
51.4.4. Orbiting the Terrain with the Mouse 51-16
51.4.5. Teleporting the Observer 51-17
51.5. Moving the Observer in the 2D View 51-18
51.5.1. Moving the Observer from the Keyboard in the 2D View 51-18
51.5.2. Moving the Observer Using the Mouse in the 2D Projection 51-19
51.6. Zooming the Observer 51-19
51.6.1. Zooming the Observer in the 3D Projection 51-20
51.6.2. Zooming the Observer in the 2D Projection 51-21
51.6.3. Setting the Scale Factor for Paged LODs 51-21
51.6.4. Setting the Magnification LOD Angle for Paged LODs 51-22
51.6.5. Zooming In to the Ocean Over Long Distances 51-23
51.6.6. Zooming to Extents 51-23
51.7. Centering the Observer on a Simulation Object 51-24
51.7.1. Centering on an Object from the Object Console Summary Panel 51-24
51.7.2. Centering on an Object From an Information Panel .. 51-24
51.8. Simulating Camera Jitter 51-25
51.9. Forcing Orientation North 51-26

51.10. Enabling and Disabling View Constraints (3D Only) 51-27
51.11. Controlling Navigation Speed 51-27
51.11.1. Enabling or Disabling Speed Scaling 51-28
51.11.2. Changing the Observer's Movement Speed 51-29
51.12. Saving and Recalling Views 51-30
51.12.1. Recalling a View 51-31
51.12.2. Deleting Views 51-32
51.12.3. Loading Views Automatically When You Load a Terrain 51-32
51.12.4. Specifying the Scenario Startup View 51-33
51.12.5. Saving Views to a File 51-34
51.12.6. Loading Views 51-34
51.13. Controlling the Observer from Other Applications 51-35
51.13.1. Enabling the Processing of View Control Messages .. 51-35

Chapter 52. Attaching the Observer to Objects

52.1. Attaching the Observer to a Simulation Object or Prop 52-2
52.1.1. Attaching the Observer to Another Observer 52-4
52.1.2. Attaching to an Articulated Part 52-5
52.1.3. Attaching the Observer in Mimic Track or Tether Track Mode 52-5
52.1.4. Attaching the Observer in Mimic Location Track or Tether Location Track Mode 52-5
52.1.5. Filtering the Object Attachment List 52-6
52.1.6. Detaching from a Simulation Object 52-6
52.1.7. Hiding the Attached Model (3D Projection Only) 52-7
52.1.8. Attaching to Props 52-7
52.1.9. Improving Performance when Attaching to Ground Objects 52-7
52.2. Attach Modes in the 3D View 52-8
52.2.1. Absolute Mode 52-8
52.2.2. Follow Mode 52-9
52.2.3. Mimic Mode 52-10
52.2.4. Mimic Track Mode 52-11
52.2.5. Mimic Location Track Mode 52-11
52.2.6. Space Follow Mode 52-12
52.2.7. Tether Mode 52-13
52.2.8. Tether Track Mode 52-14
52.2.9. Tether Location Track Mode 52-14
52.2.10. Track Mode 52-15
52.3. Attach Modes in the 2D View 52-16
52.4. Attaching the Observer at Startup 52-16

Chapter 53. Inset Views, Sensor FOVs, HUDs, and Depth of Field

53.1. Inset Views 53-2
53.2. Displaying Cockpit Displays 53-4
53.3. Displaying Instrument Panels 53-6
53.3.1. Configuring Entities to Use Instrument Panels 53-7

53.4. Displaying Sensor Views 53-8
53.4.1. Displaying a Sensor Field of View Cone 53-9
53.4.2. Controlling the Sensor View 53-10
53.5. Using the Depth of Field Effect 53-13
53.5.1. Configuring Depth of Field 53-15

Section XI Terrain

Chapter 54. Introduction to Terrain Usage in VR-Forces

54.1. Composing Terrains (and Creating MTF Files) 54-2
54.2. Supported File Formats 54-2
54.2.1. Terrain Data 54-3
54.2.2. Elevation Data 54-3
54.2.3. Supported Image Formats 54-4
54.2.4. Supported Feature Data Formats in the Front-end 54-4
54.3. Feature Layers and Props 54-4
54.3.1. Geometric Representation of Point Feature Data 54-5
54.3.2. Feature Data Types Supported by the Back-end 54-5
54.4. Coordinate Systems 54-6
54.4.1. The UTM Coordinate System 54-6
54.4.2. Geocentric Terrain Databases 54-6
54.5. Loading Supported Databases 54-7
54.6. Using Large Terrain Databases 54-7
54.6.1. Terrain Paging 54-8
54.6.2. Terrain Streaming 54-9
54.7. Using Ground Clamping 54-9
54.7.1. Configuring Ground Clamping 54-11

Chapter 55. Displaying Terrain Information

55.1. Introduction 55-2
55.2. Coloring Terrain Based on Elevation 55-3
55.2.1. Configuring Elevation Color Settings 55-4
55.3. Displaying Grid Lines 55-6
55.3.1. Configuring Grid Lines 55-7
55.4. Displaying Models and Terrain in Wireframe Mode 55-8
55.5. Filtering Earth File Layers 55-10
55.6. Displaying Feature Data (2D) 55-11
55.6.1. Displaying a List of Vector Features 55-13
55.6.2. Changing the Colors Used for Feature Data 55-14
55.7. Displaying Vector Features 55-15
55.8. Displaying Coordinates and Soil Type for a Location 55-16
55.9. Viewing the Map Scale 55-17
55.9.1. The Map Scale Overlay 55-17
55.10. Using Terrain Draw Areas to View Terrain Polygons 55-18
55.11. Configuring Visible Surfaces (GDB Soil Types) 55-19
55.12. Terrain Databases Provided with VR-Forces 55-21

Chapter 56. Using Terrain Profiles

56.1. Terrain Profiles 56-2
56.1.1. Displaying a Line's Terrain Profile 56-3
56.1.2. Configuring Terrain Profiles 56-3
56.2. Analyzing Terrain Using a Terrain Profile Line 56-5

Chapter 57. Composing Terrains

57.1. Creating a Composed Terrain 57-2
57.1.1. Considerations and Limitations for Building Terrains . 57-2
57.1.2. Saving a Terrain 57-3
57.2. Adding Elevation Data (Terrain Patches) to a Terrain 57-4
57.3. Adding Images to a Terrain 57-6
57.3.1. Changing the Display Order of Raster Maps 57-9
57.4. Adding a Feature Layer 57-11
57.5. Adding a Dynamic Ocean Layer 57-14
57.5.1. Extracting Water Textures and Adding an Ocean Layer 57-15
57.5.2. Adding a Dynamic Ocean Layer Directly 57-16
57.5.3. Setting the Ocean LOD Elevation 57-17
57.5.4. Configuring the Ocean Height Map 57-17
57.6. Adding Props to a Terrain 57-20
57.6.1. Extracting Props from a Terrain Patch 57-20
57.6.2. Adding Props from a Feature Layer 57-22
57.6.3. Viewing a List of Props 57-24
57.6.4. Selecting Props 57-24
57.6.5. Setting the Opacity of Props 57-25
57.6.6. Changing a Prop's Type 57-26
57.6.7. Changing a Prop's Position or Orientation 57-26
57.6.8. Changing a Prop's Model Definition 57-26
57.6.9. Exporting Props and SpeedTrees to a Shapefile 57-27
57.6.10. Importing Props and SpeedTrees from Shapefiles 57-27
57.7. Specifying Simulation Ocean Settings 57-28

Chapter 58. Paged and Streaming Terrains

58.1. Connecting to Terrain Servers 58-2
58.1.1. Adding Terrain Server Connections 58-4
58.1.2. Connecting to Terrain Servers through a Proxy 58-5
58.2. Loading MetaFlight Terrains 58-5
58.3. Preprocessing Paged Terrains 58-6
58.4. Manually Paging-In Terrain 58-6

Chapter 59. Loading Data Using Earth Files

59.1. Earth Files 59-3
59.2. A Simple Earth File 59-3
59.2.1. Including External Files into Earth Files 59-6
59.3. Supported Drivers 59-7
59.4. Loading Multiple DTED Files 59-7
59.5. Processing Streamed Data 59-8

59.6. Adding Feature Layers to an Earth File 59-9
59.6.1. Configuring Point Features 59-11
59.6.2. Configuring Linear Features 59-12
59.6.3. Configuring Areal Features 59-13
59.7. Extruded Buildings 59-14
59.8. Using Cut-in Sites for High Resolution Insets 59-15
59.8.1. Using the Boundary Generation Tool 59-16
59.9. Earth File Options 59-17
59.10. Referencing an Earth Gravitational Model (EGM) 59-18
59.11. Differential Processing of Elements and Attributes 59-18
59.11.1. Differential Processing of model Elements 59-19
59.12. Indexing Elevation Layers 59-20
59.12.1. Configuring Elevation Layers for Indexing 59-20
59.12.2. Generating the Elevation Layer Indexes 59-21
59.13. Streaming Buoys and Beacons 59-22
59.13.1. Specifying Model Definitions for Streamed Buoys 59-23
59.13.2. Modeling Buoys Using Texture Atlases 59-26
59.13.3. Configuring Streaming Beacons 59-27
59.13.4. Configuring Streamed Navigation Lights 59-28
59.13.5. Configuring Seasonal Buoys and Beacons 59-29
59.13.6. Streaming Daymarks 59-29
59.13.7. Streaming Topmarks 59-30
59.13.8. Generating Signal Sequences 59-30

Chapter 60. Configuring CDB Terrains

60.1. Loading CDB Databases 60-2
60.2. Configuring CDB earth Files 60-2
60.2.1. Configuring the Elevation Dataset (001) 60-3
60.2.2. Configuring the Imagery Dataset (004) 60-5
60.2.3. Configuring the GeoSpecific (GS) Dataset (100) 60-6
60.2.4. Configuring the GeoTypical (GT) Dataset (101) 60-13
60.2.5. Configuring the Extended Imagery Dataset (900) 60-15
60.3. CDB Configuration Files 60-17
60.4. Caching CDB Data (cdbtool) 60-19

Chapter 61. Generating Navigation Data

61.1. Introduction 61-2
61.1.1. How Navigation Data is Generated 61-2
61.1.2. Path Finding 61-4
61.2. Navigation Areas 61-7
61.2.1. Sectorizing Navigation Areas 61-7
61.2.2. Creating a Navigation Area 61-8
61.2.3. Editing a Navigation Area 61-11
61.2.4. Reverting Edits to a Navigation Area 61-11
61.2.5. Displaying Navigation Areas 61-11
61.2.6. Removing a Navigation Area 61-11
61.2.7. Displaying Information about a Navigation Area 61-12
61.3. Generating Navigation Data 61-13
61.3.1. Navigation Profiles 61-13

61.3.2. Generating Navigation Data for a Navigation Area ... 61-14
61.3.3. Using Newly Generated Navigation Data ... 61-16
61.3.4. Automatic Regeneration of Navigation Data ... 61-16
61.4. Generating Navigation Data for Entities ... 61-17
61.5. Importing Navigation Areas and Navigation Data ... 61-19
61.6. Displaying Navigation Data ... 61-20
61.6.1. Opening Navigation Lab from an Information Dialog Box ... 61-21
61.6.2. Showing Simulation Object Movement in Navigation Lab ... 61-22

Chapter 62. Dynamic Terrain

62.1. Introduction to Dynamic Terrain ... 62-2
62.2. Manually Changing the Terrain ... 62-2
62.2.1. Resetting Dynamic Terrain Changes ... 62-3
62.3. Removing Dynamic Terrain Damage ... 62-3
62.4. Creating Dynamic Terrain Areas ... 62-3
62.5. Dynamic Terrain Switch Types ... 62-4

Chapter 63. Processing MetaFlight Files

63.1. Building Efficient MetaFlight Terrains for VR-Forces ... 63-2
63.2. Processing MetaFlight Files for Use in VR-Forces ... 63-2
63.3. Splitting Datasets into Individual MetaFlight Files ... 63-3
63.4. Converting Virtual Texture Datasets to Tiled Textures ... 63-4
63.4.1. The mftTool ... 63-4
63.4.2. Using the mftTool to Process MetaFlight Files ... 63-6
63.4.3. Convert Geometry Grid Datasets with Tiled Textures ... 63-7
63.5. Convert Source Data into MEDF Format ... 63-7
63.6. Create an MTF File for the MetaFlight Terrain ... 63-8

Chapter 64. Terrain Performance and Configuration

64.1. Configuring File Caching ... 64-2
64.1.1. Specifying the Location to Cache Terrain Server Data 64-3
64.1.2. Caching earth File Terrain Data ... 64-3
64.1.3. Caching earth File Terrain Data Offline ... 64-5
64.1.4. Enabling Texture Compression ... 64-6
64.1.5. Clearing the File Cache ... 64-6
64.2. Configuring Paged and Streaming Terrains ... 64-7
64.3. Debugging Earth Files ... 64-7
64.4. Using Shader-based Effect Maps ... 64-10
64.4.1. Composite Effects Maps ... 64-11
64.4.2. Debugging Shaders ... 64-12
64.4.3. Reloading Shaders ... 64-13
64.5. Indirect Rendering ... 64-14
64.5.1. Scaling the LOD Distance for Indirect Rendering ... 64-15
64.5.2. Setting the LOD Transition Percentage ... 64-15
64.6. Displaying Terrain Intersection Lines ... 64-16
64.7. Displaying DDS Textures Correctly ... 64-17
64.7.1. Flipping DDS Textures Globally ... 64-17

64.8. Enabling Occlusion Queries 64-19
64.9. Converting OpenFlight File Texture Data 64-20
64.10. Drawing Bounding Volumes for Scene Graph Nodes 64-22
64.11. Simulation Terrain Debugging 64-23

Chapter 65. Configuring Feature Data

65.1. Introduction 65-2
65.2. Specifying Different Feature Names for the Front-End and Back-End 65-2
65.3. Named Queries 65-3
65.4. Feature Configuration Attributes 65-4
65.5. Attribute Transformations 65-4
65.5.1. Transformations as Aliases 65-4
65.5.2. Transformations as Derived Attributes 65-5
65.5.3. Transformations as New Attributes 65-5
65.5.4. Writing Multiple Transforms to Account for Possible Failures 65-5
65.6. Source-Specific Configuration 65-6
65.6.1. Loading Source-Specific Configuration Files 65-6
65.6.2. Redefining Queries 65-6
65.7. Extracting Feature Data from GDB to Shapefiles 65-7
65.8. Converting S-57 Feature Data to Shapefiles 65-7
65.9. Configuring Aggregate-Level Movement Restrictions 65-9
65.10. Dynamic Features 65-10

Chapter 66. Terrain Tutorials

66.1. Creating an Earth File to Load Terrain Data 66-2
66.1.1. Add a Map Element 66-2
66.1.2. Add Elevation Data to the Earth File 66-2
66.1.3. Add Imagery to the Earth File 66-3
66.1.4. Add Feature Data to the Earth File 66-3
66.1.5. Add Options to the Earth File 66-5
66.1.6. Save the Earth File 66-6
66.1.7. Load the Earth File and Save is as an MTF File 66-6
66.2. Extracting Props from a Terrain 66-8
66.2.1. Create a New Terrain and Add a Terrain Patch 66-8
66.2.2. Extract the Props 66-9
66.2.3. Change the Model Definition for a Prop 66-11

Section XII Creating and Editing Simulation Models

Chapter 67. The Simulation Object Editor and SMSs

67.1. Introduction to the Simulation Object Editor 67-3
67.2. Starting the Simulation Object Editor 67-4
67.3. Simulation Model Sets 67-6
67.3.1. Including Simulation Model Sets in Other Simulation Model Sets 67-7
67.3.2. Enabling and Disabling Read Only Status for SMSs ... 67-9
67.3.3. SMS Priority for Included and Multiple SMSs 67-10

67.3.4. Lua Scripts in Included SMSs 67-11
67.3.5. Application Level Files in Included SMSs 67-11
67.3.6. Search Paths for Referenced Files in an SMS 67-12
67.3.7. Promoting a Model to the Loaded SMS 67-12
67.3.8. Promoting a System to the Loaded SMS 67-13
67.3.9. Hiding Included Objects 67-13
67.4. Visual Model Definitions 67-14
67.4.1. User-Created Visual Definition Files 67-14
67.5. Upgrading Old Simulation Model Sets 67-15
67.5.1. The Old SMS Includes a Standard SMS 67-16
67.5.2. The Old SMS was Copied from a Standard SMS 67-16
67.5.3. The Old SMS was Completely New or Copied from *base.sms* 67-18
67.5.4. Copying Visual Data 67-18
67.5.5. Running the SMS Upgrade Tool 67-19
67.5.6. Migrating an SMS to a New Release by Hand 67-20
67.5.7. Migrating Pre 4.7 Weapon and Damage Systems 67-21
67.6. Opening a Simulation Model Set 67-22
67.7. Creating New Simulation Model Sets 67-23
67.7.1. Creating a New SMS by Including an SMS 67-23
67.7.2. Creating a New SMS by Copying an SMS 67-24
67.8. Importing .entity Files 67-25
67.9. Applying New and Updated Variables 67-25
67.10. Importing and Exporting Object Information 67-26
67.10.1. Exporting Object Information 67-26
67.10.2. Importing Object Information 67-26
67.11. Editing the Tags List 67-27
67.11.1. Adding a New Tag 67-27
67.11.2. Removing a Tag 67-29
67.11.3. Renaming a Tag 67-29
67.11.4. Undoing Tag Changes 67-29
67.11.5. Using Auto Tags 67-30
67.11.6. Assigning Tags Based on Visual Model Characteristics 67-30
67.11.7. Exporting Tags 67-30
67.11.8. Importing Tags 67-31
67.11.9. Managing Tags Using the Object Information File ... 67-31
67.12. Managing the Forces List 67-31
67.12.1. Adding a Force 67-31
67.12.2. Editing a Force 67-33
67.12.3. Removing a Force 67-33
67.13. Copying the Object List to the Clipboard 67-33

Chapter 68. Editing Common Parameters

68.1. Editing Generic Object Parameters 68-3
68.1.1. Selecting an Object to Edit 68-3
68.1.2. Editing an Object's Tags 68-4
68.1.3. Changing an Object's Object Type Enumeration 68-5
68.1.4. Changing an Object's Name 68-7
68.1.5. Specifying Whether an Object Can Be Created 68-8

68.1.6. Specifying the Object Creation Palette 68-8
68.1.7. Specifying the Default Overlay 68-8
68.1.8. Specifying the Platform 68-9
68.1.9. Editing a Simulation Object's Country List 68-9
68.1.10. Creating Icon Lists 68-10
68.1.11. Undoing (Reverting) Changes to an Object 68-12
68.2. Editing a Simulation Object's Visual Model 68-13
68.2.1. Changing the 3D Model or Colorized Model 68-14
68.2.2. Changing the 2D Icon Using a Font 68-16
68.2.3. Changing the 2D Icon Using an Image 68-16
68.2.4. Editing a Spot Report Model 68-18
68.2.5. Adding a New Visual Model or Image 68-18
68.2.6. Displaying Details about the Visual Model 68-20
68.2.7. Changing a Visual Model's Type 68-22
68.2.8. Specifying Visual Markings 68-23
68.2.9. Importing Visual Models 68-24
68.2.10. Pruning Visual Models 68-25
68.2.11. Exporting Visual Models 68-25
68.2.12. Reloading Visual Models in the VR-Forces GUI 68-25
68.3. Creating a New Simulation Object 68-26
68.3.1. Creating a New Model from an Existing Model 68-27
68.4. Deleting an Object Model 68-27
68.5. Creating a Simulation Object Group 68-28
68.5.1. Setting the Simulation Connection when Creating an SOG 68-29
68.6. Editing Tactical Graphics 68-30
68.6.1. Editing a Tactical Graphic's Menu Icon 68-30
68.6.2. Setting the Default Values for Tactical Graphics Properties 68-31
68.6.3. Specifying the Properties that Can Be Edited in the Front-End 68-32
68.6.4. Editing the Visual Model for a Tactical Graphic 68-33
68.7. Creating a Template 68-36
68.8. Editing Systems in the Simulation Object Editor 68-36
68.8.1. Creating a New System 68-37
68.8.2. Editing Probability Tables 68-38
68.8.3. Deleting a System 68-41
68.8.4. Reverting a System 68-42

Chapter 69. Editing Entities for Entity-Level Scenarios

69.1. Editing Simulation Object Parameters 69-3
69.1.1. Editing a Simulation Object's Bounding Volume (Size) 69-3
69.1.2. Displaying a Simulation Object's Bounding Volume 69-4
69.1.3. Specifying Support Points 69-5
69.1.4. Excluding Entities from Batch Bounding Volume and Support Point Updates 69-6
69.1.5. Editing Behavioral Parameters 69-6
69.1.6. Adding Object Geometry to a Simulation Object 69-7
69.1.7. Configuring Cultural Features To Be Embedded in the Terrain 69-9

69.1.8. Ground Clamping Cultural Features 69-10
69.1.9. Editing a Lifeform's Visual Model and Animation 69-11
69.2. Editing Sensor Signatures 69-14
69.3. Editing a Simulation Object's Systems 69-15
69.3.1. Adding a System 69-16
69.3.2. Selecting a Damage Model 69-17
69.3.3. Selecting a Movement Model 69-17
69.3.4. Editing a System's Properties 69-18
69.3.5. Removing a System 69-18
69.4. Creating and Editing Munitions 69-19
69.4.1. Creating a Simulated Munition 69-19
69.4.2. Creating a New Ammunition Type 69-20
69.5. Configuring Wakes 69-20
69.6. Creating Randomized Simulation Objects 69-22
69.7. Configuring Scripted Entity Movement 69-24
69.7.1. Configuring Ballistic Missile Movement 69-25
69.7.2. Configuring Animated Movement 69-27
69.7.3. Editing Movement File Configurations 69-29

Chapter 70. Editing Units for Entity-Level Scenarios

70.1. Introduction to Editing Units 70-2
70.2. Creating a Unit for Entity-Level Scenarios 70-2
70.3. Specifying an Aggregate As Option 70-3
70.4. Editing Unit Composition 70-4
70.4.1. Adding Subordinates 70-4
70.4.2. Removing Subordinates 70-5
70.4.3. Replacing Subordinates 70-6
70.4.4. Changing the Order of Subordinates 70-6
70.4.5. Editing a Subordinate Unit 70-7
70.5. Editing Formations 70-8
70.5.1. Adding Formations 70-9
70.5.2. Removing Formations 70-10
70.5.3. Renaming Formations 70-10
70.5.4. Specifying the Default Formation 70-10
70.5.5. Expanding and Collapsing the Display of Formations 70-10
70.5.6. Displaying Bounding Boxes 70-11
70.5.7. Copying Formations 70-12
70.5.8. Laying Out a Formation Automatically 70-13
70.5.9. Changing a Formation's Layout by Hand 70-13

Chapter 71. Editing Aggregate-Level Objects

71.1. Creating Objects for Aggregate-Level Modeling 71-2
71.2. Configuring Simulation Objects for Aggregate-Level Modeling 71-2
71.2.1. Aggregating Units into Higher Echelon Units 71-3
71.3. Configuring Ammunition, Weapons, and Equipment 71-4
71.3.1. Defining Ammunition, Weapons, and Equipment Items 71-7

71.4. Assemblies 71-10
71.4.1. Creating Assemblies 71-11
71.4.2. Editing Assemblies 71-12
71.4.3. Adding Assemblies to a Unit 71-13
71.4.4. Removing Assemblies 71-13
71.5. Creating Roll Up Rules 71-14
71.6. Rolling Up Assemblies 71-15
71.7. Adding Activities 71-16
71.8. Object Parameters for Aggregate-Level Modeling 71-17
71.8.1. General Parameters 71-18
71.8.2. Physical Parameters 71-18
71.8.3. Movement Parameters 71-19
71.8.4. Sensor Parameters 71-21
71.8.5. Attack Parameters 71-22
71.8.6. Defense Parameters 71-23
71.8.7. Combat Parameters 71-24
71.8.8. Electronic Warfare (EW) Parameters 71-24
71.8.9. Personnel and Equipment Parameters 71-25
71.8.10. Supplies Parameters 71-25
71.9. Configuring Combat Engineering Objects 71-26
71.10. Editing Reader/Writer Key/Value Pairs 71-27

Chapter 72. Configuring Embarkation

72.1. Enabling Embarkation 72-2
72.2. Configuring Slots 72-5
72.2.1. Adding Slots 72-5
72.2.2. Configuring Slot Parameters 72-7
72.2.3. Configuring Slot Exclusions 72-9
72.2.4. Specifying Ingress and Egress Techniques 72-10
72.3. Adding Ingress and Egress Points 72-11
72.4. Embarkation Slot Tutorial 72-14
72.4.1. Configure the SuperTanker for Embarkation 72-14
72.4.2. Configure Slots for the Supertanker 72-16
72.4.3. Add Ingress Points 72-21
72.4.4. Add Egress Points 72-22
72.4.5. Save the New Configuration 72-23
72.4.6. Test the Embarkation Configuration 72-23

Chapter 73. Adding DI-Guy Models to VR-Forces

73.1. Adding New DI-Guy Models to VR-Forces 73-2
73.1.1. Copy DI-Guy Files to the Data Directory 73-2
73.2. DI-Guy Configuration Files 73-3
73.2.1. Adding a New DI-Guy Animation 73-3
73.3. How VR-Forces Maps DI-Guy Models 73-5
73.4. Add a New DI-Guy Entity in the SOE 73-5

Chapter 74. The Object Parameter Database

74.1. The Object Parameter Database 74-2
74.1.1. VR-Forces Classes and Object Parameters 74-3
74.1.2. Object Types 74-4
74.1.3. Parameter Types 74-8
74.1.4. Local and Remote Subcomponents 74-9
74.2. Component Descriptors and Parameters 74-13
74.2.1. Common Elements of Component Descriptors 74-13
74.2.2. Configuring the Priority of Components 74-16
74.3. Platform Files 74-16
74.4. System Definition Files 74-17
74.4.1. Creating a System Definition File 74-17
74.4.2. Configuring System Connections 74-18
74.5. Adding Variable Bindings to Platform and System Files 74-22
74.5.1. Adding a Variable Binding 74-23
74.5.2. Making Variables Editable in the Simulation Object Editor 74-24
74.5.3. Adding Platform Variable Bindings to the SOE 74-24
74.5.4. Adding System Definition File Variables to the SOE . 74-26
74.6. State Properties and Parameter Properties 74-26
74.6.1. Adding State Properties and Parameter Properties ... 74-27
74.6.2. Getting and Setting State Properties Using Lua Scripts 74-29
74.6.3. Displaying State Property Values in the Information Dialog Box 74-29
74.6.4. An Example of Adding State Properties 74-30
74.7. Configuring Units 74-39
74.8. Configuring Tactical Graphics 74-39
74.9. Pathname Interpretation in Simulation Model Set Files 74-40

Chapter 75. Configuring Sensors

75.1. Configuring Sensors 75-2
75.1.1. Adding a New Sensor to a Simulation Object 75-2
75.1.2. Editing a Sensor System Definition File 75-4
75.1.3. Specifying a Sensor Signature 75-5
75.1.4. Specifying Sensor Detection Probabilities 75-9
75.1.5. Adding Detection Types (Why Doesn't My Sensor Sense Anything?) 75-11
75.1.6. Adding a New Sensor Type 75-11
75.1.7. Illumination and Time of Day 75-14
75.1.8. Configuring the Effects of Rain on Radar 75-15
75.2. Detection Tables 75-16

Chapter 76. Configuring Weapon Systems

76.1. Introduction to Weapon Systems 76-2
76.1.1. The Weapon System System Definition Template File 76-3
76.1.2. The Weapon System System Definition File 76-5
76.2. Configuring Direct Fire Weapons 76-8
76.2.1. Sensing Targets 76-9

76.2.2. Configuring Ammunition 76-10
76.2.3. Hit Probability Tables 76-12
76.2.4. Damage Probability Tables 76-13
76.3. Configuring Missile Systems 76-14
76.3.1. Missile Launcher 76-15
76.3.2. Missile Projectile 76-16
76.3.3. Laser Guided Missile Launcher 76-16
76.3.4. Laser Guided Missile Projectile 76-17
76.4. Configuring Indirect Fire Weapons 76-17
76.5. Configuring Fixed Weapons 76-19
76.6. Configuring Bomb Bay Weapon Systems 76-20
76.7. Configuring Weapon Systems in the SOE 76-20
76.7.1. Specifying Target Types 76-20
76.7.2. Configuring Hit Probability 76-21
76.7.3. Configuring the Ammunition to Use 76-21
76.7.4. Using the Target Configuration Assistant 76-22
76.7.5. Configuring the Target Priorities 76-24
76.7.6. Configuring the Target Region 76-25
76.7.7. Configuring Missile Launcher Systems in the SOE ... 76-25
76.7.8. Configuring Explosive Devices 76-26

Chapter 77. Configuring Damage Systems

77.1. Introduction to Damage Systems 77-2
77.1.1. Calculating Damage from Munitions 77-2
77.2. Configuring Damage-by-Power 77-3
77.2.1. Configuring the Power of a Detonation 77-4
77.2.2. Configuring Detonation Power in the SOE 77-6
77.2.3. The Damage File for Damage-By-Power 77-8
77.2.4. Adding a New Power Type 77-10
77.3. Configuring Damage in the SOE 77-11
77.3.1. Configuring Damage Power Entries 77-11
77.3.2. Configuring Impact Surfaces 77-12
77.3.3. Configuring the Angle of Incidence 77-13
77.3.4. Specifying Kill Data 77-13
77.4. Viewing Inter-Object Damage Probability Results 77-14
77.5. Configuring Damage for Dynamic Terrain 77-16
77.5.1. Standard Terrain Damage 77-16
77.5.2. High Fidelity Terrain Damage 77-17
77.6. Configuring Detonation Results 77-18

Chapter 78. Configuring Movement Systems

78.1. Configuring Formations 78-2
78.1.1. Formation Files 78-2
78.1.2. Creating a User-Defined Formation 78-4
78.2. Configuring Obstruction Avoidance 78-4
78.2.1. Configuring Object Types 78-5
78.2.2. Configuring Vector Feature Types 78-5
78.3. Configuring Embarkation 78-6

Chapter 79. Configuring Emitters and Radios

79.1. Configuring Emitters 79-2
79.2. Configuring Radios 79-4
79.2.1. Publishing Radio Transmitters 79-4

Chapter 80. Joystick and Keyboard Controls

80.1. Using a Joystick to Control Entities 80-2
80.1.1. Switching Between Systems 80-3
80.1.2. Disabling Joystick Control 80-4
80.2. Using the Keyboard to Control an Entity 80-4
80.3. Configuring Joysticks and Keyboard Control 80-4
80.3.1. Creating a Joystick Configuration 80-5
80.3.2. Mapping the Joystick Stick 80-7
80.3.3. Mapping Joystick Buttons and Keyboard Keys 80-8
80.3.4. Configuring "Switch Between" Options 80-11
80.3.5. Using Multiple Joysticks 80-14
80.3.6. Specifying a Favorite Joystick Configuration 80-15
80.4. Installing a Gamepad on Linux 80-15

Chapter 81. Attaching Components to Remote Entities

81.1. Attaching Components to Remote Entities 81-2

Section XIII Configuring the Graphical User Interface

Chapter 82. Display Engine and Window Management

82.1. The Display Engine 82-2
82.1.1. Window Types 82-2
82.2. Managing Display Engine Configurations 82-2
82.2.1. Adding a Window 82-2
82.2.2. Adding a Channel to a Window 82-4
82.2.3. Removing a Window 82-5
82.2.4. Removing a Channel 82-5
82.2.5. Saving a Display Engine Configuration 82-5
82.2.6. Loading a Display Engine Configuration 82-5
82.3. Changing a Window's Attributes 82-6
82.4. Changing a Channel's Attributes 82-7
82.4.1. Setting the Clipping Planes 82-9
82.4.2. Specifying the Projection Resize Policy Attribute 82-11
82.4.3. Changing a Channel's Frustum (Field of View) 82-14
82.4.4. Changing the Viewport 82-15
82.4.5. Configuring Water Visibility 82-16
82.4.6. Channel Keywords 82-17
82.5. Configuring Multichannel Displays 82-18
82.5.1. Changing the Camera's Position and Orientation Offset 82-20
82.6. Stereoscopic Displays 82-21
82.6.1. Configuring Anaglyphic Stereo 82-22
82.6.2. Configuring Polarized Stereo 82-23

Chapter 83. GUI Controls, Layouts, and Behaviors

83.1. Dockable Panels and Toolbars 83-2
83.1.1. Docking and Undocking a Panel 83-2
83.1.2. Docking and Undocking a Toolbar 83-3
83.1.3. Displaying and Hiding Panels and Toolbars 83-3
83.2. Saving Window Layouts 83-3
83.2.1. Creating a Window Layout 83-4
83.2.2. Choosing the Window Layout 83-5
83.2.3. Setting a Window Layout as the Default Layout 83-5
83.2.4. Reverting a Window Layout 83-6
83.2.5. Updating a Window Layout 83-6
83.2.6. Renaming a Window Layout 83-6
83.2.7. Deleting a Window Layout 83-7
83.2.8. Restoring a Factory Layout 83-7
83.3. Using Tear-Off Menus 83-8
83.4. Viewing the Window in Full-Screen Mode 83-8
83.4.1. Hiding the Mouse in Full Screen Mode 83-9
83.5. Using Context-Sensitive Menus 83-10
83.6. Common VR-Forces Buttons 83-10
83.7. Adding Text to the Title Bar 83-11
83.8. Specifying Display Units 83-12

Chapter 84. Configuring Toolbars, Dialog Boxes, and Menus

84.1. Introduction 84-2
84.2. Configuring Menus and Menu Commands 84-2
84.2.1. Alphabetizing the Task and Set Menus 84-3
84.2.2. Creating New Menus 84-4
84.3. Configuring Dialog Box Pages 84-5
84.4. Configuring Context-Sensitive Menus 84-7
84.4.1. Editing the Task Menu 84-8
84.4.2. Editing the Set Menu 84-8
84.5. Configuring Toolbars 84-9
84.5.1. Creating Toolbars 84-9
84.5.2. Specifying an Icon for a Toolbar Item 84-10
84.6. Locking Main GUI Elements 84-10

Chapter 85. Creating and Editing Key Mappings

85.1. Introduction 85-2
85.2. The Key Mapping Editor 85-2
85.2.1. Binary Key Mappings 85-3
85.3. Editing a Key Map 85-3
85.3.1. Adding a Key Mapping 85-3
85.3.2. Changing a Key Mapping 85-5
85.3.3. Deleting a Key Mapping 85-5
85.4. Using Combined Key Mappings 85-6
85.5. Filtering the Function List 85-7
85.6. Creating a Key Map 85-7
85.7. Deleting a Key Map 85-7

Section XIV Configuring Visual Models in the Visual Models Editor

XIV.1. Managing Model and Element Definitions XIV-1
XIV.2. Introduction to Object Modeling XIV-2

Chapter 86. Schemas

86.1. Creating and Editing Schemas 86-2
86.2. Creating Schemas 86-3
86.3. Deleting Schemas 86-4
86.4. Copying Schemas 86-4
86.5. Adding a Parameter to a Schema 86-5
86.6. Deleting a Parameter from a Schema 86-6
86.7. Editing a Schema Parameter 86-6

Chapter 87. Model Definitions

87.1. Creating and Editing Model Definitions 87-2
87.2. Work Flow Issues for Model Definitions 87-3
87.3. Creating a Model Definition 87-5
87.4. Deleting a Model Definition 87-6
87.5. Copying a Model Definition 87-7
87.6. Filtering the List of Model Definitions 87-8
87.7. Editing a Model Definition 87-8
87.7.1. Adding Parameters to a Model Definition 87-8
87.7.2. Editing a Model Definition Parameter 87-10
87.7.3. Deleting Parameters from a Model Definition 87-10
87.8. Jumping to an Element Definition 87-11
87.9. Saving Model Definitions 87-11
87.10. Importing Model Definitions 87-12
87.10.1. Importing Model Definitions from the Command Line 87-12
87.11. Exporting Model Definitions 87-13
87.11.1. Exporting Selected Model Definitions 87-13

Chapter 88. Element Definitions

88.1. Creating and Editing Element Definitions 88-2
88.2. How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions are Organized 88-3
88.2.1. Leaf File Descriptions 88-4
88.3. Work Flow Issues for Creating and Editing Element Definitions 88-5
88.4. Creating an Element Definition 88-7
88.5. Editing an Element Definition 88-8
88.5.1. Adding a Visual Definition 88-8
88.5.2. Editing a Visual Definition 88-9
88.5.3. Adding an Attribute to a Visual Definition 88-12
88.5.4. Deleting an Attribute from a Visual Definition 88-13
88.6. Jumping to a Model Definition 88-14
88.7. Deleting an Element Definition 88-14
88.8. Saving Element Definitions 88-15

88.9. Importing Element Definitions 88-15
88.9.1. Importing a Hierarchy File 88-16
88.9.2. Importing and Merging Element Definitions 88-16
88.9.3. Importing Element Definitions from the Command Line 88-17
88.10. Exporting Element Definitions 88-17
88.10.1. Exporting Interaction and Smoke Element Definitions to *.elem* Files 88-18
88.10.2. Exporting Entity, Unit, Spot Report, or Tactical Graphics Element Definitions 88-19

Chapter 89. Mapping Models and Effects

89.1. Introduction to Object Type Mapping 89-2
89.2. Work Flow Issues for Object Type Mappings 89-2
89.3. Adding an Object Type Mapping 89-4
89.4. Editing an Object Type Mapping 89-5
89.5. Filtering the Element Definition List 89-6
89.6. Jumping to an Element Definition 89-7
89.7. Deleting an Object Type Mapping 89-7
89.8. Saving Object Type Mappings 89-8
89.9. Importing Object Type Mappings 89-8
89.10. Exporting Object Type Mappings 89-8
89.11. Starting without Default Object Type Mappings 89-9

Chapter 90. Wakes, Cockpits, 2D Icons, and other Models

90.1. Configuring 2D Icons 90-3
90.1.1. Editing Font-Based 2D Icons 90-3
90.1.2. Using Images for 2D Icons 90-4
90.1.3. Adding Images for Simulation Object Icons 90-9
90.2. Adding New Cockpit Display Models 90-10
90.2.1. Installing a Cockpit DLL 90-11
90.2.2. Creating a Model Definition for a Cockpit 90-11
90.3. Adding Object Models as Props 90-13
90.3.1. Adding a Category to the Props Palette 90-14
90.4. Configuring Light Points 90-14
90.4.1. Configuring Light Point Type Models 90-16
90.4.2. Configuring Light Point Group Models 90-16
90.4.3. Adding a New Light Point Type 90-17
90.5. Configuring Wakes 90-18
90.5.1. Configuring Tidal Stream Wakes 90-20
90.6. Adding Wind-based Controls to Models 90-21
90.7. Using Texture Atlases to Skin Models 90-22
90.7.1. Configuring Texture Mapping in the Model Definition 90-22
90.7.2. Metafiles for Texture Mapping 90-23
90.8. Adding Dynamically Changeable Markings to Models 90-24
90.8.1. Adding @dis annotations to the Model 90-26
90.8.2. The Metafile for Markings 90-27
90.8.3. The Texture Mapping File 90-28

90.9. Flipping DDS Textures for a Model 90-29
90.10. Configuring SpeedTree Trees 90-30
90.10.1. Randomizing the Size of SpeedTrees 90-31
90.10.2. Displaying SpeedTrees in Planar Reflections 90-33
90.11. Best Practices for Creating Models 90-34
90.12. Disabling Indirect Rendering for an OpenFlight Model 90-35
90.13. Compressing Model Files 90-35

Chapter 91. Configuring Emitter Volumes

91.1. Configuring Emitter Volumes 91-2
91.1.1. Configuring Emitter Volume Color 91-2
91.1.2. Controlling Emitter Volume Radius 91-3
91.1.3. Configuring Emitter Volume Segments 91-4

Chapter 92. The WRM Specification (DIS Notes)

92.1. Introduction 92-2
92.2. The OpenFlight File Format 92-3
92.2.1. Node Name and Comment Fields 92-3
92.2.2. External References and Instancing 92-3
92.3. Modeling for DIS Interoperability 92-4
92.3.1. The DIS Attribute Lexicon (DAL) 92-4
92.3.2. DAL Keywords 92-5
92.3.3. Model Coordinate Systems 92-6
92.4. Articulated Parts 92-8
92.5. Entity Appearances 92-9
92.6. Animation 92-12
92.7. DIS Keyword Values 92-13

Section XV Appendixes

Appendix A. MTL Syntax

A.1. Introduction A-2
A.2. MAK Technologies Lisp (MTL) Files A-3
A.2.1. Using Environment Variables in an MTL File A-3
A.2.2. Specifying Lists of Lists A-4
A.2.3. Using Conditional Statements A-4

Appendix B. The vrfSim.mtl Configuration File

B.1. The *vrfSim.mtl* Configuration File B-2

Appendix C. Object Parameters

C.1. The Structure of a Simulation Object Parameter C-2
C.2. The Parameter Type Hierarchy C-3
C.2.1. Vrf Base Object Parameters C-3
C.2.2. Vrf Object Parameters C-7
C.2.3. Moving Object Parameters C-8
C.2.4. Parameter Types for Entity Platforms C-8

C.3. Object Type Enumerations C-9
C.3.1. Object Super-type C-9
C.3.2. AggregateKind C-9
C.3.3. ObjectKind C-10
C.3.4. Unit Category C-10
C.3.5. Unit Specific C-11
C.3.6. Point Object Category C-12
C.3.7. Linear Object Category C-12
C.3.8. Areal Object Category C-12
C.3.9. Interaction C-12

Appendix D. Deprecated Weapon System and Munition Damage Configuration

D.1. Configuring Direct Fire Weapons D-2
D.1.1. Sensing Targets D-4
D.1.2. Ammo Select Tables D-6
D.1.3. Hit Probability Tables D-7
D.1.4. Damage Probability Tables D-8
D.2. Configuring Indirect Fire Weapons D-9
D.3. Configuring Fixed Weapons D-11
D.4. Configuring Bomb Bay Weapon Systems D-12
D.5. Calculating Damage from Munitions D-13
D.6. Configuring Damage-by-Power D-14
D.6.1. Configuring the Power of a Detonation D-15
D.6.2. The Damage File for Damage-By-Power D-18
D.6.3. Adding a New Power Type D-20
D.7. Configuring Damage-by-Ammo-Type D-22
D.7.1. Configuring Damage for Damage-by-Ammo-Type D-23
D.8. Direct Fire Configuration Examples D-24
D.8.1. Direct Fire Weapon Damage-By-Power Example D-25
D.8.2. Direct Fire Weapon Damage-by-Ammo-Type Example D-30

Appendix E. Systems and System Usage

E.1. VR-Forces Systems E-2

MAK Products Glossary

Index

VR-Forces Quick Reference Card



Preface

This manual is written for persons who will install VR-Forces® or use the VR-Forces application. The manual assumes that you are familiar with your operating system and that you know how to work in your computer's graphical window environment.

Please see release documentation for information about changes and updates to VR-Forces since this manual went to press.

How This Manual is Organized

The chapters are organized as follows:

- Section I, "Introduction, Installation, and Startup"
  - Chapter 1, *Introduction to VR-Forces*, briefly describes VR-Forces and its features.
  - Chapter 2, *Installing VR-Forces*, explains how to install VR-Forces, set up license management, and localize the GUI.
  - Chapter 3, *VR-Forces Application Concepts*, provides the conceptual background for the simulation process. It will help put some of the discussion in this manual into context.
  - Chapter 4, *Starting VR-Forces* describes how to start VR-Forces, load terrain databases, and configure HLA time management.
  - Chapter 5, *Command-Line Options*, describes the command-line options for vrfGui (the front-end) and vrfSim (the back-end or simulation engine).
  - Chapter 6, *Optimizing Performance* describes how to monitor VR-Forces's performance in the Frame Rate Monitor and how to configure VR-Forces to improve performance.

- Section II, "Scenarios"
  - Chapter 7, *Creating and Running Scenarios*, explains how to create and run scenarios, how to create batch scenarios, and how to checkpoint scenarios.
  - Chapter 8, *Scenario Events* explains how to create and run scenario events.
  - Chapter 9, *Target Detection and Combat Features*, describes spot reports, laser targeting, entity hostility, and how entities take damage.
  - Chapter 10, *Indirect Fire, Ballistic Missiles, and Munition Detonations*, explains how to configure and call in indirect fire.
  - Chapter 11, *Setting Environment Conditions*, describes how to manipulate the date, time of day, and weather conditions.
  - Chapter 12, *Scenario Files* describes the files used to define a scenario and how to edit them.
- Section III, "Simulation Objects"
  - Chapter 13, *Introduction to Simulation Objects* contains conceptual details about simulation objects.
  - Chapter 14, *Creating and Placing Objects*, explains describes the common procedures for creating simulation objects, tactical graphics, and props.
  - Chapter 15, *Selecting Objects* describes procedures common to simulation objects, control objects, and tactical graphics.
  - Chapter 16, *Viewing Information about Objects* explains how simulation objects are displayed, how you can change the display of simulation objects, and how you can display information about simulation objects.
  - Chapter 17, *Embarked and Embedded Entities* explains how to embark entities. It also explains how to deploy and recover embedded entities..
  - Chapter 18, *Generating Traffic (Pattern of Life)* explains how to use the Spawn Pattern Manager to create pattern of life traffic.
  - Chapter 19, *Crowds and Multiple Simulation Objects* explains how to create multiple simulation objects at one time and how to create and task pedestrian crowds.
  - Chapter 20, *Creating and Using an Order of Battle* explains how to create an Order of Battle, how to import one into a scenario, and how to create the simulation objects in the Order of Battle.
- Section IV, "Modeling Units"
  - Chapter 21, *Entity-Level Unit Modeling*, describes working with units in entity-level scenarios.
  - Chapter 22, *How Aggregate-Level Modeling Works*, describes the aggregate-level warfare model.
  - Chapter 23, *Creating and Controlling Units*, explains how to create and edit units.
  - Chapter 24, *Displaying Units* describes how to view units.

- Section V, "Tasks"
  - Chapter 25, *Assigning Tasks*, provides general information about assigning tasks and creating behavior sets.
  - Chapter 26, *Movement Tasks for Entity-Level Scenarios*, describes movement tasks for simulation objects in entity-level scenarios.
  - Chapter 27, *Engagement Tasks for Entity-Level Scenarios*, describes combat-related tasks for simulation objects in entity-level scenarios.
  - Chapter 28, *Human and Crowd Tasks*, describes tasks for humans and crowds in entity-level scenarios.
  - Chapter 29, *Embarkation, Wait, Radio, and Other Tasks*, describes the tasks provided with VR-Forces that apply to simulation objects in entity-level scenarios.
  - Chapter 30, *Tasks for Aggregate-Level Scenarios*, describes the tasks that apply to simulation objects in aggregate-level scenarios.
  - Chapter 31, *Creating Scripted Tasks and Sets*, introduces scripts and explains how to create them. It does not cover how to write scripts.
  - Chapter 32, *Writing Scripts*, introduces the Lua script API for VR-Forces and explains how to write scripts.
  - Chapter 33, *Using Qt Designer for Script Dialog Boxes*, explains how to use Qt Designer to design dialog boxes for scripted tasks and sets instead of using the automatic dialog box generation feature.
- Section VI, "Set Data Requests"
  - Chapter 34, *Setting the State of Entity-Level Simulation Objects*, describes how to change the state of simulation objects in entity-level scenarios using set data requests.
  - Chapter 35, *Setting the State of Aggregate-Level Simulation Objects*, describes how to change the state of simulation objects in aggregate-level scenarios using set data requests.
- Section VII, "Plans"
  - Chapter 36, *Plan Concepts* describes plans and plan components.
  - Chapter 37, *Writing Plans* explains how to write plans for simulation objects and how to write global plans.
- Section VIII, "Tactical Graphics"
  - Chapter 38, *Introduction to Tactical Graphics* is a brief introduction to tactical graphics.
  - Chapter 39, *Overlays* describes the creation and use of overlays.
  - Chapter 40, *Creating and Editing Tactical Graphics* describes how to create tactical graphics other than points, lines, and areas, and how to edit tactical graphics.
  - Chapter 41, *Setting the State of Tactical Graphics* describes set data requests that apply to tactical graphics.
  - Chapter 42, *Remote Graphics* explains how to enable display of graphics generated by remote applications.

- Section IX, "Visual and Audio Effects"
  - Chapter 43, *Displaying Tactical Information and Effects* describes how to display a variety of simulation object effects such as trailing effects, shadows, and sensor volumes.
  - Chapter 44, *Visualizing 2D Icons and 3D Models* explains how to change the size of models and icons and hide them.
  - Chapter 45, *Lighting Effects* describes the lighting effects supported by VR-Forces and how to manage them.
  - Chapter 46, *Marine Effects* describes water related effects, such as displaying rain splash and water spray. It also covers wake effects and entity buoyancy.
  - Chapter 47, *Using CameraFX Sensor Views* explains how to enable and configure infrared and night vision goggle sensor views.
  - Chapter 48, *Intervisibility* explains how to determine if simulation objects can see other simulation objects or areas of the terrain.
  - Chapter 49, *Sound Effects* explains how to enable and configure sound effects for entities.
- Section X, "The Observer and Viewing Scenarios"
  - Chapter 50, *The Observer and Observer Modes* explains what an observer is and the different ways observers can be configured.
  - Chapter 51, *Moving the Observer* explains how to navigate the 3D and 2D views by moving the observer.
  - Chapter 52, *Attaching the Observer to Objects* explains how to attach the observer to simulation objects. This is an alternative to manually moving the observer through the terrain.
  - Chapter 53, *Inset Views, Sensor FOVs, HUDs, and Depth of Field* describes special secondary windows – inset views, sensor fields of view, and instrument panels, and HUDs.
- Section XI, "Terrain"
  - Chapter 54, *Introduction to Terrain Usage in VR-Forces* provides conceptual background for terrain use in VR-Forces.
  - Chapter 55, *Displaying Terrain Information* describes ways to change how terrain is displayed. These options provide different types of information about the terrain.
  - Chapter 56, *Using Terrain Profiles* explains how to display and configure terrain profiles.
  - Chapter 57, *Composing Terrains* explains how to build terrain databases by importing elevation data, imagery, and feature data.
  - Chapter 58, *Paged and Streaming Terrains* explains how to use terrain servers for paged and streaming terrains. It also provides information about preparing paged terrains for use in VR-Forces.
  - Chapter 59, *Loading Data Using Earth Files* explains the syntax of earth files so that you can customize them for viewing streaming terrain from terrain servers.

- Chapter 60, *Configuring CDB Terrains* explains the syntax used in earth files to load CDB terrains.
- Chapter 61, *Generating Navigation Data* explains how to generate navigation data for advanced navigation by humans and ground entities.
- Chapter 62, *Dynamic Terrain* explains how to use the dynamic terrain capability in VR-Forces.
- Chapter 63, *Processing MetaFlight Files* explains how to preprocess MetaFlight files so that VR-Forces can load them.
- Chapter 64, *Terrain Performance and Configuration* covers miscellaneous issues that affect terrain performance and function.
- Chapter 65, *Configuring Feature Data* explains how to create configuration files for feature data whose attributes use different values than the ones that are preconfigured by VR-Forces.
- Chapter 66, *Terrain Tutorials* provides step-by-step instructions for creating and saving a terrain database.

◆ Section XII, "Creating and Editing Simulation Models"

- Chapter 67, *The Simulation Object Editor and SMSs* explains simulation model sets (SMSs) and how to create them in the Simulation Object Editor.
- Chapter 68, *Editing Common Parameters* explains how to use the Simulation Object Editor to edit simulation object parameters that are common to entity-level and aggregate-level SMSs.
- Chapter 69, *Editing Entities for Entity-Level Scenarios* explains how to use the Simulation Object Editor to edit simulation object parameters, systems, and models that apply to individual entities.
- Chapter 70, *Editing Units for Entity-Level Scenarios* explains how to configure unit composition and formations in the Simulation Object Editor. This chapter applies to units in entity-level scenarios.
- Chapter 71, *Editing Aggregate-Level Objects* explains how to configure aggregates for aggregate-level scenarios in the Simulation Object Editor.
- Chapter 72, *Configuring Embarkation* explains how to enable embarkation and configure slots, ingress points, and egress points.
- Chapter 73, *Adding DI-Guy Models to VR-Forces* explains how to configure new DI-Guy characters.
- Chapter 74, *The Object Parameter Database* describes the object parameter database, which contains parameters for simulation objects and control objects, and how to edit it.
- Chapter 75, *Configuring Sensors* explains how to configure sensors and sensor signatures.
- Chapter 76, *Configuring Weapon Systems* explains how to configure weapon systems by editing configuration files and in the Simulation Object Editor.
- Chapter 77, *Configuring Damage Systems* explains how to configure damage systems by editing configuration files and in the Simulation Object Editor.

  - Chapter 78, *Configuring Movement Systems* explains how to configure obstacle avoidance, embarkation, and formations.
  - Chapter 79, *Configuring Emitters and Radios* explains how to configure emitters and radios.
  - Chapter 80, *Joystick and Keyboard Controls* explains how to configure joysticks and keyboard keys to control entity movement.
  - Chapter 81, *Attaching Components to Remote Entities* explains how to configure sensors and sensor signatures.
- Section XIII, "Configuring the Graphical User Interface" GUI use and customization.
  - Chapter 82, *Display Engine and Window Management* explains how to configure windows and channels and how to save and load display configurations.
  - Chapter 83, *GUI Controls, Layouts, and Behaviors* explains how to manage toolbars and other aspects of the VR-Forces GUI.
  - Chapter 84, *Configuring Toolbars, Dialog Boxes, and Menus* explains how to configure the menus, toolbars, and dialog box pages that get displayed.
  - Chapter 85, *Creating and Editing Key Mappings* explains how to edit key mappings and create new key mappings.
- Section XIV, "Configuring Visual Models in the Visual Models Editor"
  - Chapter 86, *Schemas* explains how to create and edit model schemas.
  - Chapter 87, *Model Definitions* explains how to create and edit model definitions.
  - Chapter 88, *Element Definitions* explains how to create and edit element definitions.
  - Chapter 89, *Mapping Models and Effects* explains how to map entity enumerations to the models configured in VR-Forces.
  - Chapter 90, *Wakes, Cockpits, 2D Icons, and other Models* explains how to configure some specific visual models, including 2D Icons, cockpits, and SpeedTree trees.
  - Chapter 91, *Configuring Emitter Volumes* explains how to configure color, radius, and segment size for emitter volumes.
  - Chapter 92, *The WRM Specification (DIS Notes)* explains the WRM specification for 3D modeling.
- Section XV, "Appendixes".
  - Appendix A, *MTL Syntax* explains the syntax of configuration files provided with VR-Forces.
  - Appendix B, *The vrfSim.mtl Configuration File* describes the parameters in *vrfSim.mtl*.
  - Appendix C, *Object Parameters* describes the high level parameters used in OPD files.
  - Appendix D, *Deprecated Weapon System and Munition Damage Configuration* describes the how legacy weapon systems and damage systems are configured. These systems are supported by VR-Forces but current systems are configured differently..

– Appendix E, *Systems and System Usage* lists all the systems provided with VR-Forces and provides information about them, including which entities use them.

MAK Products Glossary defines terms common to DIS, HLA, and MAK products.

VR-Forces Quick Reference Card lists command line options, keyboard shortcuts, and navigation and view options.

VR-Forces Documentation

VR-Forces documentation is provided as manuals in PDF format, online help, and HTML class documentation. The PDF files are in the *./doc* directory. The VR-Forces documentation set is as follows:

- *VR-Forces Users Guide* contains all documentation for running VR-Forces, creating and running scenarios, and configuring VR-Forces.
- *VR-Forces Migration Guide* collates API migration information for recent releases.
- Online help. The VR-Forces front-end and the Simulation Object Editor have online help accessible from the Help menu.
- *VR-Forces Developers Guide* and API documentation. Class documentation and developers guides in linked HTML pages.
- *VR-Forces Release Notes*.
- *VR-Forces First Experience Guide*. A brief introduction to the most basic features of VR-Forces.
- *VR-Forces Entity Model Catalog*. A catalog of all of the simulation objects and tactical graphics configured in the Simulation Object Editor, with basic parameter details and a screen capture of the 3D model or icon.

MAK Products

VR-Forces is a member of the VT MAK line of software products designed to streamline the process of developing and using networked simulated environments. The VT MAK product line includes the following:

- VR-Link® Network Toolkit. VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA) and the Distributed Interactive Simulation (DIS) protocol. VR-Link has built-in support for the RPR FOM and allows you to map to other FOMs. This library minimizes the time and effort required to build and maintain new HLA or DIS-compliant applications, and to integrate such compliance into existing applications.

  VR-Link includes a set of sample debugging applications and their source code. The source code serves as an example of how to use the VR-Link Toolkit to write applications. The executables provide valuable debugging services such as generating a predictable stream of HLA or DIS messages, and displaying the contents of messages transmitted on the network.

- MAK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MAK RTI is optimized for high performance. It has an API, RTIspy®, that allows you to extend the RTI using plug-in modules. It also has a graphical user interface (the RTI Assistant) that helps users with configuration tasks and managing federates and federations.
- VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a GUI, that gives you a 2D and 3D views of a simulated environment.

  You can create and view entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. You can simulate using entity-level modeling, which focuses on the actions of individual people and vehicles, and aggregate-level modeling, which focuses on the interaction of large hierarchical units.

  VR-Forces also functions as a plan view display for viewing remote simulation objects taking part in an exercise. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.

- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes three end-user applications (VR-Vantage Stealth, VR-Vantage PVD, and VR-Vantage IG) and the VR-Vantage Toolkit.
  - VR-Vantage Stealth displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to simulation objects so that it moves as they do.

  - VR-Vantage IG is a configurable desktop image generator (IG) for out the window (OTW) scenes and remote camera views. It has most of the features of the Stealth, but is optimized for its IG function.
  - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.
  - SensorFX. SensorFX is an enhanced version of VR-Vantage that uses physics based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MAK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- DI-Guy™. DI-Guy is an SDK that allows you to embed the DI-Guy library in your real-time application and populate your world with lifelike human characters. DI-Guy provides high-fidelity human characters that move realistically, respond to simple high-level commands, and travel throughout the environment as directed by the hosting visual application. It comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy character simulation is an integral part of all MAK's visual applications, including VR-Engage, VR-Forces, and VR-Vantage.

- RadarFX. RadarFX is a client-server application that simulates synthetic-aperture radar (SAR). The server application, which is based on VR-Vantage and SensorFX, loads a terrain database and, optionally, connects to simulations. A client application requests SAR images from the server. RadarFX includes a sample client application.
- VR-Engage. VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, a sensor operator, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- WebLVC Server. WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several sample applications that work with VR-Forces and VR-Vantage.

How to Contact Us

For VR-Forces technical support, information about upgrades, and information about other MAK products, you can contact us in the following ways:

Telephone

Call or fax us at:	Voice:	617-876-8085 (extension 3 for support)
	Fax:	617-876-9208

E-mail

Sales and upgrade information:	info@mak.com
Technical support:	support@mak.com

Internet

MAK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses
Product version and platform information:	www.mak.com/support/product-versions
For the free, unlicensed MAK RTI:	www.mak.com/support/bonus-material

Post

Send postal correspondence to:	VT MAK 150 Cambridge Park Drive, 3rd Floor Cambridge, MA, USA 02140

When requesting support, please tell us the product you are using, the version, and the platform on which you are running.

Document Conventions

This manual uses the following typographic conventions:

`Monospaced`	Indicates commands or values you enter.
`Monospaced Bold`	Indicates a key on the keyboard.
`Monospaced Italic`	Indicates command variables that you replace with appropriate values.
Blue text	A hypertext link to another location in this manual or another manual in the documentation set.
`{ }`	Indicates required arguments.
`[ ]`	Indicates optional arguments.
`|`	Separates options in a command where only one option may be chosen at a time.
(|)	In command syntax, indicates equivalent alternatives for a command-line option, for example, `(-h | --help)`.
/	Indicates a directory. Since MAK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
Italic	Indicates a file name, pathname, or a class name.
sans Serif	Indicates a parameter or argument.
➤	Indicates a one-step procedure.
Menu → Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose **File → Save**.
	Click the icon to run a tutorial video in the default browser.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.
	Indicates that a section is valid only for entity-level scenarios.
	Indicates that a section is valid only for aggregate-level scenarios.

Directory names are preceded with dot and slash characters that show their position with respect to the VR-Forces home directory. For example, the directory *vrforces4.7/doc* appears in the text as *./doc*.

Mouse Button Naming Conventions

An instruction to click the mouse button, refers to clicking the primary mouse button, usually the left button for right-handed mice and the right button for left-handed mice. The context-sensitive menu, also called a popup menu or right-click menu, refers to the menu displayed when you click the secondary mouse button, usually the right button on right-handed mice and the left button on left-handed mice.

Third Party Licenses

MAK software products may use code from third parties. This section contains summary license information and where required, specific license documentation required by these third parties.

The following libraries are covered by the GNU Lesser General Public License (LGPL) license:

- CIGI
- DevIL
- GEOS
- GStreamer
- LibIconV
- OPCODE
- OpenAL
- OSG
- OsgEarth
- Pthread
- Qt
- Qwt
- LibWebSockets.

The following libraries are covered by the ZLib license:

- Bullet Physics
- Zlib
- LibPNG
- LibSDL
- Minizip.

The following libraries are covered by the MIT license:

- Expat
- FreeGLUT
- GDAL
- GeoTiff
- HarfBuzz
- LibCURL
- LibSquish
- LibXML
- LUA
- LuaBind
- Proj.

The following libraries are covered by the BSD license:

- FreeType
- GLEW
- JPEG
- LibTiff
- Protobuf
- PCRE.

The following libraries are covered by the commercial licenses:

- SpeedTree
- Sundog Triton
- Sundog SilverLining
- FlexLM
- Gameware
- JRM Technologies SenSimRT and SigSimRT
- GLStudio
- OpenFlight
- CDB API.

The following libraries are covered by custom licenses:

- NVidia CG Toolkit
- FreeImage (FreeImage Public License)
- Boost (Boost Software License)
- Poco (Boost Software License).

COLLADA DOM is covered by the SCEA Shared Source license.

Thread Building Blocks is covered by the GPL license:

SQLLite is in the public domain.

Boost License

VR-Link, and all MAK software that uses VR-Link uses some code that is distributed under the Boost License. All header files that contain Boost code are properly attributed. The Boost web site is: www.boost.org.

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libXML and libICONV

VR-Link and all MAK software that uses VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MAK Products. MAK has made no modifications to these libraries. For more information about these libraries please see the following web sites:

- The LGPL license is available at: http://www.gnu.org/licenses/lgpl.html.
- Information about IconV is at: http://www.gnu.org/software/libiconv/.
- Information about LibXML is at: http://xmlsoft.org/.

Lua

Some MAK products use the Lua programming language (www.lua.org). Its license is as follows:

Copyright © 1994–2012 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Freefont OpenType Font Set

VR-Vantage applications and VR-Forces use the Freefont OpenType font set from the Free Software Foundation. It is covered by the General Public License (GPL). For details, please see: http://www.gnu.org/licenses/gpl.html

NVIDIA

The HDR implementation uses the HDR NVIDIA gameworks SDK example that has the following copyright notice:

Multiple Files in folder: es3aep-kepler\HDR/

SDK Version: v2.11

Email: gameworks@nvidia.com

Site: http://developer.nvidia.com/

Copyright (c) 2014-2015, NVIDIA CORPORATION. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of NVIDIA CORPORATION nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Third-Party Licenses for VR-Vantage Applications

VR-Vantage applications use a variety of third-party libraries. Developers who want to use these libraries may be required to purchase developer's licenses. Please see section 1.2 in *VR-Forces Front-End Developers Guide* for details.



I. Introduction, Installation, and Startup

VR-Forces is a computer generated forces (CGF) application and toolkit. VR-Forces is implemented using separate front-end (graphical user interface) and back-end (simulation engine) applications that allow tremendous flexibility for managing your simulation needs.



1. Introduction to VR-Forces

This chapter provides a high-level description of VR-Forces features.

Overview 1-3
- Entity-Level and Aggregate-Level Simulation 1-5
- Realistic Display of Vehicles and Terrain (3D View) 1-6
- Create Complex Scenarios 1-7
- Simulation Object Types Supported 1-8
- Mission Planning 1-10
- Simulation Object Tasks 1-10
- Scripted Tasks and Set Data Requests 1-11
- Tactical Graphics 1-11
- Terrain Agility and Composability 1-12
- Dynamic Terrain 1-12
- Crowd Behaviors and Pattern of Life 1-12
- Flexible, Intuitive Graphical User Interface 1-13
- Overlays 1-13
- Behaviors 1-13
- Observer Attach Modes 1-14
- Special Effects and Simulation Object Information Visualization 1-14
- Dynamic Ocean 1-14
- Lighting Effects 1-14
- Accurate Vehicle Positioning 1-15
- Batch Mode Operation 1-15
- Remote Control 1-15

The VR-Forces Toolkit 1-16
- Plug-in Architecture 1-16

DI-Guy 1-17

Helpful Utilities 1-18

Third-Party Software and Content 1-19
- SilverLining 1-19

Triton Ocean SDK 1-20
GL Studio 1-20
SpeedTree 1-21
3D Models, Terrain, and Graphical Content 1-21
OpenSceneGraph 1-22
osgEarth 1-22
Distributed Simulation Standards Supported 1-22

1.1. Overview

VR-Forces is a computer generated forces (CGF) application and toolkit. VR-Forces is implemented using separate front-end (graphical user interface) and back-end (simulation engine) applications that allow tremendous flexibility for managing your simulation needs. (For information about how this works, please see "The VR-Forces Program," on page 3-2.)

VR-Forces provides 3D and 2D views of your simulated world, integrated into one graphical user interface (GUI). The 2D view provides you with insights into a simulated battle by overlaying simulation objects and information onto 2D views of tactical, strategic, and visual databases. The 2D view answers typical questions about the placement of forces and how the terrain might affect the engagement more easily than do three-dimensional displays. It is ideally suited for viewing simulations over large areas of open terrain.

The 3D view provides three-dimensional, perspective views of the terrain and entity models. It is ideal for scenarios that require situational awareness, simulation debugging, or after action review. It is particularly useful for placing entities with precise location and orientation and for working in terrains in which entities can be at multiple altitudes at a given X, Y coordinate, such as urban terrains.

The 3D view includes exaggerated reality (XR) features. Model scaling and additional model sets help to provide a common operational picture of the simulated world.

VR-Forces is compatible with the Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) simulation standards. It supports multiple database formats and terrain servers. VR-Forces is interoperable with other MAK products, such as VR-Vantage Stealth, VR-Engage, and MAK Data Logger.

The VR-Forces front-end is built with the VR-Vantage Toolkit. The VR-Vantage Toolkit is based on OpenSceneGraph[1], so you can leverage value-added plug-ins built by the OSG community and MAK partners.

1. "OpenSceneGraph is an Open Source, cross-platform graphics toolkit for the development of high-performance graphics applications such as flight simulators, games, virtual reality and scientific visualization. It is based on the concept of a SceneGraph, providing an object-oriented framework on top of OpenGL." (http://www.openscenegraph.org)

Table 1-1 lists VR-Forces visual features supported in the 3D and 2D views for entity-level scenarios.

Table 1-1: VR-Forces features by projection (Sheet 1 of 2)

Feature	3D	2D
Terrain agility and composability	X	X
Attach modes	X	X
Observer movement modes		
Observer frame	X	X
Level observer frame	X	
Laptop	X	X
2D level observer frame		X
Model sets		
3D models	X	X (top view)
3D colorized models	X	X (top view)
2D icons	X	X
Other Features		
Audio	X	X
Cockpit displays	X	
Communications lines	X	X
Detonation and muzzle flashes	X	
DI-Guy characters	X	
Display units configuration	X	X
Dynamic ocean	X	
Dynamic terrain	X	X
Emitter volumes	X	X
Entity labels	Text panels	Configurable symbol decorations
Environment effects	X	
Extruded tactical graphics	X	
Fire and detonation lines	X	X
Forward+ lighting	X	
High dynamic range lighting	X	
Illumination Flares	X	
Intervisibility	X	X
Paged terrain	X	X

Table 1-1: VR-Forces features by projection (Sheet 2 of 2)

Feature	3D	2D
Parachuting	X	X
Radar jamming	X	X
Range rings	X	X
Remote draw API	X	X (2D graphics only)
Sea spray effects	X	
Sensors	X	X (Stealth only)
Sensor contact lines	X	X
Shader-based effects maps	X	
Shadows	X	
SpeedTree trees	X	
Streaming buoys and beacons	X	X
Streaming terrain	X	X
Tactical Data Link support	X	X
Tactical graphics	X	X
Tactical smoke	X	
Task visualization	X	X
Terrain profiles	X	X
Tracers	X	
Track histories	X	X
Trailing effects	X	
View controls	X	X
Wireframe view	X	X
Zoom	X	X

1.1.1. Entity-Level and Aggregate-Level Simulation

VR-Forces supports two approaches to simulation – entity-level simulation and aggregate-level simulation. In entity-level simulations, VR-Forces models individual entities, such as ships, tanks, automobiles, airplanes, and people. Interaction, and for military simulations, warfare, takes place between individual entities. You can aggregate entities into hierarchical units, but for the most part they are still modeled as individuals.

Aggregate-level simulations do not model individual entity interactions and combat. They are concerned with the high level movement and interaction of large numbers of personnel and equipment. Aggregate-level combat is modeled based on the relative strengths and weaknesses of hierarchical units in personnel, equipment, and location, not on the engagement of the individual entities that they represent. Aggregate-level simulations are used for modeling the operational tempo of large area/theater level missions overseen by command staff level officers. This is useful for training staff officers and stimulating Command and Control systems (for example, C2, C4I, C4ISR, and Mission Command systems). (Aggregate-level simulations include some entity models to facilitate certain types of munition attacks. However, they use the same modeling approach as units. They do not behave like the comparable entities in entity-level simulations.)



Most of the information in VR-Forces documentation applies to both types of modeling. The procedures for creating scenarios, adding simulation objects and tactical graphics, assigning tasks and writing plans, and displaying scenario information are almost identical in both types of scenarios. Therefore, if you want to know how to create a scenario that uses aggregate-level modeling, for example, just follow the procedures in Chapter 7, *Creating and Running Scenarios*. Or, if you want to learn about assigning tasks in entity-level scenarios, just read Chapter 25, *Assigning Tasks*.

If a section only applies to entity-level scenarios, it has an entity icon () in the margin. If a section only applies to aggregate-level scenarios, it has a unit icon () in the margin.

For conceptual information about the differences between entity-level modeling and aggregate-level modeling, please see:

- "Entity-Level Modeling and Aggregate-Level Modeling," on page 13-9.
- Chapter 21, *Entity-Level Unit Modeling*.
- Chapter 22, *How Aggregate-Level Modeling Works*.

1.1.2. Realistic Display of Vehicles and Terrain (3D View)

VR-Forces provides a realistic display using terrain and moving-model geometry in the OpenFlight format. It includes many 3D vehicle models, some of which display movement of articulated parts, such as turrets and rotors. These models can change to show a damaged or destroyed state. You can provide your own models and map them to object types. For details about terrain, please see Chapter 57, *Composing Terrains*. For details about models, please see Chapter 89, *Mapping Models and Effects*.



Aggregate-level scenarios do not use 3D vehicle models.

1.1.3. Create Complex Scenarios

VR-Forces lets you create complex scenarios to meet your simulation requirements. You can:

- Create and delete simulation objects.
- Create, delete, and edit tactical graphics.
- Assign a sequence of tasks and set data requests to a simulation object as a plan.
- Assign tasks and set data requests independently of plans.
- Create global plans that run independently of any simulation object.
- Observe the behavior of simulation objects in response to their environment.
- Create, save, and load scenarios that consist of:
  - Terrain database information.
  - Simulation objects, such as individual platforms, humans, and units.
  - Tactical graphics.
  - A plan for each simulation object.
- Manage hostility relationships.
- Manage spot reports.
- Determine line-of-sight between simulation objects.
- Aggregate and disaggregate, and expand and collapse units.
- Create overlays and add tactical graphics and simulation objects to them.
- Schedule scenario events to inject non-simulated input, such as text, video, and graphics to scenario participants.
- Apply weather conditions to a particular area of the terrain.

1.1.4. Simulation Object Types Supported

Using VR-Forces, you can interactively add simulation objects to a simulation and you can aggregate them into higher echelon units. Simulation objects provided for entity-level scenarios include:

- Ground entities, such as:
  - M1A2 Tank
  - T-80 Tank
  - HMMWV Utility Vehicle
  - BTR-80 APC
  - M2A2 Bradley IFV
  - IEDs
  - Civilian vehicles, such as automobiles, police cars, and fire engines.
- Air entities, such as:
  - F-16 Falcon
  - F/A-18 Hornet
  - A-10 Thunderbolt
  - VTUAV Cypher
  - MiG-29 Fulcrum
  - AH-64 Apache
  - Civilian aircraft, such as Boeing 747 and Airbus A320.
- Surface and subsurface entities, such as:
  - AAVC7A1 Landing craft
  - LCAC
  - Aircraft carrier
  - Guided missile frigate
  - Submarine
  - Sailboats
  - Cargo ships.
- Life forms, such as:
  - Dismounted infantry
  - Civilian.
- Units, such as platoons and companies.

Aggregate-level scenarios have simulation object types such as:

- ADA PLT
- Air Attack company
- Artillery battery
- Engineering company
- Ground attack aircraft
- Mechanized infantry company
- Mortar platoon
- Rifle company
- Tank company
- Destroyer
- Submarine.

Simulation Object States

You can interactively modify aspects of a simulation object's state, including:

- Speed.
- On-board stores such as fuel and ammunition.
- Location.
- Heading.

For more information about changing a simulation object's state, please see Section VI, *Set Data Requests*.

You can modify simulation object parameters by editing the object parameter database and other configuration files.

Embarkation

Simulation objects can embark on and disembark from other simulation objects, for example, dismounted infantry can embark on a truck, HMMWVs can embark on an LCAC, and fixed-wing aircraft can land on and take off from aircraft carriers.



To use embarkation in HLA federations, you must use RPR FOM 2.0, draft 17 or later. For information about specifying the correct FOM, please see "Starting VR-Forces," on page 4-3 and "Specifying the RPR FOM Version," on page 5-20.

1.1.5. Mission Planning

You can create plans for simulation objects that contain tasks, set data requests, conditional statements and global commands. You can save plans and edit them.

Conditional statements specify that tasks will be executed only if a specific condition is true or false. Some of the conditions tested are:

- Is a simulation object in a specific area?
- Is a simulation object destroyed?
- Is a simulation object to the left or right of a phase line?

In addition to conditional statements that are checked only once (`If` statements), VR-Forces supports triggers, `While` statements, and `Wait Until` statements.

Global Plans

Global plans exist independently of any simulation object. They allow you to assign tasks and set data requests to simulation objects and execute other commands. In essence, they allow you to script the sorts of ad hoc actions that a VR-Forces user could take through the GUI. This provides a great amount of flexibility to automatically modify simulation objects and tactical graphics in response to events.

For more information, please see Chapter 37, *Writing Plans*.

1.1.6. Simulation Object Tasks

You can assign tasks to simulation objects as part of a plan or independently. The tasks supported in entity-level scenarios include:

- Move to an object.
- Patrol along a route.
- Patrol between two objects.
- Follow an entity.
- Fire for effect.
- Take off and land.
- Fly to a heading.

The tasks supported in aggregate-level scenarios include:

- Move to an object.
- Patrol along a route.
- Patrol between two objects.
- Construct a minefield (and other combat engineering objects).
- Breach an obstacle.
- Bomb a location.

Independently assigned tasks override the tasks in a plan, allowing you to respond to the course of a simulation as events transpire, without stopping to edit a simulation object's plan. For more information, please see Chapter 25, *Assigning Tasks*.

1.1.7. Scripted Tasks and Set Data Requests

You can create new tasks and set data requests by writing scripts using the Lua scripting language. Input dialog boxes are created automatically. The scripted action can be added to the Task or Set menu. This feature requires some programming skills, but you do not need a developers license or a C++ development environment. For more information please see Chapter 31, *Creating Scripted Tasks and Sets*.

Reactive tasks are a special kind of scripted task. They monitor the simulation and get executed only if some special condition is met. You can manage assignment of reactive tasks in the GUI. For more information, please see "Reactive Tasks," on page 25-11.

Background processes are another special type of scripted task. They run continuously and manage ongoing simulation object activities. There is no user interface for managing them. For more information, please see "Background Processes," on page 32-32.

You can organize scripts into groups called Behavior Sets. By assigning Behavior Sets to forces you can control whether or not scripts are available to simulation objects in those forces. For more information, please see "Using Behavior Sets to Manage Scripts," on page 25-18.

1.1.8. Tactical Graphics

Using VR-Forces, you can interactively create and name tactical graphics, such as:

- Points
- Phase lines
- Lines
- Areas
- Obstacles.

Simulation objects are aware of some types of tactical graphics and can be assigned tasks such as following a route, or moving between points. VR-Forces can also determine if specified simulation objects are in an area and this information can be used as part of mission planning.

Aggregate-level scenarios support combat engineering objects, which can be areal, point, or linear objects that could impede, damage, or defend a simulation object.

Other types of tactical graphics are for informational and tactical management purposes and do not affect the simulation. For more information, please see Chapter 39, *Overlays*.

1.1.9. Terrain Agility and Composability

VR-Forces allows you to build your terrain at runtime using a variety of database and vector formats. It can load OpenFlight DTED, CTDB, CDB, MetaFlight, and GDB (MAK terrain format) databases. It can load feature data from shapefiles, S-57 files, and GDB files. You can also superimpose geospatial images such as GeoTiffs or other raster image files on the terrain.

You can load local terrain data and you can stream data from networked servers using the osgEarth earth file format.



VR-Forces only supports geocentric terrains in osgEarth earth files.

You can save these composed terrains in the MTF file format. For details, please see Chapter 57, *Composing Terrains*. For a list of supported file formats, please see "Supported File Formats," on page 54-2.

VR-Forces supports the following coordinate systems:

- UTM
- Geocentric
- MGRS
- Latitude, longitude
- Database.

You can overlay raster images, such as CADRG, BMP, and geotiff files on terrain databases.

1.1.10. Dynamic Terrain

Terrain state can be changed using switch nodes in the terrain models. Terrain changes can be caused by munition damage, user initiated changes, and using entity tasks.

Cultural features can be embedded in the terrain and navigation data can be dynamically regenerated so that entities can navigate around and through the objects.

1.1.11. Crowd Behaviors and Pattern of Life

You can quickly create civilian crowds that have inherent behaviors of wandering and fleeing munition detonations. You can assign tasks to crowds, such as gathering around a person or point. For details, please see Chapter 19, *Crowds and Multiple Simulation Objects*.

You can use the pattern of life feature to generate civilian and vehicular traffic using various spawn patterns to simulate typical background traffic for scenarios. For details, please see Chapter 18, *Generating Traffic (Pattern of Life)*.

1.1.12. Flexible, Intuitive Graphical User Interface

VR-Forces's graphical user interface (GUI) allows you to manipulate VR-Forces from menus, dockable control panels, and the keyboard.

The VR-Forces display includes the following features:

- Manipulate the simulation map by zooming and panning.
- Select and track simulation objects.
- Turn display features on and off.
- Display a hierarchical, echelon view of simulation objects and an embarkation view.
- Display entity labels that include several types of information.
- Display line-of-sight (intervisibility) for simulation objects.

You can hide and display the control panels, keep them docked to the screen, or undock them to make more space available for the display window. The GUI controls can be hidden for full-screen demonstrations.

The GUI follows standard conventions for modern windowing systems. For additional details, please see Appendix 83, *GUI Controls, Layouts, and Behaviors*. For descriptions of individual windows and dialog boxes, please see online help.

1.1.13. Overlays

Overlays mimic clear plastic sheets that a planner might use to draw graphics on top of a terrain map. All objects are automatically assigned to overlays. You can add overlays and move objects between overlays. You can hide the display of objects on an overlay. For details, please see "Overlays," on page 39-1.

1.1.14. Behaviors

Simulation objects understand how to interact with their environment. For example, they can:

- Respond appropriately to other simulation objects and tactical graphics, for example, avoiding simulation objects, waiting for orders, and so on.
- Respond to detected targets, for example, deciding which of multiple targets to acquire, which weapon to fire, and what ammunition to use.
- Follow a route.

1.1.15. Observer Attach Modes

The observer represents your eyepoint in the simulated terrain. VR-Forces offers attach modes that let you:

- Manually control the observer's position and orientation. (The observer represents the user's view, or eyepoint, in the simulated world.)
- Follow a simulation object, maintaining a consistent positional offset, with a matching heading.
- Mimic a vehicle's position and orientation. You can place the observer inside the vehicle for a driver's-eye view, or fly outside of it, moving in tandem with the vehicle.
- Tether the observer to a simulation object, maintaining a consistent positional offset without changing the observer's heading.
- Automatically track a vehicle's movement from a fixed viewpoint, as if watching from a control tower.

For more information, please see Chapter 51, *Moving the Observer*.

1.1.16. Special Effects and Simulation Object Information Visualization

VR-Forces includes features that help you understand the interaction between simulation objects that add a realistic tone to interactions. These features include:

- Fire and detonation lines.
- Track histories.
- Range rings.
- Radio communications lines.
- Trailing effects such as footprints, tire tracks, and wakes.

1.1.17. Dynamic Ocean

VR-Forces provides dynamic ocean visualization in the Stealth and Out-the-Window observer modes. The ocean shows waves, swells, and spray effects. Surface entities have realistic wakes and buoyancy behavior. Changes to wind speed and direction affect wave action.

1.1.18. Lighting Effects

VR-Forces supports a variety of lighting effects, including:

- Dynamic lighting
- High dynamic range lighting
- Ocean planar reflection
- Lens flare
- Crepuscular (sun) rays.

Shader-based Effect Maps

VR-Forces supports several types of shader-based effect texture maps. Shaders are computer programs that simulate the effect of light on the terrain and objects in the simulation. VR-Forces makes extensive use of them for visualization effects.

VR-Forces supports the following types of shader-based effect maps:

- Normal, or bump, maps. Normal maps give terrains the appearance of relief, such as a rocky landscape.
- Specular maps. Specular maps affect the highlight color of objects.
- Ambient occlusion maps. Ambient occlusion maps model areas that do not receive direct light, such as cracks and crevices and shaded areas of terrain and models. These areas are lit only by ambient light.
- Reflection maps. Reflection maps affect the reflectivity of surfaces, such as windows. Reflection maps reflect objects in the sky, not the terrain.
- Emissive maps. Emissive maps control the emissivity of whatever they are applied to based on the current ambient light values.

Lighting effects are described in Chapter 45, *Lighting Effects*.

1.1.19. Accurate Vehicle Positioning

VR-Forces can smooth the trajectories of moving vehicles to compensate for discontinuous positional data. The ground clamping feature can ensure that all ground entities are placed correctly on the terrain surface. For details, please see "Trajectory Smoothing," on page 6-21 and "Using Ground Clamping," on page 54-9.

1.1.20. Batch Mode Operation

Batch mode allows you to run a scenario multiple times or run several different scenarios, without direct action through the graphical user interface. This is useful for running unattended testing or to test the results of a scenario under differing conditions. You can record the results of batch mode simulations using the MAK Data Logger, for later review. For more information, please see "Running VR-Forces in Batch Mode," on page 7-45.

1.1.21. Remote Control

You can control the VR-Forces observer remotely through view control messages, which can be generated by MAK applications such as MAK Data Logger, and by programs that use the VR-Vantage Control Toolkit. For details, please see "Controlling the Observer from Other Applications," on page 51-35.

You can also send view control messages from plans. For details, please see "Sending View Control Messages," on page 37-37.

1.2. The VR-Forces Toolkit

The VR-Forces toolkit lets you develop CGF applications. You can enhance the capabilities of the VR-Forces simulation engine and GUI, or integrate the simulation engine into your own user interface. The toolkit has several APIs. The Simulation API gives you control over:

- Behaviors
- Components
- Object types
- Parameters
- Messages
- Resources
- Tactical graphics
- Plans
- Tasks.

The Remote Control API allows you to control a VR-Forces simulation engine from a remote application, such as a graphical user interface (GUI). The Terrain API lets you manipulate terrain databases.

The VR-Forces front-end is built with the VR-Vantage Toolkit. The toolkit is included with VR-Forces, allowing you to modify or extend the front-end. For details about the VR-Forces Toolkit, please see *VR-Forces Developers Guide*.

1.2.1. Plug-in Architecture

You can modify the VR-Forces front-end and back-end by rebuilding the applications or by writing plug-ins. In fact, much of the basic functionality of the front-end is implemented using plug-ins rather than as part of the main executable. Plug-ins are dynamically linked libraries that VR-Forces can load at startup. The rationale for choosing which method to use is described in API documentation. You can manage the use of plug-ins in the front-end. For details, please see "Managing Plug-ins," on page 4-33.

1.3. DI-Guy

VR-Forces uses DI-Guy software and content for human character animation.

VR-Forces comes with DI-Guy functionality built-in and with a large set of DI-Guy characters, appearances, and animations. A VR-Forces customer does not need a DI-Guy license of any kind to use DI-Guy animated characters in a VR-Forces-based application (including custom applications built using the VR-Forces Toolkit).

However, you need a DI-Guy Developer's license if you want to:

- Develop new character types, or alter the geometry, appearances, textures, or motions of DI-Guy characters.
- Directly access the DI-Guy API to modify or extend the way that VR-Forces uses the DI-Guy API or access DI-Guy functionality that MAK has not already integrated into VR-Forces.

DI-Guy libraries, models, textures, motion files, and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of DI-Guy software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

For more information, please see "DI-Guy Animation," on page 28-7 and "DI-Guy Characteristics (Appearance, Head, Body Weight)," on page 34-15, and "DI-Guy Configuration Files," on page 73-3 and "Editing a Lifeform's Visual Model and Animation," on page 69-11.

1.4. Helpful Utilities

VR-Forces includes the following utilities for configuring and troubleshooting VR-Forces:

- The Simulation Object Editor is a graphical user interface that lets you add and remove component systems from simulation objects and specify common simulation object parameters. You can create new simulation objects in the editor. You can also create new simulation model sets. The Simulation Object Editor is also used to manage tactical graphics. For details, please see Chapter 67, *The Simulation Object Editor and SMSs*.
- The vrfMsgDump utility displays the contents of VR-Forces messages discovered on the network. For details about vrfMsgDump, please see *VR-Forces Developers Guide*.
- The Boundary Generation Tool lets you calculate the boundaries of a high resolution terrain inset for inclusion in an earth file. For details, please see"Using the Boundary Generation Tool," on page 59-16.
- The medfTool lets you encrypt 3D model files. For details, please see "Compressing Model Files," on page 90-35.
- The mftTool lets you process MetaFlight terrains so that they will work in VR-Forces. For details, please see "The mftTool," on page 63-4.
- osgearth_cache. This utility lets you cache terrain data from earth files offline. For details, please see "Caching earth File Terrain Data Offline," on page 64-5.

1.5. Third-Party Software and Content

VR-Forces includes software and content licensed from third parties, including:

- SilverLining™: real-time sky and 3D cloud rendering from Sundog Software.
- Triton Ocean SDK™: 3D ocean and ship wakes from Sundog Software.
- GL Studio®: interactive cockpit instrumentation and HMI from DiSTI®.
- SpeedTree®: animated, 3D foliage from Interactive Data Visualization (IDV).
- 3D models, terrain, and graphical content from Simthetiq, RealDB, TurboSquid, and TerraSim.
- OpenSceneGraph: an open source 3D graphics toolkit hosted at http://www.openscenegraph.org.
- osgEarth: an open source streaming terrain plug-in by Pelican Mapping, at http://osgEarth.org.

The run-time and developers' rights for VR-Forces customers vary from vendor to vendor. Please read this section carefully to understand which rights are included with VR-Forces licenses, and which rights require additional third-party licenses.

1.5.1. SilverLining

VR-Forces uses SilverLining software and content to compute lighting and to render the sky, clouds, sun, and moon. SilverLining is developed by Sundog Software (http://www.sundog-soft.com).

Most VR-Forces customers do not need to buy a SilverLining license of any kind. In general, a VR-Forces customer has the right to use the SilverLining technology that is built into VR-Forces in any VR-Forces-based application (including custom applications built using the Toolkit). However, there are two exceptions:

- If you want to extend or change the way that MAK has integrated SilverLining into VR-Forces by writing directly to the SilverLining API, then you need a SilverLining Developer's License.
- If you are building a commercial video game or other "mass-market" (consumer-level) commercial product on top of the VR-Forces Toolkit, and the resulting product includes the SilverLining functionality, then you need a SilverLining Developer's License.

SilverLining libraries, textures and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of SilverLining software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.2. Triton Ocean SDK

VR-Forces uses the Triton Ocean SDK to render 3D oceans and ship wakes. Triton Ocean is developed by Sundog Software (http://www.sundog-soft.com).

Most VR-Forces customers do not need to buy a Triton Ocean license of any kind. In general, a VR-Forces customer has the right to use the Triton Ocean technology that is built into VR-Forces in any VR-Forces-based application (including custom applications built using the Toolkit). However, there are two exceptions:

- If you want to extend or change the way that MAK has integrated Triton Ocean into VR-Forces by writing directly to the Triton Ocean API, then you need a SilverLining Developer's License.
- If you are building a commercial video game or other "mass-market" (consumer-level) commercial product on top of the VR-Forces Toolkit, and the resulting product includes the Triton Ocean functionality, then you need a Triton Ocean Developer's License.

Triton Ocean libraries, textures and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of Triton Ocean software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.3. GL Studio

VR-Forces uses GL Studio software and content to render interactive cockpit instrumentation displays. GL Studio is developed by DiSTI (http://www.disti.com).

VR-Forces is delivered with several generic cockpit displays. You can use the built-in cockpit displays in any VR-Forces-based application (including custom applications built using the Toolkit), without a GL Studio license of any kind.

If you want to build custom cockpits using the GL Studio editor or GL Studio API, you need a GL Studio Developer's license. If you want to execute custom cockpits in VR-Forces-based applications (regardless of whether the custom cockpits are built by you or a third party), you need GL Studio Run-Time licenses.

GL Studio libraries and graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of GL Studio software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.4. SpeedTree

VR-Forces uses SpeedTree software and content for animated real-time 3D foliage and vegetation. SpeedTree is developed by Interactive Data Visualization (IDV) (http://www.speedtree.com).

A VR-Forces customer does not need a SpeedTree license of any kind to use the SpeedTree functionality that is built into the VR-Forces applications that MAK delivers. You may build and run plug-ins to our out-of-the-box applications without a SpeedTree license, as long as your plug-ins do not need to directly access the SpeedTree API.

However, you need a SpeedTree Developer's license if you want to:

- Write a plug-in that uses the SpeedTree API directly to extend or modify the way that SpeedTree is integrated into VR-Forces.
- Use the VR-Forces Toolkit to develop new applications, and you want your applications to include SpeedTree functionality.

SpeedTree libraries, models, textures, and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of SpeedTree software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.5. 3D Models, Terrain, and Graphical Content

VR-Forces includes a rich set of 3D models for vehicles, weapons, cultural features and urban clutter (signs, barriers, lampposts, and so on). It also includes several sample terrain databases to help you get started with VR-Forces, and to help demonstrate VR-Forces. Much of this content is derived from 3D data that we have licensed from third parties:

- Many of the high-quality vehicle models come from Simthetiq (http://www.simthetiq.com).
- Some models come from Real DB (http://www.realdb.qc.ca/company).
- Some of the middle-eastern building models and urban clutter objects that are used in our sample terrain databases were licensed from Turbosquid (http://www.turbosquid.com).

All of this content is distributed in the VR-Forces package solely so that VR-Forces can use it. Use of any of the VR-Forces models, textures, terrains, or other content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.6. OpenSceneGraph

VR-Forces uses OpenSceneGraph for its underlying scene graph representation, rendering, file loading, and many other capabilities. OpenSceneGraph is an open source, cross-platform graphics toolkit for the development of high-performance graphics applications. The OpenSceneGraph source repository is maintained by Robert Osfield at http://www.openscenegraph.org. It is distributed under the OpenSceneGraph Public License (OSGPL), which is based on the GNU Lesser General Public License (LGPL).

MAK will provide modified source code for OSG upon request, as per the OSGPL license. Please contact support@mak.com to obtain the download links for our modified source.

1.5.7. osgEarth

VR-Forces uses osgEarth to import streaming terrain elevation and imagery data. The data can be streamed from external servers and sources or from a directory on the computer running VR-Forces. osgEarth is an open source plug-in to OpenSceneGraph, maintained by Pelican Mapping at http://osgEarth.org. osgEarth is distributed under the GNU Lesser General Public License (LGPL).

MAK will provide modified source code for osgEarth upon request, as per the OSGPL license. Please contact support@mak.com to obtain the download links for our modified source.

1.6. Distributed Simulation Standards Supported

VR-Forces supports the HLA 1.3 specification, the current draft of the IEEE 1516 C++ API maintained by the SISO Dynamic Link Compatible RTI API Product Development Group, and HLA Evolved (IEEE 1516-2010). It also supports the Distributed Interactive Simulation (DIS) protocol, versions 4, 5, 6, and 7.

VR-Forces has built-in support for the HLA RPR-FOM and can support other FOMs through the FOM Mapping feature. For information about FOM mapping, please see "Specifying a FOM Mapper," on page 5-20. Please see *VR-Forces Release Notes* for a list of the versions of DIS supported.

VR-Forces supports time management for HLA exercises.



2. Installing VR-Forces

This chapter explains how to install VR-Forces on your computer. It also explains how to install other software required for use with VR-Forces.

Installing VR-Forces 2-2
- Installing VR-Forces on Windows 2-2
- Installing VR-Forces on a Linux System 2-3
- Uninstalling VR-Forces on Windows 2-3

The VR-Forces Directory Structure 2-4

Installing and Setting Up the MAK License Manager 2-6
- Specifying the License Server 2-7

Installing an RTI 2-9
- Installing the MAK RTI 2-10

Localizing the Graphical User Interface 2-11
- Translating Other Interface Files 2-14
- Translating VR-Forces Scripts and Console Messages 2-15
- Applying the Language Files 2-16

2.1. Installing VR-Forces

A full VR-Forces installation includes the application, data, the License Manager software, and for HLA operation, an RTI (or Runtime Infrastructure.)

VR-Forces has an application installer and a data package (*.MAK-targz*) that has multiple files. You must run the application installer and then install the data package. The application installer can automatically install the data package if you configure it to do so. MAK also provides optional data packages that have additional terrains.

To install the VR-Forces toolkit, you must enter an installation code. Be sure you have received your code from your MAK salesperson before you start the installation process. If you do not have an installation code, you can install the runtime files. When you receive your installation code you can install the toolkit.

Before you install VR-Forces, please read the Release Notes to see if there are any special instructions for installation.

2.1.1. Installing VR-Forces on Windows

When you install large applications on Windows, there may be a delay of up to several minutes from the time you try to run the setup program to the time that an installation dialog box is displayed. This is due to how Windows scans setup programs before it executes them. If you experience this problem, turning off User Access Control can reduce or eliminate this delay.

Windows versions of VR-Forces are provided as executable installer files on DVD, or as downloaded files. The installers are named to indicate the compiler used to build that version of VR-Forces.

➤ To install VR-Forces, run the application installer. Follow the instructions in the installation wizard. You can specify that the application installer automatically find and install the data package (it must be in the same directory as the application installer), or you can specify that the MAK Data Setup Wizard open and let you select the data package to install.

2.1.2. Installing VR-Forces on a Linux System

Linux versions of VR-Forces are provided as compressed tar files on DVD, or as downloaded files.



If you install VR-Forces in one directory and user data in another directory, VR-Forces will not start.

To install VR-Forces on Linux:

1. Create the directory in which you want to install VR-Forces.
2. Copy the tar file to the install directory.
3. Uncompress and untar the application file:

   `tar -vxzf application.tar.gz`

   where *application* is the product release identification.
4. The data package consists of multiple files that have the extension *MAK-targz*. Uncompress and untar the data package files and copy them into the installation directory, or uncompress them and redirect the output, as follows:

   `tar -xzf makData<version>.MAK-targz -C /path-to-vrforces4.7`

2.1.3. Uninstalling VR-Forces on Windows

When you uninstall VR-Forces, the uninstaller does not uninstall the data package. You must delete the files manually.

2.2. The VR-Forces Directory Structure

The VR-Forces installation process creates a directory structure like the one shown in Figure 2-1. On PCs, the default installation directory is *C:\MAK\vrforces4.7*.)

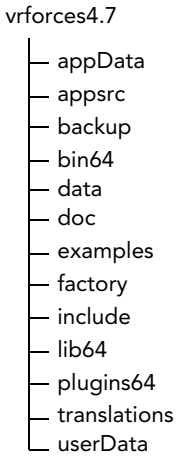


Figure 2-1. VR-Forces directory structure

Table 2-1 describes the contents of the directories installed by VR-Forces.

Table 2-1: Contents of VR-Forces directories (Sheet 1 of 2)

Directory	Directory contents
./appData	Schema, model definitions, settings, configuration files, and other data.
./appsrc	Source files and project files for API developers.
./backup	Backup configuration files.
./bin64 or *./bin*	Product executables and DLLs.
./data	The *./data* directory has several subdirectories that contain: ◆ Bitmaps and icons. ◆ Simulation model sets. ◆ Terrain database source files. ◆ Visual model data.
./doc	VR-Forces documentation.
./examples	Sample code for toolkit users.
./factory	Backups of the application data and settings provided by MAK.
./include	Header files for use by API developers.
./lib64 or */lib*	Library files.
./plugins64 or *./plugins*	Plug-in DLLs.

Table 2-1: Contents of VR-Forces directories (Sheet 2 of 2)

Directory	Directory contents
./translations	Files and an executable for localizing the GUI.
./userData	Scenarios, terrain files, and other files that get created by users as they use VR-Forces.

2.3. Installing and Setting Up the MAK License Manager

Before you can use a MAK product, you must obtain a valid license file, install the MAK License Manager, and configure the license server and client machines. The License Manager uses a client-server architecture, so you do not need to install the License Manager on every computer on which you install MAK products. You only need to install it on the computer that you will use as the license server.



If you have already installed the License Manager for another MAK product, you do not have to install it again. You just need to make sure you have licenses for your newly installed products.

The License Manager installer is included on MAK installation media. It is separate from the product installers. You can download the installers from our web site at https://www.mak.com/support/license-support.

Complete installation and configuration instructions are included with the License Manager installer. Instructions are also available at the license support page.

Some customers use dongle licenses instead of running a license server. Instructions for using dongles are in the License Manager documentation.

2.3.1. Specifying the License Server

The first time you run a MAK application on a particular computer, the License Setup dialog box opens (Figure 2-2). It prompts you to enter the hostname of the license server and optionally, a port number.

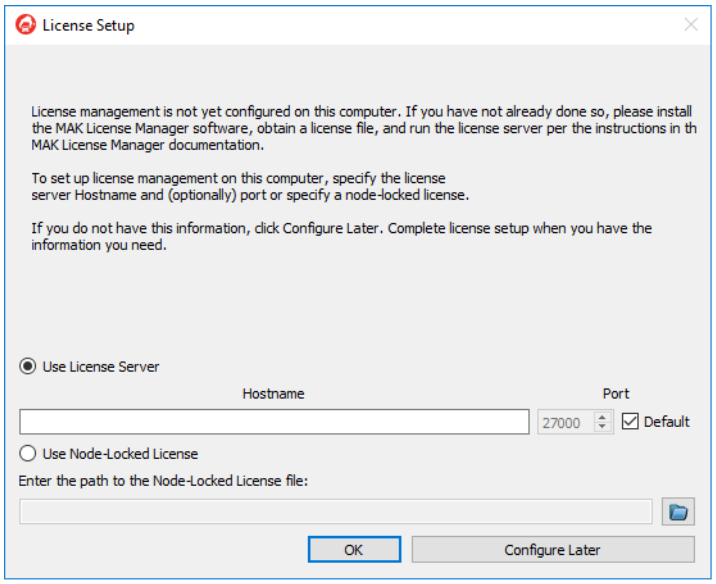


Figure 2-2. License Setup dialog box

If you do not know the hostname of the license server, click Configure Later. When you have the hostname, you can start the application again and complete the dialog box. You will not be able to run any MAK applications until you set up license management.

If you know the hostname, type it in the Hostname box and then click OK. The application will start.

Under limited circumstances, MAK issues node-locked licenses. If you have a node-locked license, select the Use Node Locked License option and enter the path to the license file.



- If you are running MAK products on the license server machine, it is also a client, so you must specify the license server on that machine, too.
- If you change the license server, the saved configuration will no longer be valid and the License Setup dialog box will open the next time you start a MAK application.
- You can clear the saved license configuration by deleting the cache file. On Windows, it is *C:\Users\user_name\AppData\Roaming\MAK\licenses<n>.txt*. (The *AppData* directory is hidden by default.) On Linux, it is *.mak/license<n>.txt*. (There may be more than one cache file, for example, *licenses1.txt* and *licenses2.txt*.)

The MAKLMGRD_LICENSE_FILE Environment Variable

As an alternative to using the License Setup procedure described in the previous section, you can configure the license server in an environment variable. The MAKLMGRD_LICENSE_FILE environment variable identifies the server machine. If you set this environment variable, it overrides the settings stored by the License Setup procedure.

The syntax for the environment variable is: `@Server_name`. For example, if the server machine is oak, set the environment variable to `@oak`.

The following sections explain how to set environment variables on the different platforms that MAK products run on.

Windows

To add the MAKLMGRD_LICENSE_FILE in Windows:

1. On the Start menu, click the Settings button. The Settings window opens.
2. In the search box type environment. A list of choices is displayed.
3. Select Edit the System Environment Variables. The System Properties dialog box opens.
4. Click Environment Variables. The Environment Variables dialog box opens.
5. In the System Variables group box, click New. The New System Variable dialog box opens.
6. In the Variable Name field, enter MAKLMGRD_LICENSE_FILE.
7. In the Variable Value field, enter `@server_name`, where `server_name` is the name of the license server.
8. Click OK to back out of each dialog box and set the variable.

Linux

On Linux, you set environment variables in your *.cshrc* (or equivalent startup file). Set the variable similarly to the following example:

```
setenv MAKLMGRD_LICENSE_FILE @oak
```

If you are using the sh or bash shells, you set environment variables in your *.profile* file (or *.bashrc*). Set the variable similarly to the following example:

```
MAKLMGRD_LICENSE_FILE=@oak
export MAKLMGRD_LICENSE_FILE
```

Do not put spaces around the equal (=) sign.

You are ready to run the license server and use your new licenses or MAK products.

2.4. Installing an RTI

An RTI is a software library (and perhaps supporting executables) that implements an HLA Interface Specification. In HLA, applications exchange FOM data through RTI calls, which means that all HLA applications need to use an RTI.

Because of differences in the low-level network mechanisms used by different RTI implementations (which include, but are not limited to packet layout), applications that want to interoperate in the same federation execution must use the same RTI implementation.

Because RTIs are usually provided as dynamic libraries that implement a fixed API, a federation can often switch from one RTI implementation to another between runs (without even recompiling the applications), but during each run, all participants must agree on which RTI to use, much as they must also agree on which FOM to use.

For the most recent information about the RTI versions supported by MAK products, please see the release notes for your MAK application.

2.4.1. Installing the MAK RTI

To install the MAK RTI, follow the instructions in Chapter 2 of *MAK RTI Users Guide*.

Configuring Your System to Use the MAK RTI

The RTI dynamic libraries must be located somewhere on your dynamic library search path. The path is specified in the PATH environment variable.The path is indicated by an environment variable as follows:

- Linux — LD_LIBRARY_PATH.
- Windows — PATH.

The MAK RTI needs to know where to find the following configuration files:

- The federation configuration file (*.fed*, *.fdd*, or *.xml*) for your federation execution (required).
- The RID file (*rid.mtl*) (optional).

Put the configuration files in the directory from which you are running, or set the environment variable RTI_CONFIG to the directory that contains them.

Running Applications with the MAK RTI

To run a MAK application with the MAK RTI:

1. Be sure the license server is running.
2. Start the application. The RTI Assistant will prompt you to choose an RTI configuration.
3. Choose a configuration. If necessary start the rtiexec.
4. Click Connect. The application should run.

In many cases, you do not need to run the rtiexec to use the MAK RTI, however you can run the rtiexec if you want to. (It is required to use certain features of the MAK RTI.) For more information, please see your RTI documentation.



If you are running VR-Forces, it is recommended that you run the rtiexec.

2.5. Localizing the Graphical User Interface

You can localize your MAK product's graphical user interface so that it uses languages other than English. Your product includes a file called *./translations/product_untranslated.ts*, for example *logger_untranslated.ts* or *vrforces_untranslated.ts*. The file contains entries for all of the application-specific menu and dialog box text. The Qt Linguist tool, which is included with VR-Forces, lets you map the English text provided by MAK to your chosen language.



The *untranslated.ts* file does not include all of the text that may be displayed in the GUI or in console and error messages.

Some strings, such as those on the Create menu, are in configuration files. Others are in settings files. You can translate these strings by identifying the appropriate configuration file and translating the strings.

VR-Forces has a utility that can scan system scripts and SMS files and create translation files for user-visible text. For details, please see "Translating VR-Forces Scripts and Console Messages," on page 2-15.

To edit a resource file:

1. Start the Qt Linguist utility. It is *./translations/linguist.exe*
2. In the Qt Linguist window, choose **File** → **Open**. Select the file *./translations/product_untranslated.ts*. The Settings for filename dialog box opens (Figure 2-3).

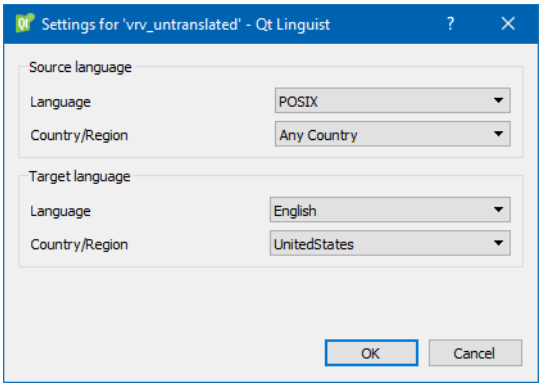


Figure 2-3. Linguist settings dialog box

3. Select values for the source language, target language, and target region or country.
4. Click OK. The translation file is opened and a list of classes is displayed on the left side of the window (Figure 2-4).

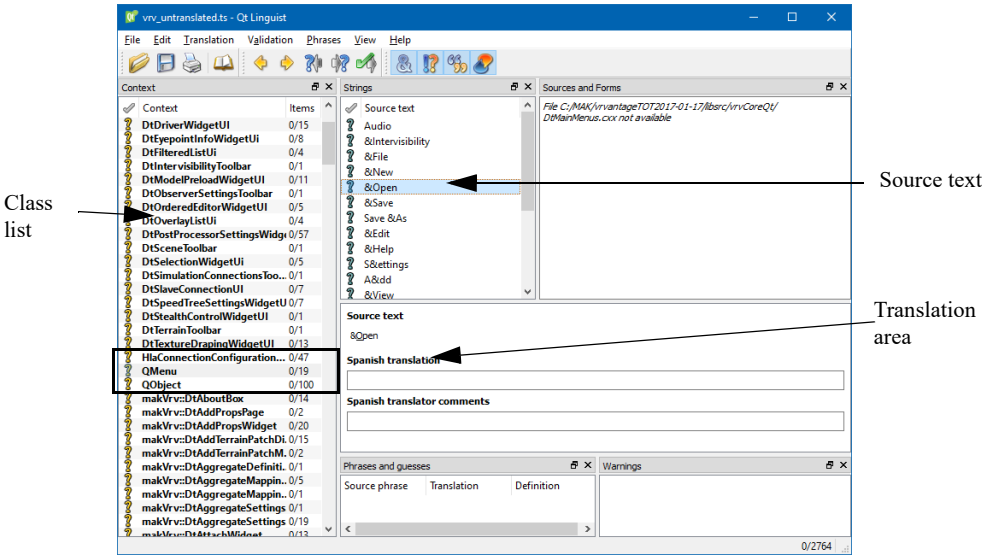


Figure 2-4. Linguist window

5. To translate text, you select each class in turn and translate the text associated with that class. You do not have to understand anything about what the classes mean, you just have to translate the text. In Figure 2-4, *QMenu* is selected. The Strings window lists text that can be translated. Text that has not been translated has a question mark in the checkmark column.
6. In the Source Text list, select the word you want to translate. The text is displayed under Source Text in the translation area (the window below the source text list.) Figure 2-5 shows the text Open selected.

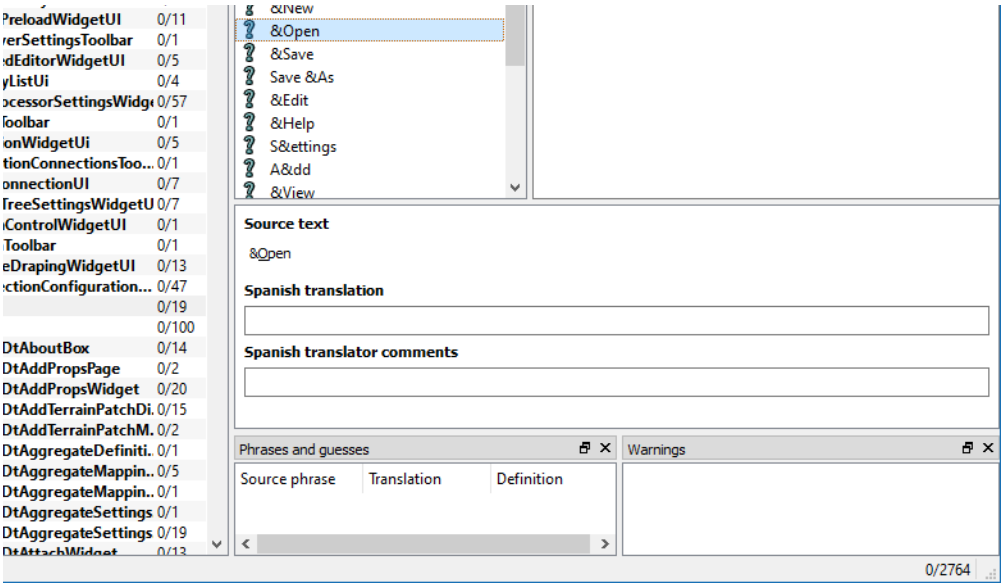


Figure 2-5. Text selected for translation

7. Place the cursor in the text box below *Language* Translation, where *Language* is the target language that you selected when you opened the translation file.
8. Type the translation text.
9. When you are done and you are certain of the translation, select the Done and Next button ( ) on the toolbar. A green check mark replaces the question mark (Figure 2-6). If you move to a different text string without clicking Done and Next, a yellow question mark is displayed for the string you translated. When you translate all elements for a class, a green check mark replaces the question mark next to the class name.

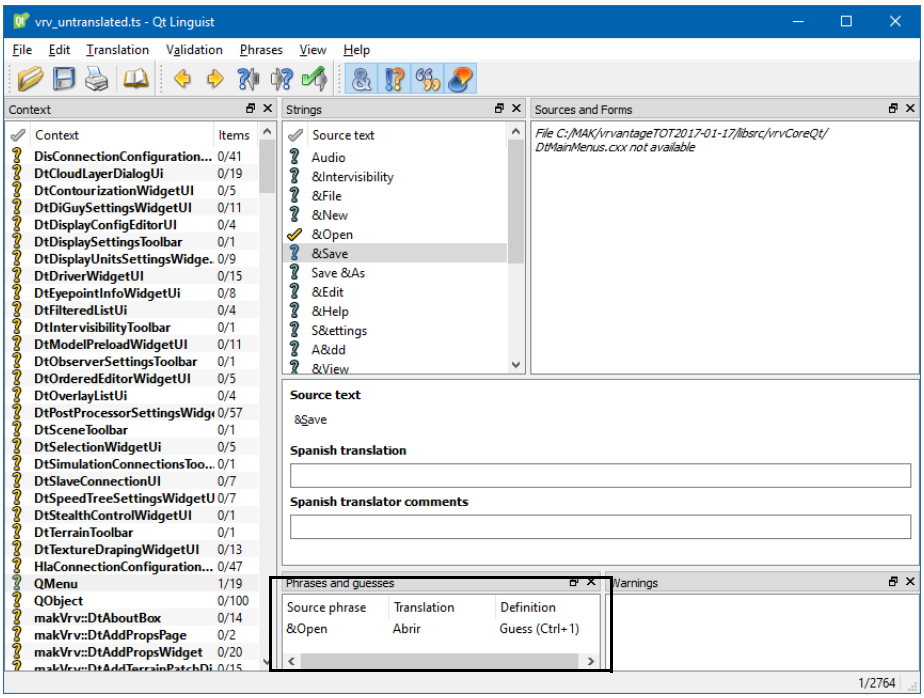


Figure 2-6. Completed translation for Open

10. Repeat this process for each element of source text for each class.
11. Choose **File** → **Release**. The file is saved in the format *filename.qm*.
12. Rename the file to a name that indicates the language to which you have translated the file, using the two-character abbreviation typically used with your language. For example, a Spanish translation would be renamed *product_es.qm*.
13. Save your translated *.ts* file.

2.5.1. Translating Other Interface Files

The *vrforces_untranslated.ts* file contains all the text used in the VR-Forces interface. However, there may be other text and messages that are not specific to VR-Forces that are generated by the software that implements the VR-Forces GUI. To translate this text, you need to translate the file *qt_untranslated.ts*. Follow the same procedure as you did for *vrforces_untranslated.ts*.

2.5.2. Translating VR-Forces Scripts and Console Messages

The *vrforces_untranslated.ts* file includes text for dialog boxes and interface elements created using the Qt Toolkit. However, there are many messages generated by code and the dialog boxes for scripts are generated programmatically, rather than through Qt ui files. Therefore they are not included in *vrforces_untranslated.ts*. VR-Forces has a utility that can scan all system scripts (those in a simulation model set) as well as some of the common SMS files, and create a TS file for each script and SMS. You can then translate the text in the individual TS files. This script also creates a TS file for DI-Guy files.

For this process to work, you must code messages in your scripts in a certain way. For details, please see "Coding Messages for Translation," on page 32-23. VR-Forces also has a way to code console messages so that they can be extracted and translated. For details, please see 17.8.1, Coding Object Console Messages for Translation, in *VR-Forces Developers Guide*.

Generating TS Files for Scripts and Simulation Model Sets

If you code your scripts to support translation, you create TS files by running the translationFileCreate utility. By default this application scans *./data/simulationModelSets* and its subdirectories for all system scripts and some generic files, and *./appData/settings* for DI-Guy files.

The utility creates an *untranslated.ts* file for each lua script. This file is saved in the directory that contains the script. It also creates a TS file for each simulation model set, which is saved in the SMS directory, and a *diguy_untranslated.ts* file, which is saved in *./translations*.

When you translate a script file, the translated file must remain with the back-end(s) that are hosting the scripts. The rest of the translated files must exist on the user machine. The *.qm* file for DI-Guy and VR-Forces must live in the translations directory and the *.qm* file for the simulation model sets with the *.sms* files.

The syntax for translationFileCreate is as follows:

```
translationFileCreate.exe [--directory filename] ...
   [--smsfile string] ... [--nodiguy] [--nosms]
   [--noscripts] [--] [-v] [-h]
```

Table 2-2: translationFileCreate arguments

Argument	Description
`--directory` *directory_name*	Directory to scan for files of any of the supported types (lua, sysdef, xml, mtl) as per creation rules (scripts, systems, supporting). Accepted multiple times.
`(-h | --help)`	Displays usage information and exits.
`--nodiguy`	Do not parse the DI-Guy configuration files.
`--noscripts`	Do not parse Lua files for translation.

Table 2-2: translationFileCreate arguments

Argument	Description
`--nosms`	Do not parse the simulation model set directories for their supporting translatable items.
`--smsfile` *`SMS_filename`*	Specific SMS file to use when translating simulation model set directories. Accepted multiple times. If not specified, processes all SMS files.
`(-v | --version)`	Displays version information and exits.
`(-- | --ignore_rest)`	Ignores the rest of the labeled arguments following this flag.

Generating TS Files for Object Console Messages

If the messages in your source code are coded for extraction to TS files, you can generate an *untranslated.ts* file for them. To generate TS files, run:

```
lupdate source_directory -ts my_new_translations.ts
```

The resulting TS file is saved to the location pointed to by the path given to `-ts`.

2.5.3. Applying the Language Files

To put the language file into effect, use one of the following methods:

- Set your computer's environment to the proper language (LANG environment variable).
- Set the locale to the proper language.
- Start VR-Forces with the `-G` *`locale`* command line option, where *`locale`* is the two-letter country code for the local language.



3. VR-Forces Application Concepts

This chapter describes at a conceptual level the components of a VR-Forces simulation and how they work together.

The VR-Forces Program 3-2
Front-end and Back-end Concepts 3-2
How Front-ends and Back-ends Work Together 3-3
How VR-Forces Back-ends are Identified 3-4
VR-Forces Sessions 3-4
Coordinating Multiple Front-ends 3-6
Working with Multiple Back-ends 3-7
Objects 3-8
The Object Parameter Database 3-8
Local Objects and Remote Objects 3-9
Representing and Managing Time in VR-Forces 3-10
Simulation Time 3-10
Time of Day 3-10
Exercise Clock Modes 3-12
Interactions 3-13
Advanced Terrain Navigation 3-13
Managing VR-Forces Settings 3-13
Synchronizing Settings Among VR-Forces Installations 3-16
Global Settings and Observer-Specific Settings 3-16

3.1. The VR-Forces Program

Unlike most computer applications, which have one executable file, VR-Forces has two executable files: a front-end, or graphical user interface (GUI) and a back-end, or simulation engine. The front-end executable is vrfGui. The back-end executable is vrfSim*protocol*, where *protocol* is DIS, HLA13, HLA1516, or HLA1516e.

By separating the simulation engine from the visualization program, VR-Forces allows you to run multiple front-ends, back-ends, or both. If you run multiple front-ends, several VR-Forces users can work together to create and manage scenarios. Using session IDs, you can specify which front-ends control which back-ends. The front-ends also display simulation objects received over the network from other simulations participating in an exercise. (You have a limited ability to interact with these remote objects. For more information, please see "Local Objects and Remote Objects," on page 3-9, "Writing Plans for Remote Entities," on page 37-41 and "Attaching Components to Remote Entities," on page 81-2.)

If you run multiple simulation engines, the work of simulating simulation objects and tactical graphics can be shared by multiple computers, which should improve performance. If you save a scenario, VR-Forces records which objects are simulated by each back-end, so that the configuration can be reproduced the next time you run the scenario.



In this manual, most procedures apply to interaction with the front-end. For the sake of simplicity, unless we are discussing a subject that applies specifically to either the front-end or back-end, we refer to VR-Forces in the sense of the combined interactions of both executables.

3.2. Front-end and Back-end Concepts

Although the separation of front-ends and back-ends, and the ability to run multiple instances of each adds to the power and flexibility of VR-Forces, you need to understand how the various executables interact to ensure that your scenarios are created and run smoothly. In this section, we describe:

- How front-ends and back-ends work together.
- How back-ends are uniquely identified.
- Use of sessions.
- How to coordinate multiple front-ends.
- How scenarios are loaded and saved when you run multiple back-ends.

3.2.1. How Front-ends and Back-ends Work Together

The VR-Forces front-end is an interface to the simulation engine. It communicates with the back-end by sending messages over the network. (This relationship exists even if you are not connected to a network.) It displays the state updates and interactions that the back-end generates and sends over the network. Figure 3-1 illustrates loading a scenario and running a simulation.

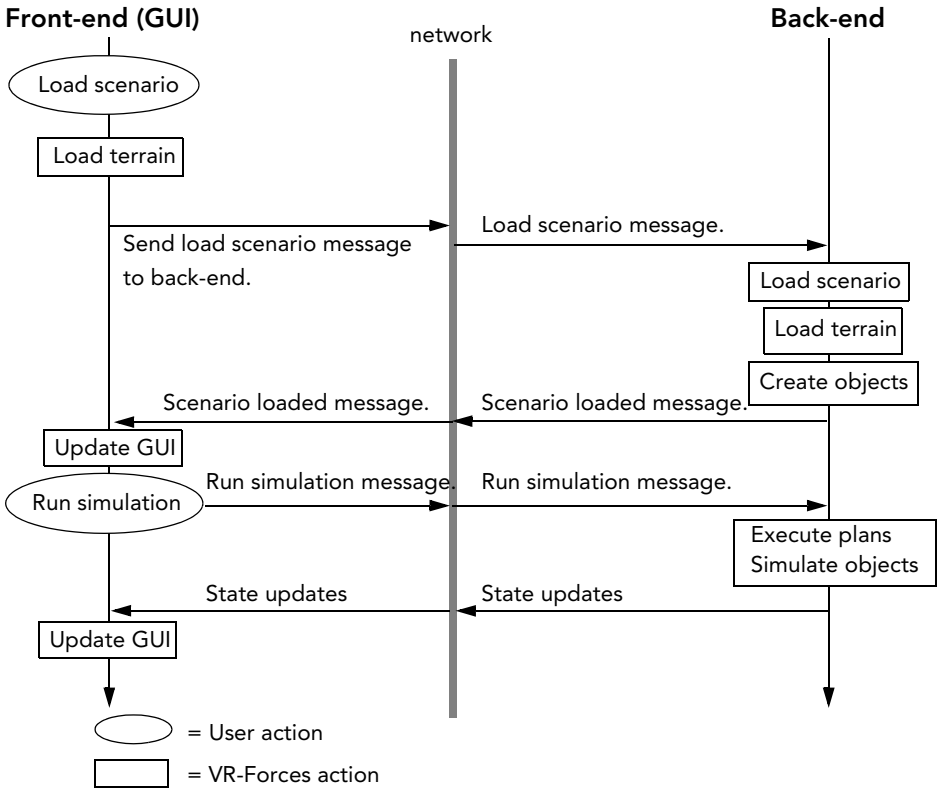


Figure 3-1. Front-end and back-end interaction

3.2.2. How VR-Forces Back-ends are Identified

If you run multiple back-ends, you can specify which back-end you want to use when you create simulation objects and tactical graphics. This feature allows you to spread the simulation load across multiple PCs. However, to do this, each back-end must have a unique ID. VR-Forces back-ends are identified by their site ID and application number. Each back-end must have a unique site:application identifier. You assign the site:application identifier when you start an application. It is up to you to make sure that there are no duplicate identifiers. (For details, please see "Starting VR-Forces," on page 4-3.)



The site ID and application number concepts are borrowed from DIS. They apply to VR-Forces when it is running in HLA as well.

3.2.3. VR-Forces Sessions

VR-Forces uses sessions to help manage the relationships between front-ends and back-ends. Back-ends are always part of a session. Front-ends can join a session or operate independently of one. When a front-end is joined to a session, it can control the back-ends in that session. When a front-end is not joined to a session, it cannot control back-ends. However, it can open a terrain and view exercises running on its port. In this case it is just a viewer.

Use of sessions helps ensure consistency among front-ends that are controlling a common set of back-ends. The session mechanism ensures that:

- All front-ends and back-ends in a session run the same scenario and use the same terrain.
- When a front-end joins a session that is running a scenario, it automatically loads the correct terrain for the scenario.
- If any front-end in a session tries to create a scenario or load a scenario, and a scenario is already running, the user is warned that this action will destroy the currently running scenario.
- If any front-end in a session loads a scenario or creates a new scenario, all of the back-ends and other front-ends in that session take the same action.

Sessions provide the following additional benefits:

- They allow you to specify which front-ends control which back-ends. This feature could be useful, for example, if you want one set of VR-Forces applications to simulate friendly forces and another set to simulate opposing forces. If all applications were in the same session, participants could send orders to the simulation objects in the opposing force. By using separate sessions, this is not possible.

  In Figure 3-2, the front-end with session ID 1 can control the back-ends with that same session ID. The front-end with session ID 2, controls the back-end with session ID 2.

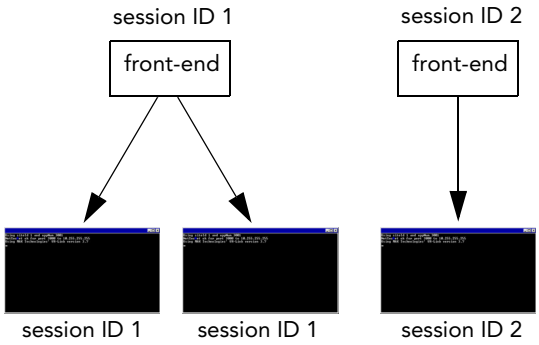


Figure 3-2. Using session ID to control back-ends

- Sessions allow you to load different terrain databases in different VR-Forces executables that are part of the same exercise. This feature may be helpful if you are participating in an exercise that covers a large geographical area, but only need to observe portions of that total terrain. Loading databases that cover a subset of the total exercise zone could improve the performance of your computer or the resolution of the terrain database.

  In Figure 3-3, the two sessions have loaded databases that cover different (but overlapping) portions of California, USA. The entire exercise would cover the terrain covered by two databases.



Figure 3-3. Using session IDs to load different terrain databases

If two different sessions use terrain databases that overlap, note the following possible sources of confusion:

- The front-ends in each session will see simulation objects from the other session that are simulated in the common portions of the terrain. However, the front-end for one session will have no ability to control the simulation objects that are simulated by the other session.
- If you save the scenario in one of these sessions, it only saves the simulation objects simulated in the session. It does not save the simulation objects from the other session, even though they may be visible in a front-end.
- If one of the sessions pauses or rewinds, the front-end in the other session will see the simulation objects in the overlapping region pause or return to their origin.

For more information, please see "Specifying a Session ID," on page 4-9 and "Managing a Front-end's Session Connection," on page 4-15.

3.2.4. Coordinating Multiple Front-ends

Figure 3-1 shows how a VR-Forces front-end sends messages to a back-end and displays the simulation messages sent out by the back-end. When you run VR-Forces with multiple front-ends, any of them can send control messages to the back-ends and all of them display the state updates. However, you would not want users at different front-ends to send conflicting or overlapping messages. For example, you would not want two different users to try to load the same scenario. Therefore, when you run multiple front-ends, bear the following points in mind:

- Only one front-end in each session can load a scenario or create a new one (they can all create the objects in the scenario).
- Any front-end can save a scenario. However, this could lead to multiple, unsynchronized versions of a scenario. It is probably best to reserve saving of scenarios to a specific front-end.
- Any object that you create in any of the front-ends is visible in the others and can be managed from them.
- You can close the scenario from any front-end.

VR-Forces provides some help managing multiple front-ends. For example, if one front-end loads a scenario, VR-Forces automatically asks the other front-ends to load the appropriate terrain. And if you close a scenario in one front-end, VR-Forces automatically prompts the other front-ends to close their terrains.

3.2.5. Working with Multiple Back-ends

As noted previously, you can distribute simulation of simulation objects among multiple back-ends. The available back-ends are listed in the Selected Simulation Engine Toolbar (Figure 3-4). If you are running multiple back-ends, when you create simulation objects, you specify which back-end you want to create them on by selecting a back-end in the toolbar. (For more information, please see "Selecting the Object to Create," on page 14-4.)

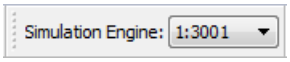


Figure 3-4. Selected Simulation Engine toolbar

When you run multiple back-ends, bear the following points in mind:

- When you save a scenario created using multiple back-ends, the object map file ("The Object Map File," on page 12-9) records each object and the site and application ID of the back-end on which it was simulated. When you want to replay that scenario, you should run a set of back-ends with the same site and application IDs. (If you do not, you can remap objects to the back-ends that you are running.)
- Multiple back-ends must all be started and ready when a scenario is loaded. There is no way for a required back-end to join an exercise after it has started and catch up to the current state of the exercise.
- If, during an exercise, a back-end drops out of the exercise, there is no good way for it to rejoin the exercise. If you load the scenario being run, it will be at the beginning, not at the point that the rest of the participants had reached.
- If you want a back-end to join an exercise after it has started, you must explicitly load the terrain database using the `-T` command-line option. The back-end will not receive any information about existing simulation objects and tactical graphics, but it can be used to create new simulation objects and tactical graphics going forward in the simulation.
- When you save a scenario that uses multiple back-ends, the final scenario files are composed of information generated by each back-end and sent to the front-end, where it is combined. Therefore, if any of the back-ends has dropped out of the exercise or is otherwise unavailable, the scenario data will be incomplete.

Remapping Back-ends

The previous section lists the issues you should consider when creating, saving, and loading scenarios that use multiple back-ends. If you choose to run a scenario that requires multiple back-ends and some of the back-ends are not available, VR-Forces lets you remap objects from the missing back-ends to the available back-ends. You can remap back-ends automatically or manually.

If a scenario is missing its object map (*.omp*) file, you can remap the objects and create a new object map file. By default, in combined mode all the simulation objects are remapped to the combined mode back-end. By default, in separate mode the remapping dialog box lists the missing back-end as "UNKNOWN".

You might also want to manually remap back-ends to load balance a scenario. For more information, please see "Load Balancing Scenarios," on page 7-30.

3.3. Objects

All simulation objects and tactical graphics are objects. The procedures for selecting, viewing, and organizing different kinds of objects are fairly similar. However since their purposes and use are quite different, for the most part we discuss simulation objects and tactical graphics separately.

3.3.1. The Object Parameter Database

VR-Forces creates objects based on parameter specifications in the object parameter database (OPD). In the OPD, object types are identified by an 8-field enumeration based on the SISO Enumeration Document (*Reference for Enumeration for Simulation Interoperability*). (The DIS enumerations have also been incorporated into the HLA RPR FOM.) In this manual, when we refer to entity or force type enumerations, we are referring to this set of enumerations. For information about the object parameter database and how to edit it, please see the chapters in Section XII, *Creating and Editing Simulation Models*.

The objects in the OPD are organized into simulation model sets (SMS). An SMS contains all of the objects and supporting configuration files needed to populate a scenario. VR-Forces provides the following SMSs:

- *EntityLevel.sms*. Used by entity-level scenarios.
- *AggregateLevel.sms*. Used by aggregate-level scenarios.
- *base.sms*. Included in *EntityLevel.sms* and *AggregateLevel.sms* for use in both types of scenarios.

3.3.2. Local Objects and Remote Objects

From the point of view of a particular VR-Forces back-end, a local object is any object that is simulated in that back-end. In that same back-end, a remote object is an object that is received over the network from an external application, such as another VR-Forces simulation engine, or a non-VR-Forces simulation. A given VR-Forces application participating in a networked exercise may know about a mixture of local and remote objects.

There are two categories of remote objects: objects simulated by VR-Forces applications, and objects simulated by applications other than VR-Forces. When VR-Forces learns about a non-VR-Forces remote object, it knows only what is transmitted in the standard DIS or HLA message. By contrast, VR-Forces remote objects share additional state information over the network, including a unique identifier (UUID), a string name, an echelon ID (if the simulation object is organized), and a string label.



The VR-Forces front-end allows you to manage all the objects in a VR-Forces session regardless of which back-end is simulating them. Therefore, in the interest of simplicity, we refer to simulation objects simulated by a VR-Forces back-end, whether local or remote, simply as simulation objects and we refer to simulation objects simulated by applications other than VR-Forces as remote simulation objects.

You cannot control remote simulation objects from VR-Forces. However, you can attach VR-Forces components such as sensors and radios to remote simulation objects and you can write plans that use these components. For more information, please see "Writing Plans for Remote Entities," on page 37-41 and "Attaching Components to Remote Entities," on page 81-2.

3.4. Representing and Managing Time in VR-Forces

VR-Forces maintains an exercise clock. The clock keeps track of elapsed simulation time, date, and time of day. In HLA federations, it may be affected by federation time. These concepts also have a relationship to wall-clock, or real time.

3.4.1. Simulation Time

Elapsed simulation time is the amount of time that has passed since the simulation started, and is expressed in days, hours, minutes, and seconds in the format 00:00:00:00.

Application developers can manipulate simulation time. Unless a local developer tells you otherwise, simulation time starts at 00:00:00:00.

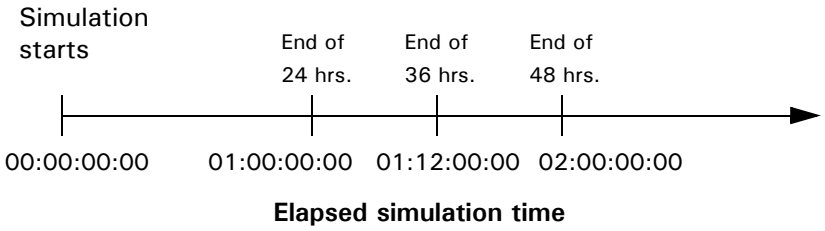


Figure 3-5. Simulation time

3.4.2. Time of Day

VR-Forces represents the day/night cycle with the Time of Day value. By default, it starts at 10:00 A.M in the time zone of the first simulation object created in a scenario, but can be set when you create a scenario. Time of day affects the simulation. For example, visual sensors are less able to detect objects at night than they are in the day.

Normally, time of day advances with simulation time, however you can change the time of day after the scenario is created, and even while the scenario is running. You can reset the time of day at any time through the environment settings, or through a plan with the Set Time of Day command. For details, please see "Introduction to Environment Conditions," on page 11-3.

The display of time of day in the interface is 00:00 in hours and minutes. Time of day does not track number of days; when the time of day reaches midnight, it starts over at 00:00.

Advancing Time Using Simulation Time or Wall Clock Time

By default, time advances based on the simulation time. When VR-Forces uses simulation time for time advancement, simulation objects dead reckon correctly in the GUI when you run faster or slower than real time. (This also affects animations, such as helicopter rotors and ocean movement.) However, simulation objects published by non-VR-Forces federates will not dead reckon correctly. VR-Forces provides the option of advancing time based on wall clock time. In this mode, the time scale is not taken into account for dead reckoning.

To specify how time advances:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 3-6).

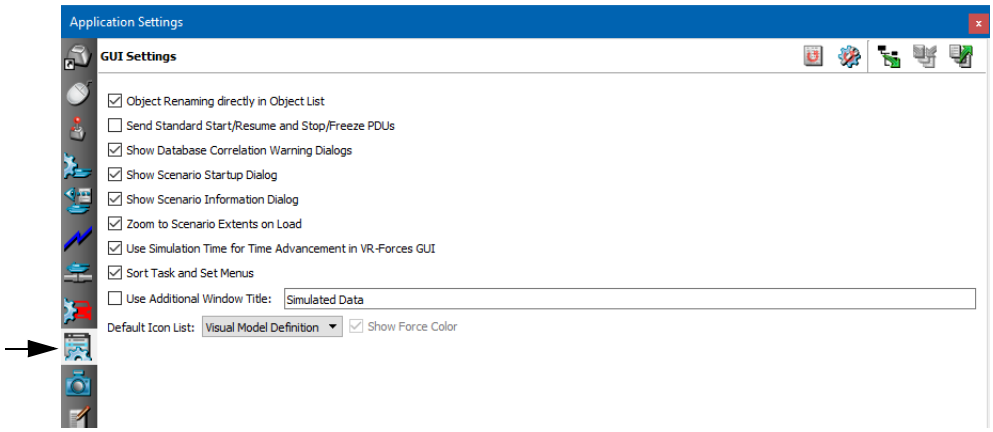


Figure 3-6. GUI Settings page

3. Select or clear the Use Simulation Time for Time Advancement in VR-Forces GUI check box.

3.4.3. Exercise Clock Modes

The exercise clock supports the following modes:

- Variable-Frame Run-To-Complete mode advances simulation time by the amount of time passed since the last time the exercise clock was ticked. This mode is typical for distributed, interactive simulations. It does not provide repeatable results. Do not use this mode in time-managed HLA federations.
- Fixed-Frame Best-Effort mode advances simulation time by a fixed amount each frame, even if a frame takes longer than the fixed amount to compute. If the simulation takes less than the fixed frame time to compute, it waits until the frame time has elapsed before starting another.

  This mode is suitable for viewing or interactive use if your hardware can handle the frame rate. It is suitable for distributed use only in time-managed HLA federations.
- Fixed-Frame Run-To-Complete mode advances simulation time by a fixed amount each frame, even if a frame takes longer than the fixed amount to compute, just as in Fixed-Frame Best-Effort mode. However, if the simulation takes less then the fixed frame time to compute, it does not wait for the remainder of the frame time to elapse before starting the next frame.

  This mode is most useful for situations where you want a simulation to run with internal consistency and high fidelity, and want it to run to completion, but do not need to observe the simulation. So, for example, you might run a simulation overnight and view the results the following day. Fixed-Frame Run-To-Complete mode is not suited for interactive use. It is suitable for distributed use only in time-managed HLA federations.



If you want to change the default clock mode, you must use the correct syntax. For information about how to set the exercise clock mode, please see Chapter 12, *Scenario Files*.

Running a Simulation Faster than Real Time

VR-Forces supports simulation at rates faster than real-time by applying a multiplier to the tick time. For details, please see "Changing the Simulation Speed," on page 7-34.



- Do not run your simulation at faster than real-time if you are interacting with other real-time simulations.
- Running a simulation faster than real-time can cause performance of simulation object models to degrade.

3.5. Interactions

VR-Forces displays interactions, such as detonations, that it receives in interaction messages or PDUs. You cannot select or otherwise manipulate them. You can only view them.

3.6. Advanced Terrain Navigation

VR-Forces supports advanced terrain navigation for entities based on the Autodesk® Gameware Navigation artificial intelligence software. Using Gameware Navigation Lab, you can create topological models of the virtual 3D environment that allow VR-Forces to compute, in real time, an optimized path for entity movement. As entities follow the path, they automatically avoid obstacles and other entities. The topological models are called navigation data.



Gameware Navigation Lab is included with VR-Forces. You can use it to view the navigation data included with VR-Forces terrain databases. If you want to generate navigation data, you must purchase an additional license.

3.7. Managing VR-Forces Settings

VR-Forces has the following types of settings:

- Independent, or display, settings are the individual settings managed on the Settings menu and the pages of the Display Settings dialog box. Changing an independent setting, such as enabling track histories or wireframe mode, does not affect other settings.

  Independent settings are categorized as being global settings or observer-specific settings. Global settings apply regardless of the observer mode. Observer-specific settings can be different for each observer mode. For more information, please see "Global Settings and Observer-Specific Settings," on page 3-16.

- Dependent settings are the schemas, model definitions, element definitions, and object type mappings that collectively determine how simulation objects and objects are modeled in the scene. Changing a dependent setting, such as a model definition may affect settings that depend on it, such as an entity definition.

VR-Forces manages settings as follows:

- As you make changes to settings during a session, it saves the changes, except for model definitions, element definitions, and simulation object type mappings.
- It remembers the settings in effect when you exit and uses them as the default settings in your next session.
- During any particular session, you can revert to the default settings.
- You can save the current settings to a file.
- You can import saved settings.
- You can restore settings to those that were in effect when VR-Forces was first installed (factory settings).

Dependent settings have two additional options that independent settings do not have:

- You can save or restore some types of settings by specific setting.
- You can merge settings from a file into the current settings.

The procedures for saving and restoring settings are the same regardless of the types of settings. You start the process by clicking a button on a settings dialog box page. The way that VR-Forces saves and restores depends on the type of setting, as follows:

- When you initiate a save or restore operation on independent settings, it saves or restores all of the settings on the dialog box page from which you started the save or restore. If you have changed a setting on another page, it is not affected.
- When you initiate a restore operation on dependent settings, related settings may also be restored. VR-Forces displays a message listing the settings that will be restored.
- When you save dependent settings, you either export all settings of a given type or selected settings. For details, please see relevant chapters in Section XIV, *Configuring Visual Models in the Visual Models Editor*.

Figure 3-7 illustrates the common settings management buttons. (The Visual Model Editors dialog box has some additional buttons.)



Figure 3-7. Settings buttons

- Revert to the default settings. Overrides any changes made to settings during this session.
- Restore the factory settings. Changes settings to the settings used when VR-Forces was installed. (The factory settings and default settings may be different.)
- Import settings from a file. Replaces the current settings with those stored in the selected file.
- Merge in settings from a file. Merging works as follows:
  - Saved settings that do not have a match in the current settings are added.
  - Saved settings replace matching settings.
  - Current settings that are not replaced (that is, for which there is no match in the saved settings) remain valid.

  Figure 3-8 illustrates how merged settings work. Setting A in the saved settings has no match in current settings, so it is added to the merged settings. Current settings B_2 and C_2 are replaced with saved settings B_1 and C_1. Current setting D has no match in saved settings, so it is retained in the merged settings.

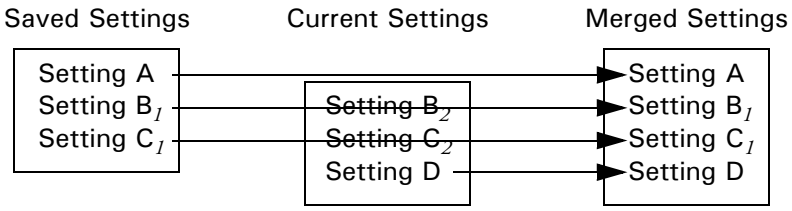


Figure 3-8. Merged settings

- Export all settings to a file. Saves the current settings to the selected file.
- Export selected setting to a file.

- The buttons described in this section are disabled on dialog box pages on which their function is not supported. This includes most of the pages on the Terrain Settings dialog box.
- VR-Forces saves model definition and element definitions to settings to *./appData/definitions*. Other settings get saved to *./appData/settings/vrfGui*.
- If you load an alternative settings file at startup, it is treated like an imported settings file.

3.7.1. Synchronizing Settings Among VR-Forces Installations

If your simulation is using several front-ends and back-ends on multiple computers, you may want to use the same settings across all of them, particularly if you have added model definitions and mappings to the factory settings. To synchronize settings, copy the data directories (*./appData*, *./data*, and *./userData*) from the computer on which you have updated them to the computers on which the other front-ends and back-ends are installed. Then import the settings into any Settings page that you customized on the source VR-Forces machine.

3.7.2. Global Settings and Observer-Specific Settings

VR-Forces has two types of independent settings – global settings and observer-specific settings.

Global settings affect the display regardless of which observer mode you are using. Global settings include simulation object labels, tactical graphics, and cockpit displays. The settings on the Settings menu are all global settings.

Observer-specific settings can be different for each observer mode. In fact, given a similar projection and model set, it is the different settings that distinguish one observer mode from another. For example, one difference between Stealth observer mode and Out-the-Window observer mode is that Stealth mode has tactical smoke enabled and Out-the-Window mode has it disabled.



4. Starting VR-Forces

This chapter explains how to start and stop VR-Forces and load terrain databases.

Starting VR-Forces 4-3
- Starting VR-Forces from the VR-Forces Launcher 4-4
- Starting Independent VR-Forces Executables 4-9
- Specifying a Session ID 4-9
- VR-Forces Startup Tutorial 4-9

The VR-Forces Window 4-11
- Opening New Windows 4-13
- Printing the VR-Forces Display 4-14
- Taking Screen Captures 4-14

Managing a Front-end's Session Connection 4-15
- Joining a Session 4-16
- Resigning from a Session 4-17
- Configuring Session Messages and Join at Startup 4-17

Using HLA Time Management 4-19
- VR-Forces Simulation Time Versus Federation Time 4-19
- Configuring Time Management for HLA Exercises 4-19

Opening a Terrain Database 4-21
- Loading a Terrain Database at Startup 4-22
- Displaying Terrain Correlation Warnings 4-23
- Closing a Terrain 4-23

Exiting VR-Forces 4-23
- Disabling the Quit Prompt 4-24

Configuring Simulation Connections 4-24
- Opening the Simulation Connections Configuration Dialog Box 4-25
- Adding a Simulation Connection 4-26
- Editing a Simulation Connection 4-27
- Simulation Connection Parameters 4-28

Deleting a Simulation Connection 4-30
Configuring Auto Connect 4-31
Displaying Connection Information 4-31
Managing Plug-ins 4-33
Loading Plug-ins 4-33
Adding a Plug-in 4-36
Specifying the DLLs for a Plug-in 4-37
Adding a Plug-in Configuration 4-38
Deleting a Plug-in Configuration 4-38
Deleting a Plug-in 4-39
Viewing a List of Loaded Plug-ins 4-39
The VR-Forces Log Files 4-40

4.1. Starting VR-Forces

In "The VR-Forces Program," on page 3-2, we describe the VR-Forces front-end and back-end applications. You can start front-ends and back-ends separately, which is called separate mode, or you can use the VR-Forces Launcher to start them together, which is called combined mode. The only real difference between the two startup modes is that in combined mode, when you shut down the front-end, the back-end automatically shuts down also.

By default, all front-ends and back-ends start with session ID 1. Front-ends automatically try to join a session. If no back-ends are running when a front-end starts, it does not join a session. However, if at a later time a back-end with the same session ID starts, the front-end automatically joins the session.



If you run VR-Forces for HLA, be sure that you have installed an RTI and configured your system to use it. For details, please see "Installing an RTI," on page 2-9.

For convenience you can create batch files or scripts to start VR-Forces. Please see Chapter 5, *Command-Line Options* for a list of command-line options.

4.1.1. Starting VR-Forces from the VR-Forces Launcher

You can run VR-Forces using DIS, HLA 1.3, HLA 1516, or HLA Evolved. For the HLA specifications, you can run using one of several different versions of the RPR FOM. With all these different possible startup configurations, particularly for HLA, it can be confusing to keep track of the options for starting VR-Forces. The VR-Forces Launcher simplifies the process of specifying the simulation standard to use when starting VR-Forces.

The VR-Forces Launcher lets you start in combined mode or separate mode.

To start VR-Forces using the VR-Forces Launcher:

1. On the Start menu, choose **MAK Technologies** → **VR-Forces** *configuration*.

   Where *configuration* is:

   – VR-Forces GUI
   – VR-Forces Simulation Engine
   – VR-Forces GUI + Simulation Engine.



Windows 10 has a flatter Start menu structure than previous versions of Windows. It does not support nested folders. If you are interested in using a nested menu for starting MAK applications on Windows 10, please read this blog: https://www.mak.com/learn/blog/entry/tech-tip-from-fred-starting-mak-applications-on-windows-10

To start the Launcher on Linux (or from the Windows command line), in a console window, enter the following command:

```
vrfLauncher options
```

For descriptions of the vrfLauncher command-line options, please see "Command-Line Options for vrfLauncher," on page 5-16. The basic startup configurations for vrfLauncher are:

– Start vrfGui and vrfSim (combined mode):

```
vrfLauncher
```

– Start vrfGui only:

```
vrfLauncher -F
```

– Start vrfSim only:

```
vrfLauncher -B
```

– Start in combined mode and do not display the Simulation Connections Configuration dialog box:

```
vrfLauncher --usePredefinedConnection connection
```

where *connection* is a previously defined configuration connection. If the connection name has spaces, you must surround it with quotes, for example, "DIS (7)". If you want to use one of the connections provided with VR-Forces, you must start VR-Forces from the Launcher at least once before you try to use it from the command line. This allows VR-Forces to populate the network interface values.

If this is the first time that you are running VR-Forces, or if you have not specified a default startup connection, the Simulation Connections Configuration dialog box opens (Figure 4-1).

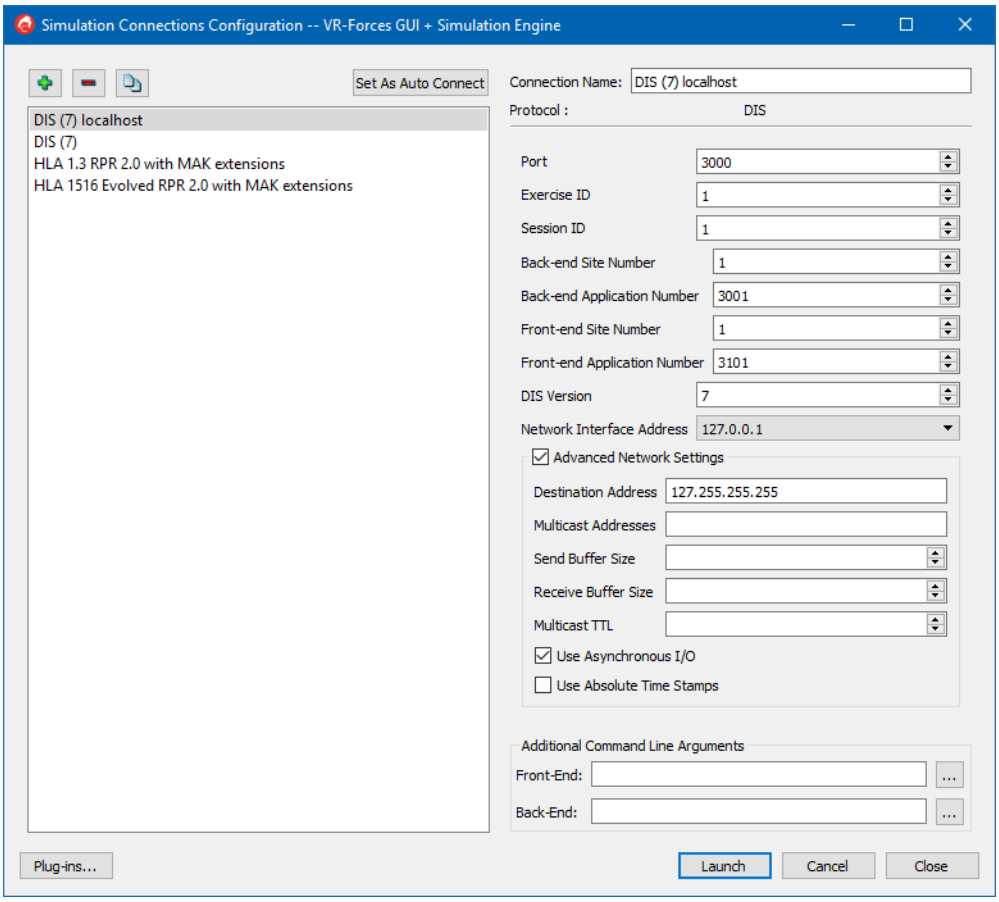


Figure 4-1. Simulation Connections Configuration dialog box (DIS)

2. In the Simulation Connections Configuration dialog box, select a connection configuration. (Optionally, create a new configuration. For details about creating and editing simulation connections, please see "Configuring Simulation Connections," on page 4-24.)

3. Optionally, edit the connection's parameters. If you change a parameter, it is saved automatically.

If you are using a lightweight RTI connection on Linux, you must enable Ignore Advisories or the simulation engine will not publish entities.

---

4. Optionally, specify additional command line arguments for the front-end, back-end, or both. For a list of command line arguments, please see Chapter 5, *Command-Line Options*.

- If you click the Browse button at the end of the Additional Command Line Arguments entry for the front-end or back-end, VR-Forces displays the command that will be executed when you click Launch. You can copy the command and use it in a batch file, script, or for debugging.
- If you start the Launcher from the command line and use the **`F--`** or **`B--`** syntax, those commands are not shown as additional command line arguments. However, they will be executed when you click Launch.

---

5. Click Launch. The VR-Forces executables start. When the VR-Forces applications finish their startup process, the Scenario Startup dialog box opens (Figure 4-2). It lets you quickly create or load a scenario using common options.



Figure 4-2. Scenario Startup dialog box

6. To create a scenario or load a scenario, select one of the options and click OK. The options are as follows:
   - New scenario. Select a terrain and simulation model set. VR-Forces creates a scenario using default scenario parameters. To specify scenario parameters when you create a new scenario, please follow the procedure in "Creating a Scenario," on page 7-3.
   - Load a scenario. Choose a scenario from the recently used list or select Load Scenario From Disk. If you choose the latter, when you click OK, a file chooser dialog box opens. This option is the same as choosing **File** → **Load Scenario** in the front-end.
7. If you do not want any of the options, click Close. (This closes the Scenario Setup dialog box; it does not close VR-Forces.)

Enabling and Disabling the Scenario Startup Dialog Box

If you do not want VR-Forces to display the Scenario Startup dialog box every time it starts up, you can disable it. You can enable the dialog box at a later time.

To enable or disable the Scenario Startup dialog box:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 4-3).

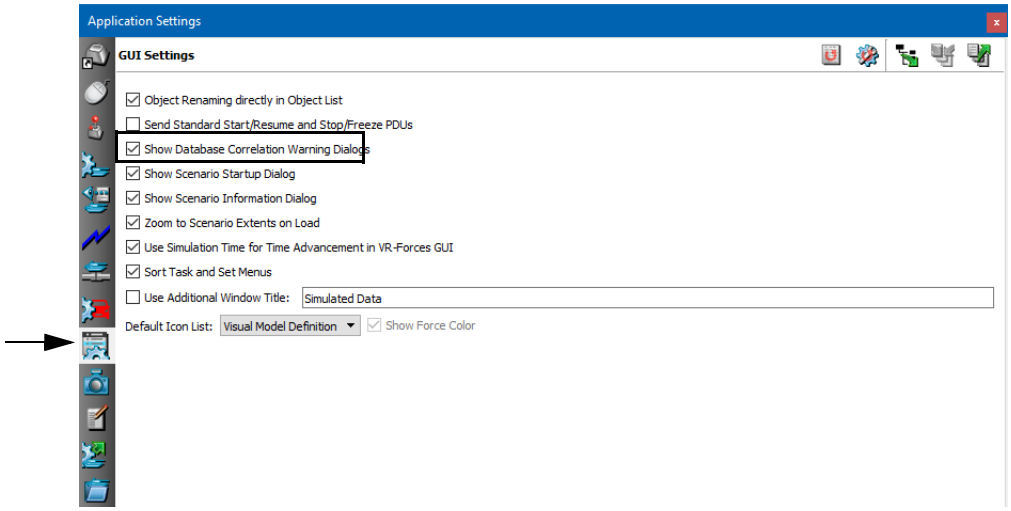


Figure 4-3. GUI Settings page

3. Select or clear the Show Scenario Startup Dialog check box.

> You can also disable display of the Scenario Startup dialog box by selecting Do Not Show This Dialog At Startup on the dialog box itself.

4.1.2. Starting Independent VR-Forces Executables

You can start independent VR-Forces executables from the launcher or the command line.

To start independent VR-Forces executables from the command line:

1. Open a console window.
2. Change to the *./bin64* directory.
3. Enter a command for each VR-Forces executable you want to run. You must specify the protocol to use. For vrfSim, the protocol is part of the executable name. For vrfGui, it is a command-line argument. Specify a unique site:application identifier for each executable, for example:

   ```
   vrfGui -s 1 -a 3000 {--dis | --hla13 | --hla1516 |
      --hla1516e} options
   vrfSimprotocol -s 1 -a 3001 options
   ```

   where *protocol* is `DIS`, `HLA13`, `HLA1516`, or `HLA1516e`.

A VR-Forces window opens for each front-end you start and a console window opens for each back-end. For an example of running multiple front and back-ends, please see "VR-Forces Startup Tutorial," on page 4-9.

4.1.3. Specifying a Session ID

➤ To assign a session ID, specify it in the VR-Forces Launcher or use the `-i` command line option.

```
vrfGui -s 1 -a 3000 -i 25 {--dis | --hla13 | --hla1516 |
   --hla1516e} options]
vrfSimprotocol -s 2 -a 3002 -i 25
```

where *protocol* is `DIS`, `HLA13`, `HLA1516`, or `HLA1516e`.

The default session ID is 1. You cannot change the session ID during runtime.

4.1.4. VR-Forces Startup Tutorial

This section is a brief example of how to run multiple front and back-ends on multiple computers. Examples are provided for DIS and HLA. For both protocols, assume the configuration in Figure 4-4. This configuration has two front-ends and three back-ends. Each is on a different computer. Assume that they are all running in the same session.

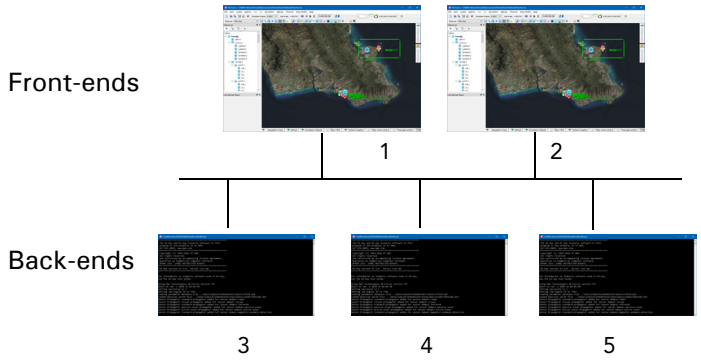


Figure 4-4. VR-Forces multicomputer configuration



As an alternative to starting individual front-ends and back-ends from the command line and specifying their protocol information in the command line, you can use vrfLauncher to start the front-ends and back-ends. Be sure to specify unique site and application numbers in vrfLauncher. For details about vrfLauncher command syntax, please see "Command-Line Options for vrfLauncher," on page 5-16.

DIS Configuration

In a DIS exercise, you typically specify a port so that the exercise traffic is isolated from other network traffic. Each front-end and back-end must have a unique site:application ID.

Run VR-Forces on each PC using the following commands:

- PC 1:
  ```
  vrfGui -s 1 -a 3000 --dis -P 1234
  ```
- PC 2:
  ```
  vrfGui -s 1 -a 3001 --dis -P 1234
  ```
- PC 3:
  ```
  vrfSimDIS -s 1 -a 3002 -P 1234
  ```
- PC 4:
  ```
  vrfSimDIS -s 1 -a 3003 -P 1234
  ```
- PC 5:
  ```
  vrfSimDIS -s 1 -a 3004 -P 1234
  ```

HLA Configuration

In an HLA exercise, you could simply specify the site:application ID and use default values to specify the HLA configuration parameters. Specification of the site:application ID is identical to that in DIS. For this example, we use HLA 1.3 and specify a FOM Mapper other than the default (RPR FOM 1).

Run VR-Forces on each PC using the following commands:

- PC 1:

```
vrfGui -s 1 -a 3000 --hla -x MAK-RPR-2.0
  --rprFomVersion 2.0017 -F MAK-RPR2-1-1.fed
```

- PC 2:

```
vrfGui -s 1 -a 3001 --hla -x MAK-RPR-2.0
  --rprFomVersion 2.0017 -F MAK-RPR2-1-1.fed
```

- PC 3:

```
vrfSimHLA13 -s 1 -a 3002 -x MAK-RPR-2.0
  --rprFomVersion 2.0017 -F MAK-RPR2-1-1.fed
```

- PC 4:

```
vrfSimHLA13 -s 1 -a 3003 -x MAK-RPR-2.0
  --rprFomVersion 2.0017 -F MAK-RPR2-1-1.fed
```

- PC 5:

```
vrfSimHLA13 -s 1 -a 3004 -x MAK-RPR-2.0
  --rprFomVersion 2.0017 -F MAK-RPR2-1-1.fed
```

4.2. The VR-Forces Window

VR-Forces shows all the simulation objects and interactions on a terrain database. The VR-Forces window has the following main components (Figure 4-5):

- Menus. Menus for accessing VR-Forces commands.
- Simulation control toolbars:
  - Simulation Control toolbar. Allows you to start, pause, and rewind a scenario.
  - Simulation Selection toolbar. Displays site and application numbers of multiple back-ends, if present.
  - Simulation Time toolbar. Displays the elapsed simulation time.
  - Simulation Time Scale toolbar. Allows you to control the speed at which the simulation runs.
- Application panels. Dockable panels that provide information and let you manage the observer.
- Convenience toolbars. Toolbars that let you quickly select frequently used menu commands and configuration settings.
- Object creation palettes. Palettes for creating simulation objects, tactical graphics, and props.

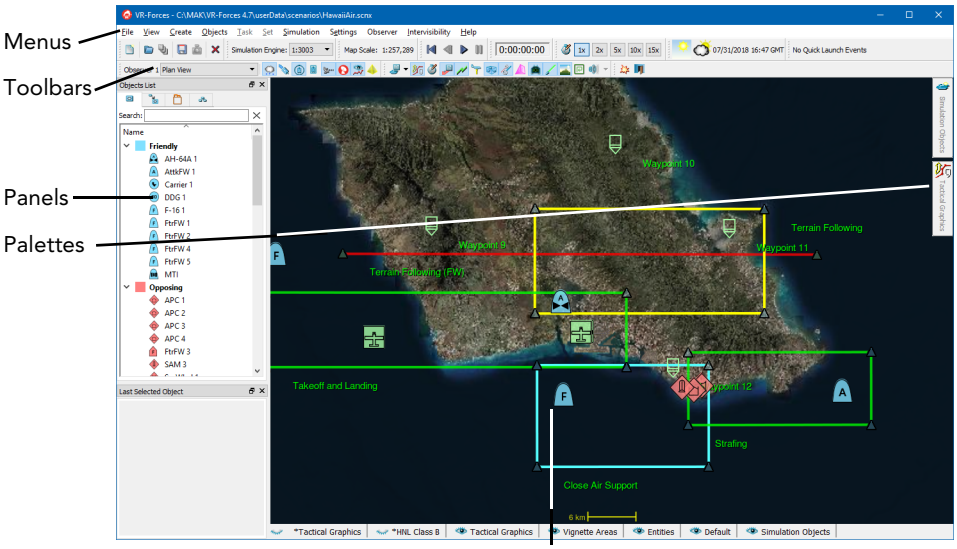


Figure 4-5. VR-Forces window (2D)

You can view the terrain in 2D (Figure 4-5), 3D (Figure 4-6), and exaggerated reality (XR) (Figure 4-7) modes.

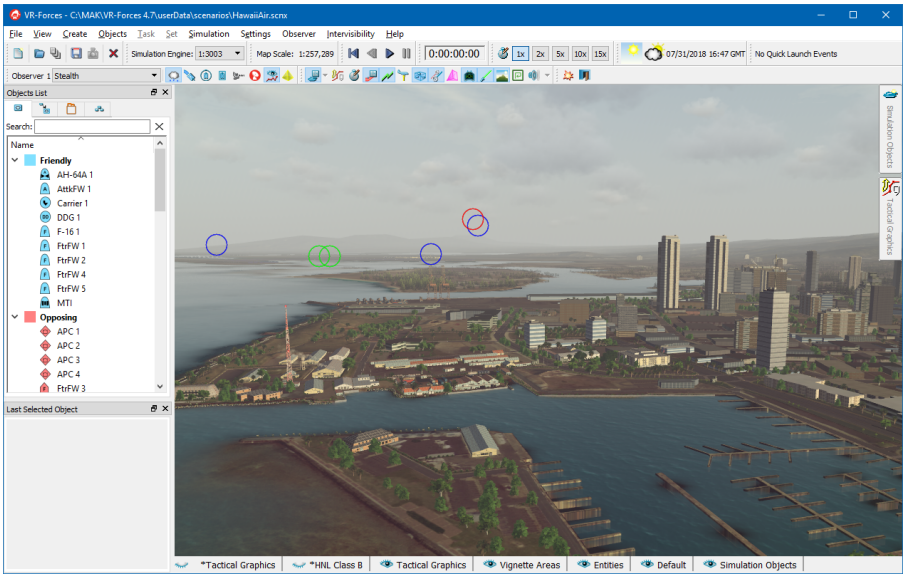


Figure 4-6. VR-Forces window (3D)

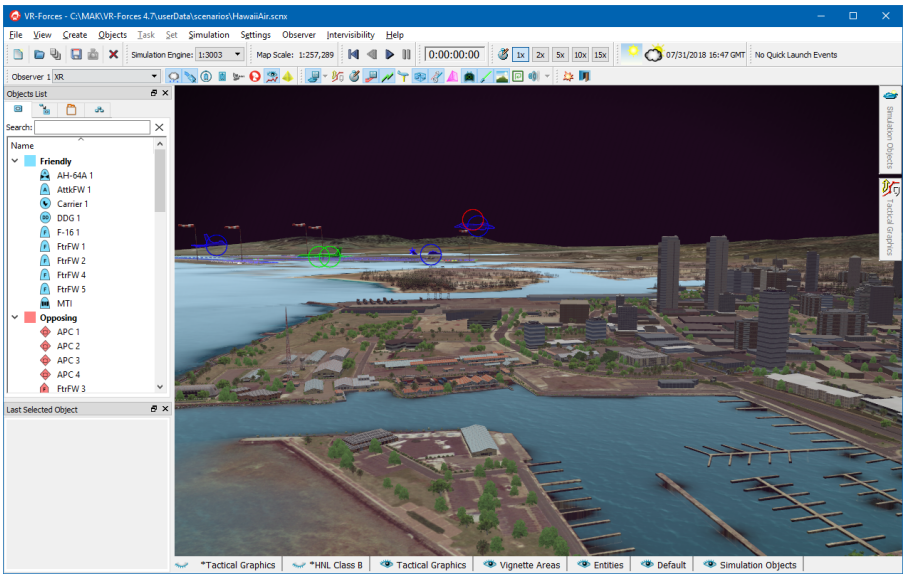


Figure 4-7. VR-Forces window (XR)

You can undock toolbars and panels and display them anyplace on your desktop. You can hide any toolbar that you do not want to be visible. (For details, please see Chapter 83, *GUI Controls, Layouts, and Behaviors*.)

4.2.1. Opening New Windows

In addition to the main VR-Forces window, you can open secondary windows. These windows do not have any of the menus or toolbars of the main window. However, you can navigate the windows using the same keyboard and mouse options as the main window. When a secondary window has focus, the settings on the Observer Settings toolbar apply to that secondary window and changes you make affect the window.

You can open windows with a 2D view or 3D view. Once a window is open, you can change the view by changing the observer mode.

Opening a secondary window is not the same thing as running a second instance of the front-end. When you close a front-end, all secondary windows associated with it are also closed.

➤ To open a new window, choose **File** → **New Stealth Window** or **File** → **New Plan View Window**.



You can also open a new window by opening an inset view. For details, please see "Inset Views," on page 53-2.

4.2.2. Printing the VR-Forces Display

You can print the VR-Forces display to any printer configured on your computer. None of the GUI controls or tabs get printed.

To print the VR-Forces display:

1. Choose **File** → **Terrain** → **Print**. The Print dialog box opens.
2. Select a printer.
3. Click Print.

4.2.3. Taking Screen Captures

You can take screen captures of the VR-Forces window. None of the GUI controls or panels get saved.

By default, screen captures are saved to *./userData/screenshots*. You can change the default location. You can change the key used for screen captures. For details about changing key mappings, please see Chapter 85, *Creating and Editing Key Mappings*.

To take a screen capture:

1. Choose **File** → **Terrain** → **Save Terrain as Image**. The Save Map Area dialog box opens.
2. Select the directory in which you want to save the image.
3. Type a name for the file.
4. Select the format you want to save it as.
5. Click Save.



You can also take a screen capture by pressing **CTRL+F5**. It is saved to the default directory with a default name.

Specifying the Location of Screen Captures

To specify the location where screen captures are saved:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the General Application Settings page (Figure 4-8).

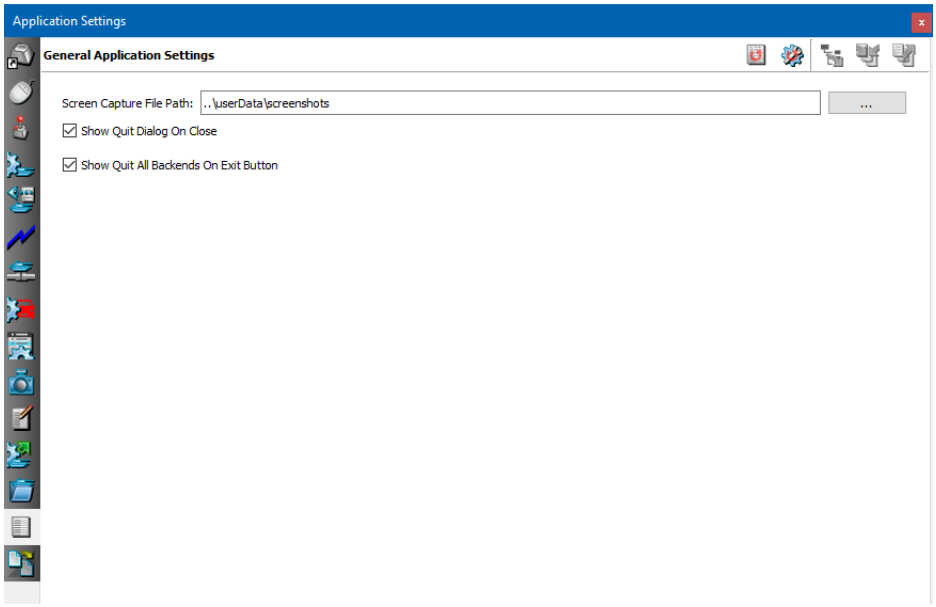


Figure 4-8. General Application Settings page

3. In the Screen Capture File Path box, specify the location where you want to save screen captures.

4.3. Managing a Front-end's Session Connection

A front-end can be joined to a session or not joined. When a front-end is not joined to a session, it cannot create, load, or manage a scenario. It can only load a terrain and function as a viewer. You can tell a front-end's session status by the availability of Join Session and Resign Session commands on the File menu.

By default, when you start a front-end it automatically tries to join a session. If there is no session running, it polls the network until one becomes available and then automatically joins it. Once a front-end is joined to a session, you can manually resign from and rejoin it.

When a front-end joins a session, its behavior depends on the session settings set in the Application Settings dialog box, Session Settings page. The options are:

- Ask to join the session.
- Always join the session with the session database.
- Always join the session with whatever database is currently loaded.

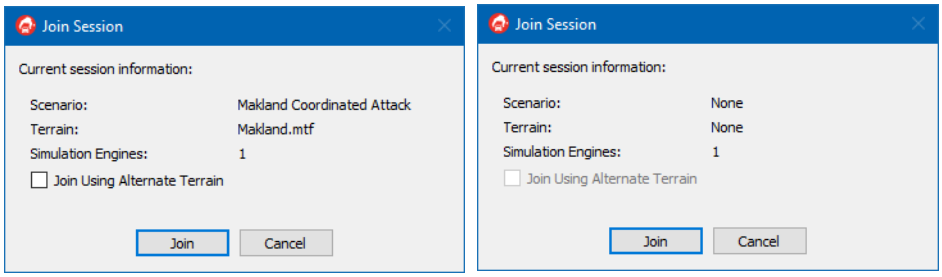
! If you do not load the terrain specified by the scenario, it is possible that visual and simulation terrains will not be synchronized.

4.3.1. Joining a Session

When you join a session, all menu commands related to scenario management are enabled. If a scenario is loaded in the session, the terrain is automatically loaded in the joining front-end.

To join a session:

1. Choose **File** → **Join Session**. If VR-Forces is configured to always join a session with the session database, the front-end loads the session database and joins the session. If VR-Forces is not configured to automatically use the session database, the Join Session dialog box opens (Figure 4-9). If a scenario is running, its details are listed.



Scenario running — No scenario running

Figure 4-9. Join Session dialog box

2. If a scenario is running, you can specify that you want to load an alternate terrain:
   a. Select the Join Using Alternate Terrain check box.
   b. Click Join. An Open Database dialog box opens.
   c. Select the terrain that you want to load.
   d. Click Open. The terrain is loaded and the front-end joins the session.

! If you join a session using an alternate terrain, the front-end (visual) and back-end (simulation) terrains might not be synchronized.

---

3. Click Join. The front-end joins the session.

4.3.2. Resigning from a Session

When you resign from a session, all menu commands related to scenario management are disabled. If you have a scenario loaded, the terrain is closed.

➤ To resign from a session, choose **File** → **Resign From Session**.

4.3.3. Configuring Session Messages and Join at Startup

You can configure the display of session-related messages and whether or not the front-end tries to join a session or open a terrain database automatically at startup.

To configure sessions:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Session Settings page (Figure 4-3).
3. Select the options that you want to enable, as follows:
   - Automatically Join Session. If selected, when the front-end starts, automatically join a session if it is available. If cleared, the front-end does not join a session and does not prompt you to do so.
   - Automatically Open Session Database without Joining Session. This option lets VR-Forces automatically load the terrain database for the current session and put you in view only mode. You cannot create objects or control them. It interacts with Automatically Join Session as shown in Table 4-1.

Table 4-1: Join session and open session database interaction

Automatically Join Session	Open Session Database without Joining	Result
☑	☐	At startup, VR-Forces joins the session, loads the terrain, and you have full control of objects and scenario actions.
☐	☑	At startup, VR-Forces loads the terrain database. You have no ability to create published objects or control the scenario. (You can create local tactical graphics.) If you clear this check box after the terrain loads, it remains loaded.
☑	☑	If you resign from a session, the terrain database stays open and you cannot create published objects or control the scenario. (You can create local tactical graphics.)
☐	☐	If, after starting VR-Forces, you select Automatically Open Session Database without Joining Session and a session is open, the terrain is loaded.

  - Join Session Options. You can select one of the following options for joining a session:
    - Ask to Join Session. Provides session data (Figure 4-9) and prompts you to approve whenever you join a session.
    - Always Join Session with Session Database. Always join a session using the database in the loaded scenario. (Does not display a Join Session dialog box.)
    - Always Join Session with Open Database. Always joins a session with whichever terrain is currently open in the front-end. This might generate a terrain correlation warning.
  - Show Session Terrain Change Prompts. Enable if you want to get prompted when the terrain in a session changes
  - Show Scenario Startup Dialog. Enable if you want to be prompted when a new session starts.

4. Click Close.

4.4. Using HLA Time Management

In HLA, use of time management allows the rate at which a federate proceeds to be controlled by another federate. Federates being controlled by other federates are said to be constrained. Federates that control other federates are time regulating. A federate can be both time regulating and constrained. It can also be unconstrained.

The main reasons why you might want to enable time management for VR-Forces are to run VR-Forces as part of a wider federation that is using time management or to synchronize multiple VR-Forces back-ends. Without time management, multiple back-ends can stay synchronized only if they all are able to maintain the same frame rate. However, if they cannot do so, they will no longer have a shared concept of simulation time and will become unsynchronized. By using time management services, multiple back-ends can be configured to run in lock-step with respect to simulation time, ensuring that events will always happen in the same order.



VR-Forces supports time management in the back-end only.

For details about enabling time management, please see "Configuring Time Management for HLA Exercises," on page 4-19.

4.4.1. VR-Forces Simulation Time Versus Federation Time

Time management synchronizes federates to federation time. Federation time is the common model of time for the federation as whole. It is the time used in time stamp ordered messages (TSO), exchanged by time-managed federates. During the lifetime of a federation, federation time always increases. (There is no notion of rewinding back to an earlier time as with VR-Forces simulation time.)

VR-Forces simulation time is not the same as HLA federation time. It is the elapsed time within a scenario (the time displayed in the Simulation Time toolbar). When you first start a simulation, federation time and simulation time may be similar. However, if you rewind a time-managed simulation, the discrepancy between the two time streams will increase. This should not have any effect on running and rewinding scenarios. As long as all the back-ends start from the same simulation state when you begin a simulation, they will be correctly synchronized (against federation time) as the simulation progresses.

4.4.2. Configuring Time Management for HLA Exercises

To use time management services in an HLA exercise, you must properly configure an RTI and enable time management in the VR-Forces back-end.

Time management is enabled at the application level, not the scenario level. Scenarios created without use of time management work in time-managed mode. No time management-specific information is saved as part of a scenario.

If you enable time management in the back-end, then whenever you create or load a scenario that specifies a fixed-frame mode (that is, the frame-mode parameter in the scenario file is set to either fixed-frame best effort or fixed-frame-run-to-complete), the back-end will become a regulating and constrained federate. It will remain in this state until the scenario is closed.

A time-managed back-end advances simulation time at a rate that is a function of the frame-mode specified in the scenario it is executing. If the frame-mode is fixed-frame, then it tries to advance simulation time at a rate approximating real time. However, if the back-end has a heavy simulation load, or another time-constrained federate is advancing at a relatively slower rate, the back-end may not achieve this rate. If the frame-mode is fixed-frame-run-to-complete, the back-end tries to advance simulation time as fast as possible, independent of wall-clock time.

RTI Requirements for Time Management

To use time management in VR-Forces, your RTI must be properly configured to support this service. If you are using the MAK RTI, set the RTI_useRtiExec and RTI_timeMgmt parameters to 1. When you start the federation, you must run the rtiexec.

If you are using another RTI, please consult its documentation for configuration instructions.

Enabling Time Management for the Back-end

To enable time management in the back-end, use one of the following methods:

- Set runInTimeManagementMode to 1 in *vrfSim.mtl*.
- Start the back-end with the `--timeManagement` command-line option. For example, to start the back-end in time management mode, the sample FED file configured for use with time management, and the federation name MAK-RPR1-TimeManagement-1-1:

```
vrfSimHLAprotocol -s 1 -a 3001 --timeManagement
   -F MAK-RPR1-TimeManagement-1-1.fed -x MAK-RPR1-TimeMan-
   agement-1-1
```



- The back-end will only run in time management mode (time regulating and constrained) while it has a scenario loaded.
- If a scenario is configured with frame-mode set to variable-frame, the back-end will not become time regulating or constrained. It will print a warning to the console and run without time management enabled.

4.5. Opening a Terrain Database

If you are using multiple front-ends, when you load or create a scenario, the front-end on which you issue the command loads the correct terrain database. The other front-ends prompt you to load the terrain. However, you can choose to load a different terrain. Also, if you want to use the front-end as a viewer, you can manually load a terrain database. (For a brief description of the databases supplied with VR-Forces, please see "Terrain Databases Provided with VR-Forces," on page 55-21.)



If you load a different terrain in the front-end than the one loaded in the back-end, there might be correlation issues between the terrains. VR-Forces can warn you about correlation issues. For details, please see "Displaying Terrain Correlation Warnings," on page 4-23.

You can only "open" terrains in MTF format. To open any other terrain format, you must compose a terrain, as described in Chapter 57, *Composing Terrains*, or use an earth file, as described in Chapter 58, *Paged and Streaming Terrains*.

The first time you load a terrain (including the terrains supplied with VR-Forces), VR-Forces caches textures in its fast load format. This is a slow process. However, thereafter the terrain will load quickly. If you want to avoid runtime caching, you can preprocess the terrain with the medfTool before you load it. (For details about the medfTool, please see "Compressing Model Files," on page 90-35.) If you want to disable runtime caching, you can do so as described in "Configuring File Caching," on page 64-2.

It is possible to open a terrain without having a scenario loaded. In such cases you can create tactical graphics and add props, but you cannot create simulation objects. To create and manipulate simulation objects, you must load or create a scenario.

To open a terrain database:

1. In the VR-Forces window, choose **File** → **Terrain** → **Open Terrain**. The Open Terrain dialog box opens.
2. Select the terrain that you want to open.
3. Click Open. The database is displayed in the VR-Forces window (Figure 4-10).

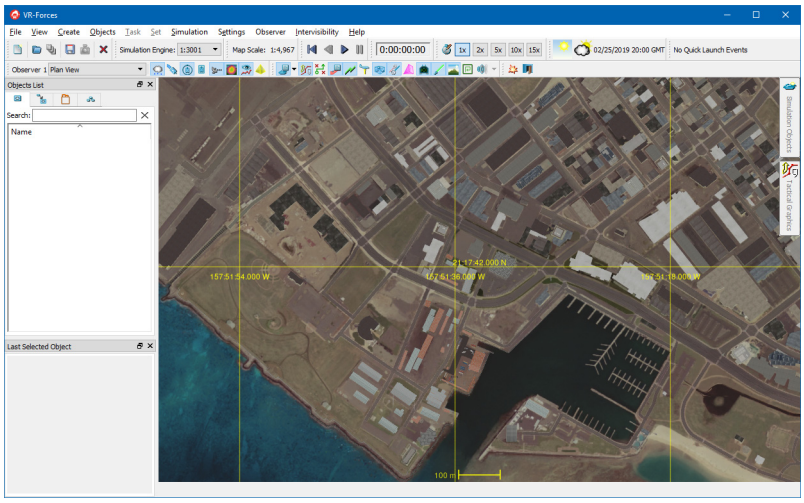


Figure 4-10. VR-Forces window with a database loaded

- If you load a terrain database while one is currently loaded, VR-Forces closes the current database and then opens the new one.
- Paged terrains might not be visible when loaded in Plan View mode. To see the terrain, press **e** to zoom in. This problem potentially exists for all paged terrains whose page-in distance is less than the default observer distance in Plan View mode.
- Some MetaFlight terrains do not display in the 2D view when first loaded. To see the terrain, change to a 3D view mode, and press the space bar (center view on terrain) to move the eyepoint to the terrain location. After this, the terrain can be viewed in the 2D view.

4.5.1. Loading a Terrain Database at Startup

You can load a terrain database at startup by specifying it on the command line. If you start VR-Forces while a session is running, it can load the appropriate terrain database automatically. For details, please see "Configuring Session Messages and Join at Startup," on page 4-17.

➤ To load a terrain database when you start VR-Forces, use the `-T` command line option.

```
vrfSimprotocol -s 1 -a 3001 -T "../userData/terrains/myDa-
   tabase.mtf"
```

where *protocol* is `DIS`, `HLA13`, `HLA1516`, or `HLA1516e`.

4.5.2. Displaying Terrain Correlation Warnings

By default, VR-Forces displays a warning message if the terrain loaded in the front-end might not correlate well with the terrain loaded in the back-end. You can enable and disable this warning message.

To enable or disable terrain correlation warning messages:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 4-3).
3. Select or clear the Show Database Correlation Warning Dialogs check box.
4. Click Close.

4.5.3. Closing a Terrain

To close a terrain, use one of the following methods:

- Choose **File** → **Terrain** → **Close Terrain**.
- Choose **File** → **Terrain** → **New Terrain**.
- Open a different terrain.

4.6. Exiting VR-Forces

When you exit VR-Forces from the GUI, all GUI windows close. In combined mode, the simulation engine that started with the GUI also closes. If you are not running in combined mode or have multiple simulation engines, you must also close all simulation engines (either manually or using the correct Exit option). The Yes options on the exit prompt work as follows:

- Yes. In independent mode, closes the GUI. In combined mode closes the GUI and the simulation engine that was started with the GUI.
- Yes, and Quit All Back-Ends. Closes the GUI and all back-ends in the session, regardless of how you started them, which computer is running them, and which operating system they are running on.

You can disable the exit prompt if you want to. In that case exiting acts as if you clicked Yes on the exit prompt. For details, please see "Disabling the Quit Prompt," on page 4-24.

To exit the VR-Forces GUI:

1. Choose **File** → **Exit**. An exit prompt opens.
2. Click the button for the exit option you want.

➤ To exit a simulation engine from the console window, type **q** and press **ENTER**.

4.6.1. Disabling the Quit Prompt

When you exit VR-Forces, you are prompted to confirm that you want to exit. You can disable this prompt.

To disable the quit prompt:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the General Application Settings page (Figure 4-8).

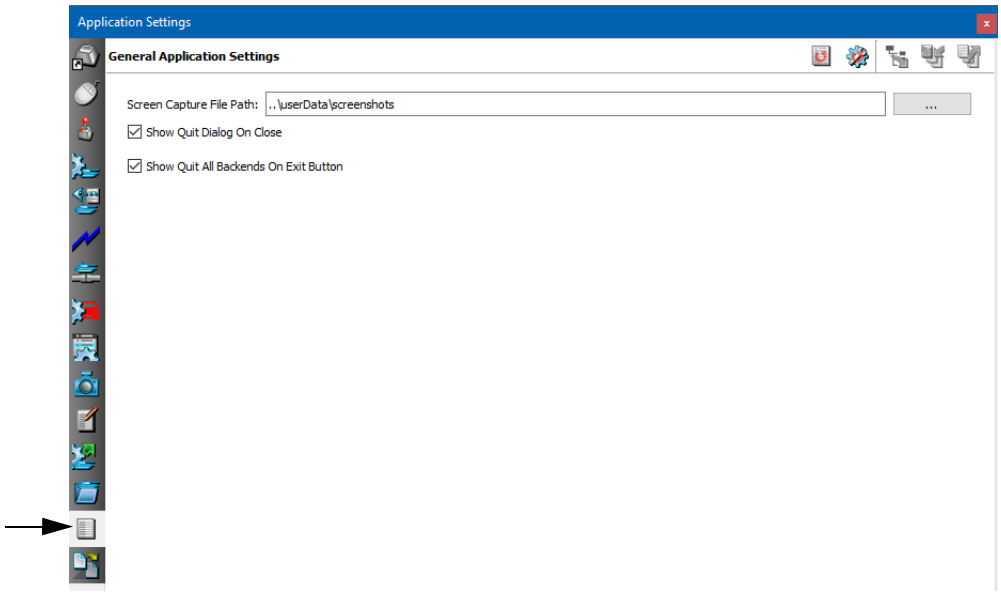


Figure 4-11. General Application Settings page

3. Clear the Show Quit Dialog On Close check box.

4.7. Configuring Simulation Connections

A simulation connection specifies the connection parameters for a DIS or HLA simulation connection. VR-Forces comes with the following connection configurations:

- DIS (7) localhost (for use when disconnected from a network or if you do not want to interact with network traffic).
- DIS (7).
- HLA 1.3 RPR FOM 2.0 with MAK extensions.
- HLA 1516 Evolved RPR FOM 2.0 with MAK extensions.

You can edit the supplied configurations or add your own connection configurations that use, for example, a different DIS port or your own FOM Mapper.

4.7.1. Opening the Simulation Connections Configuration Dialog Box

Simulation connection configurations are maintained in the Simulation Connections Configuration dialog box. This dialog box opens automatically when you start VR-Forces or run the vrfLauncher, unless you choose a configuration as the default connection for auto connection. If Auto Connect is enabled and you want to change the connection, you can open the dialog box from the Tools menu.

➤ To open the Simulation Connections Configuration dialog box from the Windows Start menu, choose **MAK Technologies** → **Configure VR-Forces Connections**.

To start the vrfLauncher on Linux (or from the Windows command line), in a console window, enter the following command:

```
vrfLauncher options
```

For descriptions of the vrfLauncher command-line options, please see "Command-Line Options for vrfLauncher," on page 5-16.

4.7.2. Adding a Simulation Connection

When you add a new simulation connection, it has the default values for the protocol that you select. You can then customize it to meet your needs. You can also add new connections by copying an existing connection.

To add a simulation connection:

1. On the Windows Start menu, choose **MAK Technologies → Configure VR-Forces Connections**. The Simulation Connections Configuration dialog box opens (Figure 4-12).

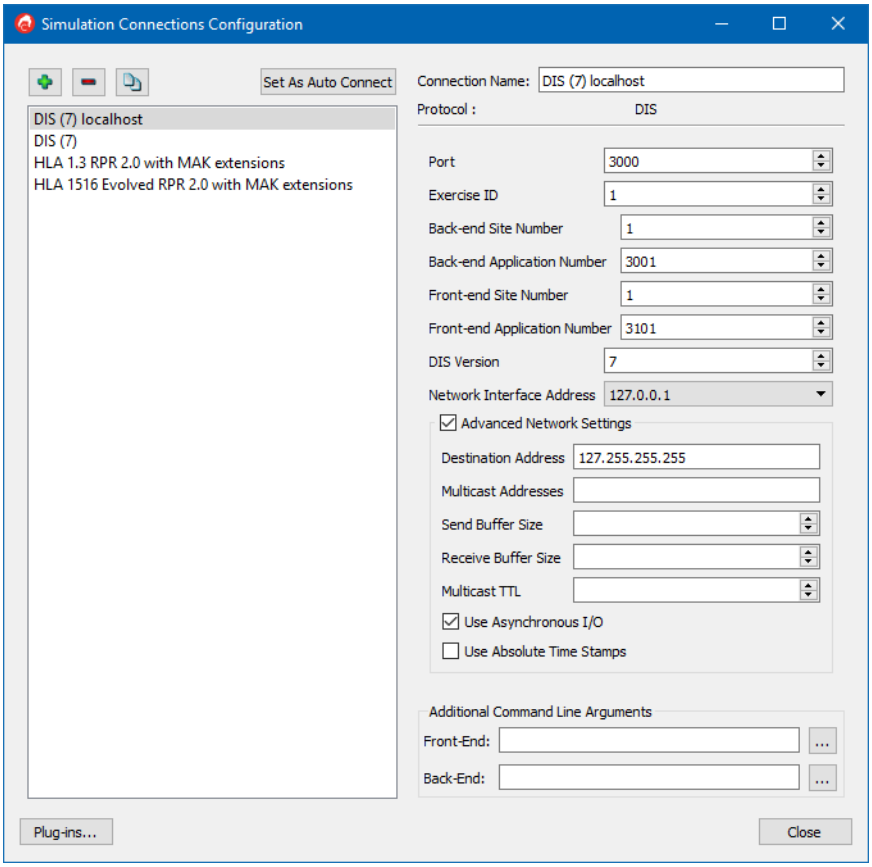


Figure 4-12. Simulation Connections Configuration dialog box

2. Click the Add Connection button (+). The Create Connection dialog box opens (Figure 4-13).

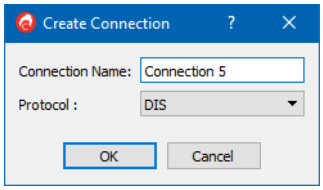


Figure 4-13. Create Connection dialog box

3. In the Connection Name box, type a name for the connection.
4. Select a protocol from the Protocol list.
5. Click OK.
6. The connection is added to the list in the Simulation Connections Configuration dialog box.
7. Select the new connection in the list.
8. Edit the parameters as desired. They are described in "Simulation Connection Parameters," on page 4-28.

Copying a Simulation Connection

You can create a new simulation connection by copying an existing connection.

To copy a simulation connection:

1. On the Windows Start menu, choose **MAK Technologies → Configure VR-Forces Connections**. The Simulation Connections Configuration dialog box opens (Figure 4-12).
2. Click the Copy Connection button (). The Copy Connection dialog box opens.
3. Accept the default name or type a new name.
4. Click OK.
5. Edit the parameters as desired. They are described in "Simulation Connection Parameters," on page 4-28.

4.7.3. Editing a Simulation Connection

To edit a simulation connection:

1. On the Windows Start menu, choose **MAK Technologies → Configure VR-Forces Connections**. The Simulation Connections Configuration dialog box opens (Figure 4-12).
2. In the list of connections, select the connection that you want to edit.
3. Edit the parameters as desired. They are described in "Simulation Connection Parameters," on page 4-28.

4.7.4. Simulation Connection Parameters

The HLA and DIS simulation connection parameters are equivalent to command-line options for DIS and HLA. Table 4-2 describes the connection parameters for DIS and cross references the parameters to their descriptions in Chapter 5, *Command-Line Options*. Table 4-3 describes the connection parameters for DIS and cross references the parameters to their descriptions in Chapter 5, *Command-Line Options*.

Table 4-2: Connection configuration parameters for DIS

Parameter	Command-line equivalent	Description
Port	`-P`	Specifies the DIS port. "Specifying the Port Number," on page 5-21.
Exercise ID	`-x`	Specifies the DIS exercise ID. "Specifying the Exercise ID," on page 5-22.
Back-end Site Number Front-end Site Number	`-s`	Specifies the site ID. These values can be the same or different. It is entirely arbitrary. "Specifying the Site ID and Application Number," on page 5-18.
Back-end Application Number Front-end Application Number	`-a`	Specifies the application number. The default front-end application number is created by adding 100 to the back-end application number. "Specifying the Site ID and Application Number," on page 5-18.
DIS Version	`--disVersion`	Specifies the DIS version: 4, 5, 6 or 7. Default: 7. "Specifying the DIS Version," on page 5-22.
Network Interface Address	`--deviceAddress`	Specifies the address of the network card to use for UDP traffic.
Destination Address	`-A`	Specifies a specific destination address (point-to-point). "Specifying Point-to-Point or Multicast Operation," on page 5-22.
Multicast Addresses	`-S`	Specifies one or more multicast addresses. "Specifying Point-to-Point or Multicast Operation," on page 5-22.
Send Buffer Size	`--sendBufferSize`	Specifies the send buffer size. Default: -1.
Receive Buffer Size	`--recvBufferSize`	Specifies the receive buffer size.
Multicast TTL	`--mcastTtl`	Specifies the multicast time-to-live. Default: -1.
Use Asynchronous I/O	`--useAsyncIO`	Use asynchronous I/O. Default: False.

Table 4-2: Connection configuration parameters for DIS

Parameter	Command-line equivalent	Description
Use Absolute Time Stamps	`--useAbsolute-TimeStamps`	Specifies use of absolute time stamps instead of relative time stamps.

> **!** If you select Use Absolute Time Stamps, it is your responsibility to ensure that the time clocks of exercise participants are synchronized. If they are not synchronized, VR-Forces may not work properly.

Table 4-3: Connection configuration parameters for HLA

Parameter	Command-line equivalent	Description
Network Interface Address	`--deviceAddress`	The IP address of the host computer.
Federation Name	`-x`	Specifies the name of the federation execution. "Specifying the Federation Execution," on page 5-19.
FED File Name	`-F`	Specifies the FED file to use. "Specifying the FED File," on page 5-19.
Back-end Site Number Front-end Site Number	`-s`	Specifies the site ID. These values can be the same or different. It is entirely arbitrary. "Specifying the Site ID and Application Number," on page 5-18.
Back-end Application Number Front-end Application Number	`-a`	Specifies the application number. The default front-end application number is created by adding 100 to the back-end application number. "Specifying the Site ID and Application Number," on page 5-18.
RPR FOM Version	`--rprFomVersion`	"Specifying the RPR FOM Version," on page 5-20.
FOM Mapper Library	`-f`	Specifies the VR-Link FOM Mapper to use. "Specifying a FOM Mapper," on page 5-20.
Initialization String	`--fomMapperInit-Data`	Specifies initialization data required by the FOM Mapper, if any. "Specifying FOM Mapper Initialization Data," on page 5-20.

Table 4-3: Connection configuration parameters for HLA

Parameter	Command-line equivalent	Description
Local Settings Designator	`-s`	Specifies a string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage.
FOM Modules	`--fomModules`	Specifies FOM Modules. (HLA Evolved only.)
Ignore Advisories	`--useAdvisories`	Specifies whether or not to use HLA advisories.
! If you are using a lightweight RTI connection on Linux, you must enable Ignore Advisories or the simulation engine will not publish entities.		
Use Absolute Time Stamps	`--useAbsolute-TimeStamps`	Specifies use of absolute time stamps instead of relative time stamps.
! If you select Use Absolute Time Stamps, it is your responsibility to ensure that the time clocks of exercise participants are synchronized. If they are not synchronized, VR-Forces may not work properly.		

4.7.5. Deleting a Simulation Connection

To delete a simulation connection:

1. Choose **MAK Technologies** → **Configure VR-Forces Connections**. The Simulation Connections Configuration dialog box opens (Figure 4-12).
2. In the list of connections, select the connection that you want to delete.
3. Click the Delete button (−). A confirmation prompt opens.
4. Click Yes.

4.7.6. Configuring Auto Connect

If you know that you will usually or always want to use the same simulation connection, you can set up Auto Connect and you will not have to select a connection every time you start VR-Forces.

To configure Auto Connect:

1. Choose **MAK Technologies** → **Configure VR-Forces Connections**. The Simulation Connections Configuration dialog box opens (Figure 4-12).
2. In the list of connections, select the connection that you want to use every time you start VR-Forces.
3. Click Set as Auto Connect. A star is displayed next to the selected connection.

Disabling Auto Connect

To disable Auto Connect:

1. Choose **MAK Technologies** → **Configure VR-Forces Connections**. The Simulation Connections Configuration dialog box opens (Figure 4-12).
2. In the list of connections, select the connection that is configured for auto connect.
3. Click Set as Auto Connect. Auto connect is disabled and the star is removed from next to the connection.

4.7.7. Displaying Connection Information

You can display connection information while you are running VR-Forces. This may be useful for diagnostic purposes.

➤ To view connection information, choose **Settings** → **Connection Information**. The Connection Information dialog box opens (Figure 4-14).

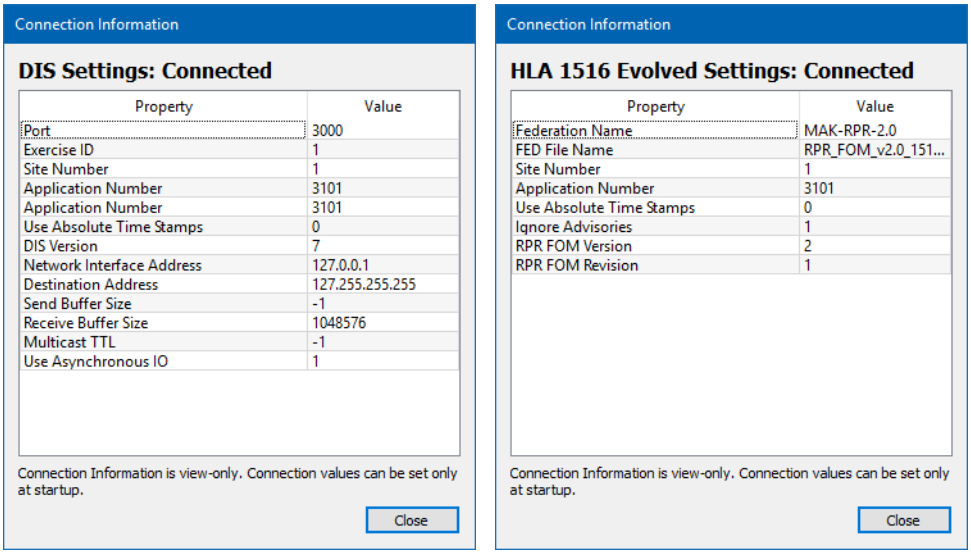


Figure 4-14. Connection Information dialog box

4.8. Managing Plug-ins

One way to extend the functionality of VR-Forces is by loading plug-ins that have been written for it by MAK or other software vendors. You can manage which plug-ins get loaded when VR-Forces starts up. You can also view information about the plug-ins that have been loaded.

Plug-ins are implemented as one or more dynamic linked libraries (DLL or SO files). The DLLs can be release or debug, for vrfSim or vrfGui, for Windows or Linux, and for a specific protocol or protocol-independent. Each association of an application, platform, build type, and transport type is called a configuration. The Plug-ins Editor lets you specify the DLLs to use for each of the desired configurations.



Changes to plug-in configurations do not take effect until the next time you start the affected VR-Forces application.

4.8.1. Loading Plug-ins

You can specify which plug-ins to load in either of the following ways:

- In the Plug-ins dialog box, specify the plug-ins to load the next time VR-Forces starts.
- In the Simulation Connections Configuration dialog box, specify the plug-ins to load at startup.

Specifying the Plug-ins to Load in the Plug-ins Dialog Box

The Plug-ins dialog box lets you specify the plug-ins that will be loaded the next time that you start VR-Forces.

To specify that a plug-in be loaded or not loaded in the Plug-ins dialog box:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 4-15).

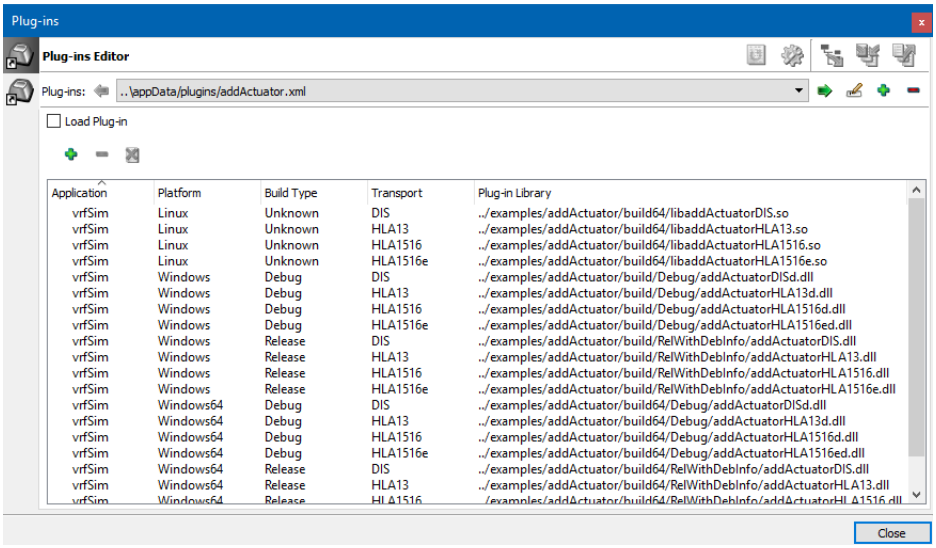


Figure 4-15. Plug-ins dialog box, Plug-ins Editor page

3. In the list of plug-ins at the top of the page, select the one that you want to edit. The configurations for that plug-in are listed in the window.
4. Select or clear the Load Plug-in check box.

Specifying the Plug-ins to Load in the Simulation Connections Configuration

You can select the plug-ins that you want to load at startup.

To specify the plug-ins to load in the simulation connections configuration:

1. Start VR-Forces using the VR-Forces Launcher, as described in "Starting VR-Forces from the VR-Forces Launcher," on page 4-4. The Simulation Connections Configuration dialog box opens (Figure 4-1).
2. Click Plug-ins. The Plug-ins Selection dialog box opens (Figure 4-16). The dialog box lists all of the plug-ins that are configured for VR-Forces. If a plug-in has previously been selected for loading, in this dialog box or in the Plug-ins Editor, it is checked.

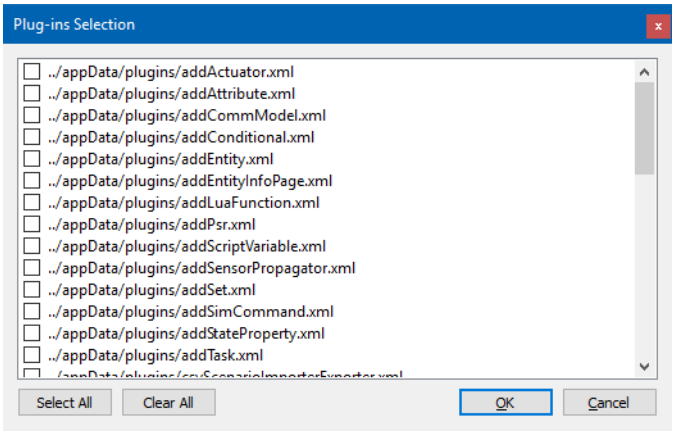


Figure 4-16. Plug-ins Selection dialog box

3. Select the check boxes for the plug-ins that you want to load. Clear the check boxes for any plug-ins that you do not want to load.
4. Click OK. The plug-in configuration applies to all of the startup configurations.

4.8.2. Adding a Plug-in

When you add a plug-in, VR-Forces creates an XML file (in *./appData/plugins*) that contains all of the information VR-Forces needs to load the plug-in. New plug-ins are added with entries for each of the configurations (application, platform, build type, transport). After you add a plug-in, you must specify the DLLs for each configuration that you want to support.

To add a plug-in:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens (Figure 4-15).
2. Select the Plug-ins Editor page.
3. Click the Add button (+) at the upper right corner of the page (#1 in Figure 4-17). The New Plug-in dialog box opens.
4. Type a name for the plug-in.
5. Click OK. The plug-in is added to the Plug-ins list. It has configurations for each of the combinations of application, platform, build type, and transport (Figure 4-17).

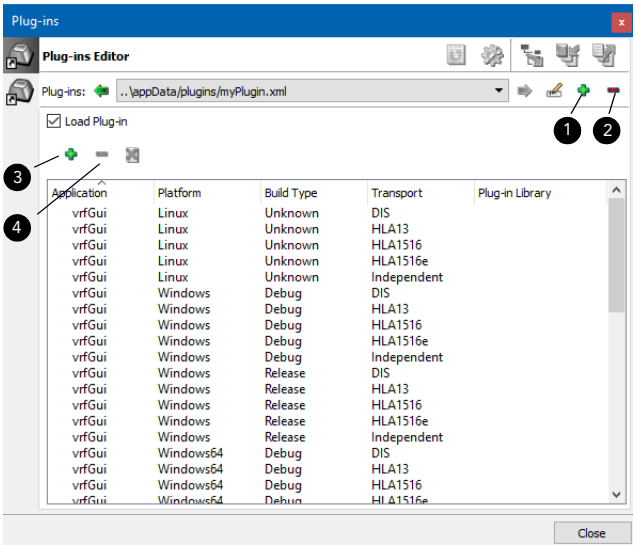


Figure 4-17. New plug-in

6. Specify the DLL for each configuration that you want to support, as described in "Specifying the DLLs for a Plug-in," on page 4-37.

4.8.3. Specifying the DLLs for a Plug-in

Each plug-in configuration that you want to be able to load must specify the DLL to load for that configuration.

To specify the DLL for a plug-in configuration:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens (Figure 4-15).
2. Select the Plug-ins Editor page (Figure 4-18).

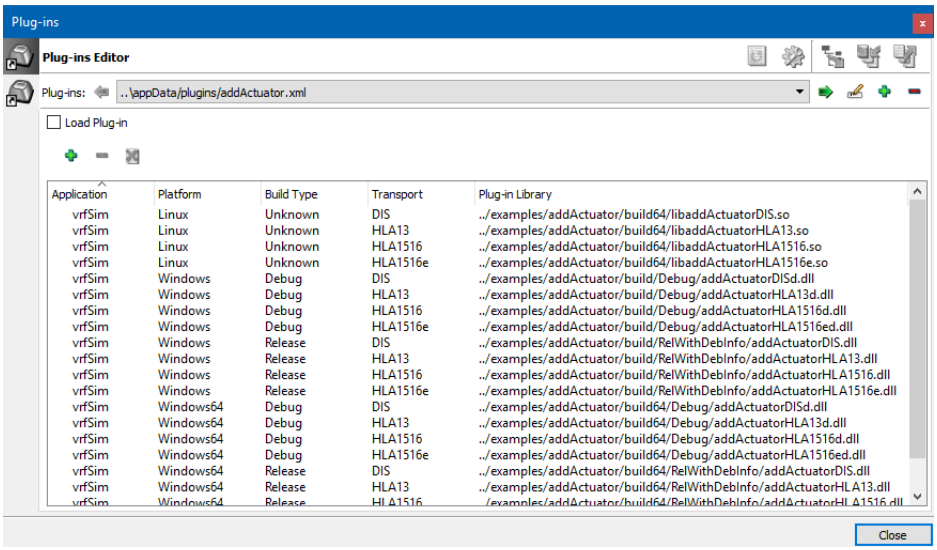


Figure 4-18. Plug-ins Editor

3. In the list of plug-ins at the top of the page, select the one that you want to edit.
4. Click the Plug-in Library column for the configuration that you want to edit. The Select Plugin dialog box opens.
5. Select the DLL that you want to load for this configuration.
6. Click Open. The value is changed.

Removing a Plug-in's DLL

You can remove the DLL assignment for a plug-in configuration without removing the configuration.

To remove the DLL assignment for a plug-in:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens (Figure 4-15).
2. Select the Plug-ins Editor page.
3. Select the configuration whose DLL assignment you want to remove.
4. Click the Erase button ( ).

4.8.4. Adding a Plug-in Configuration

To add a plug-in configuration:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens (Figure 4-15).
2. Select the Plug-ins Editor page.
3. In the list of plug-ins at the top of the page, select the one that you want to edit.
4. Click the Add button (+) above the list of configurations (#3 in Figure 4-17). The Add Plug-in dialog box opens (Figure 4-19).

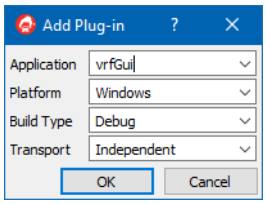


Figure 4-19. Add Plug-in dialog box

5. For each parameter (Application, Platform, Build Type, and Transport), select a value from the list.
6. Click OK. The configuration is added to the list.
7. Specify a DLL for the configuration, as described in "Specifying the DLLs for a Plug-in," on page 4-37.

4.8.5. Deleting a Plug-in Configuration

To delete a plug-in configuration:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens (Figure 4-15).
2. Select the Plug-ins Editor page.
3. In the list of plug-ins at the top of the page, select the one that you want to edit.
4. Select the configuration that you want to delete.
5. Click the Delete button (−) that is above the list of configurations (#4 in Figure 4-17).

4.8.6. Deleting a Plug-in

When you delete a plug-in, you delete the XML configuration file in *./appData/plugins* that specifies the plug-in configurations and whether or not to load it.

To delete a plug-in:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens (Figure 4-15).
2. Select the Plug-ins Editor page.
3. In the list of plug-ins at the top of the page, select the one that you want to delete.
4. Click the Delete button (−) to the right of the plug-ins list (#2 in Figure 4-17).

4.8.7. Viewing a List of Loaded Plug-ins

You can view a list of the plug-ins that are loaded when you start up VR-Forces. Plug-ins may include a description that you can also view.

To view the details of loaded plug-ins:

1. Choose **Settings** → **Plug-ins**. The Plug-ins dialog box opens (Figure 4-20).
2. Select the Plug-ins Loaded page (Figure 4-20).

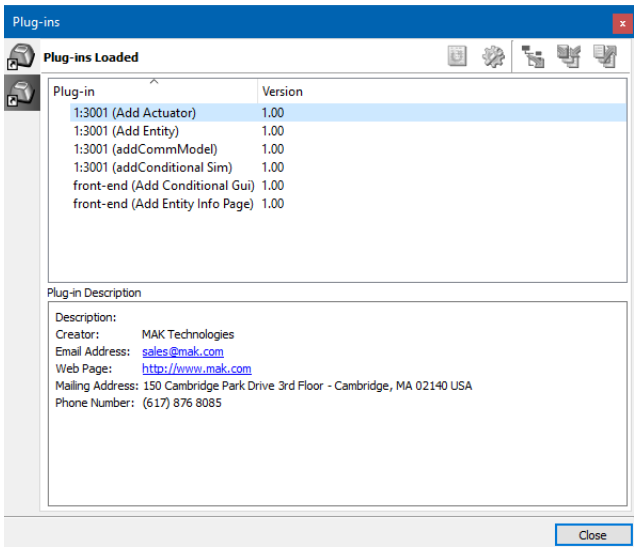


Figure 4-20. Plug-ins dialog box, Plug-ins Loaded page

3. To view details about a plug-in, select it in the list.

4.9. The VR-Forces Log Files

On Windows, VR-Forces creates two log files, *vrfSim.log* and *vrfGui.log*. Both log files are written to the directory from which you run VR-Forces.

On Linux, in combined mode, VR-Forces automatically creates a log file, *vrfSim.log* for back-end output. Front-end output is sent to the console. If you want to view back-end output, open a second console window, and do `tail -f vrfSim.log`. If you want to send front-end output to log files, or if you are running in separate mode and want to create a log file for either the front-end or back-end, you need to redirect your output to a file, for example:

```
vrfGui > vrfGui.log
```



5. Command-Line Options

VR-Forces has command-line options for vrfGui and vrfSim.

Command-Line Options for vrfGui 5-2
Command-Line Options for vrfSim 5-9
Command-Line Options for vrfLauncher 5-16
Running in Combined Mode from the Command Line 5-17
Using the -- Command-line Argument 5-17
Protocol-Independent Command-line Options 5-17
Specifying the Site ID and Application Number 5-18
Specifying the Language to Use in the GUI 5-18
Setting the Notification Level 5-18
Loading Plug-ins 5-19
Command-Line Options for HLA Federations 5-19
Specifying the Federation Execution 5-19
Specifying the FED File 5-19
Specifying the RPR FOM Version 5-20
Specifying a FOM Mapper 5-20
Specifying the RPR FOM Revision 5-20
Specifying FOM Mapper Initialization Data 5-20
Specifying a FED File That is Appropriate for the FOM Mapper 5-21
Enabling Time Management 5-21
Command-Line Options for DIS Exercises 5-21
Specifying the Port Number 5-21
Specifying Point-to-Point or Multicast Operation 5-22
Specifying the Multicast Time-to-Live 5-22
Using Asynchronous I/O 5-22
Specifying the Exercise ID 5-22
Specifying the DIS Version 5-22

5.1. Command-Line Options for vrfGui

Table 5-1 lists the command line options for vrfGui.

Table 5-1: vrfGui command-line options (Sheet 1 of 8)

Option	Description
`(-- | --ignore_rest)`	Ignore all command-line options after this one. This option is useful for disabling part of a command line in a script for test purposes so that you do not have to retype all of the command options at a later point.
`--alphaBits` *bits*	Sets the number of alpha bits (0, 8).
`--antiAliasing` *level*	Set the anti-aliasing level (0, 2, 4, 8, or 16). Default: 4. Depending on your graphics card and driver, some combinations of anti-aliasing and bit settings (`--depthBits`, `--stencilBits`, and `--alphaBits`) may be incompatible. For example, if you set stencil bits to 8, the depth must be set to 24 and anti-aliasing may need to be set to 0 or 2.
`--appDataDir` *directory*	Specifies the location of application data.
`(-c | --showConsole)`	Display a console window.
`(-C | --autoConnect)` *{filename | connector_name}*	Automatically connect to the network using the specified connector.
`--dataDir` *directory*	Specifies the location of the data directory (terrains and so on).
`--defaultSimulationModelSet` *sms*	The simulation model set to use as the default when starting up the GUI.
`--depthBits` *bits*	Sets the depth of graphics to 24 bits or 32 bits. Default: 24.
`--dis`	Start in DIS mode, using DIS command-line arguments to specify connection parameters. DIS-specific arguments do not have to follow `--dis`; they can be in any order.
`--disableDynamicTerrain`	Disables the dynamic terrain engine. May slightly speed up terrain loading and paging.
`--disableOcclusionCulling`	Disables occlusion culling. This may improve performance.
`--discoveredModelPreloadFileName` *filename*	Specifies a file to which the list of models discovered during a session that were not preloaded is written.
`--dispSetting` *filename*	Specifies the display engine configuration to load.

Table 5-1: vrfGui command-line options (Sheet 2 of 8)

Option	Description
`--doNotCheckPluginVersions`	Do not check the version in a plug-in to make sure it can load in VR-Vantage. A plug-in gets built against a particular version of VR-Vantage. Ordinarily, when VR-Vantage loads a plug-in, it checks to be sure it was created with that version of VR-Vantage. This option prevents that check.
`--doNotLoadVrfPlugins`	Prevents loading of plug-ins in the *./plugins* directory.
`(-e | --extraArgs)` *`string`* `|` `"`*`string1 ... stringn`*`"`	Allows you to send arbitrary strings to the initializer, which can be queried for by plug-ins. This option works around the restriction on adding new command-line arguments. It can take one argument or multiple arguments. Multiple arguments must be wrapped in double quotes.
`--enableLogFileTimestamps`	Print a timestamp for each line in a log file.
`--enableVSync`	Turns on VSync.
`--entDispSetting` *`filename`*	Specifies the entity display setting to load.
`--entityDefsDir` *`directory`*	Loads the *.hier* and *.leaf* files in the specified directory.
`--envSetting` *`filename`*	Specifies the environment configuration file to load.
`(-E | --entTypeMap)` *`filename`*	Load the specified object type mapping file. The file extension determines the type of mapping.
`--factoryRootDir` *`directory`*	Set the root directory used to restore settings to factory defaults.
`--fileTransporterReceivePort` *`port`*	Port for the file transporter. This is the port used to receive order of battle and plan files.
`--forceTextShaping`	Force all text to use HarfBuzz shaping. Use this option if you need support for right-to-left languages. It may affect performance.
`--forceVBO`	Force all draw commands to use vertex buffer objects (VBOs). Results in slower loading.
`--fowPerspective` *`force`*	Shows the ground truth and spot reports perspective of the specified force, where *force* can be Friendly, Opposing, Neutral, or None. Default: None.
`--fullScreen`	Start with the window in full screen mode.
`--frameRate` *`rate`*	If rate is 0, the frame rate is variable. Otherwise, specifies a fixed frame rate for rendering.
`(-G | --locale)` *`language`*	Specifies the locale (for localization).

Table 5-1: vrfGui command-line options (Sheet 3 of 8)

Option	Description
`--gui` *filename*	Specify a GUI configuration file. This file specifies the layout of menus, toolbars, and dialog box pages. A simple filename assumes *./appdata/settings/vrfGui*. A relative filename is from the *./bin*64 directory. A full path is used as is.
`(-h | --help)`	Display command-line usage information.
`--highestPriority`	Changes the priority of the application and the main render thread to the highest available priority. This command may starve other processes.
`--hla13`	Start in HLA 1.3 mode, using HLA command-line arguments to specify connection parameters. HLA-specific arguments do not have to follow `--hla13`; they can be in any order.
`--hla1516`	Start in HLA 1516 mode, using HLA command-line arguments to specify connection parameters. HLA-specific arguments do not have to follow `--hla1516`; they can be in any order.
`--hla1516e`	Start in HLA Evolved mode, using HLA command-line arguments to specify connection parameters. HLA-specific arguments do not have to follow `--hla1516e`; they can be in any order.
`(-I | --unbatchedRendering)`	Disables instance rendering. Default: Enabled.
`(-K | --disableKDTrees)`	Disable creation of KD trees for intersections.
`(-l | --logFileName)` *filename*	Specifies the log file name.
`(-L | --loadObservers)` *filename*	Loads the saved observers in the specified file. You can use this command multiple times, for example: `--loadObservers file1 --loadObservers file2`
`--layoutSettingsFile` *filename*	Specifies the GUI layout settings file to use. *filename* specifies the base filename. It is prefixed with *default_* and the extension *.uisx* is added. Default: UILayoutSettings.
`--loadPlugin` *filename*	Specifies a plug-in to load. Accepted multiple times. The filename can be a DLL or SO or an XML file that specifies the plug-ins to load. The order in which you list multiple plug-ins on the command line does not determine the order in which they are loaded. For details, please see "Managing Plug-ins," on page 4-35.
`--loadSimulationObjectGroup` *group*	Creates a new scenario on the simulation object group editing terrain and creates an instance of the simulation object group on that terrain.

Table 5-1: vrfGui command-line options (Sheet 4 of 8)

Option	Description
`--mergeEntityTypeMap {directory | filename}`	Merges all entity type mapping files (**.metx*) in a directory or individual entity type mapping files. Can be used multiple times.
`--mergeHierarchyEntityDef filename`	Merges the specified *.hier* file into the default hierarchy (usually loaded from *./appData/definitions*.) Merging is additive only. No nodes in a previously loaded hierarchy are replaced.
`--mergeHierarchyTacticalGraphicsDef filename`	Merges the specified *.hier* file into the default hierarchy (usually loaded from *./appData/definitions*.) Merging is additive only. No nodes in a previously loaded hierarchy are replaced.
`--mergeHierarchyUnitDef filename`	Merges the specified *.hier* file into the default hierarchy (usually loaded from *./appData/definitions*.) Merging is additive only. No nodes in a previously loaded hierarchy are replaced.
`--mergeLeafEntityDef filename`	Merges the specified leaf file into the default entity element definitions.
`--mergeLeafTacticalGraphicsDef filename`	Merges the specified leaf file into the default tactical graphics element definitions.
`--mergeLeafUnitDef filename`	Merges the specified leaf file into the default unit element definitions.
`--mergeModelDefs {directory | filename}`	Merges all model definition files (**.ommx*) in the specified directory or individual model definition files. Can be used multiple times.
`--mergeTacticalGraphicsTypeMap {directory | filename}`	Merges all tactical graphics type mapping files (**.mtgx*) in a directory or individual tactical graphics type mapping files. Can be used multiple times.
`--mergeUnitTypeMap {directory | filename}`	Merges all unit type mapping files (**.magx*) in a directory or individual unit type mapping files. Can be used multiple times.
`--modelPreloadFileName filename`	Specifies a file that has a list of models to preload.
`(-n | --notifyLevel) notify_level`	Specifies the notification level for application messages, where *notify_level* is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2. For details, please see "Setting the Notification Level," on page 5-18.
`--noAppDataEntityDefs`	Do not load the entity element definitions in *./appData/definitions*.

Table 5-1: vrfGui command-line options (Sheet 5 of 8)

Option	Description
`--noAppDataEntityTypeMap`	Do not load entity type mappings in *./appData/definitions*.
`--noAppDataModelDefs`	Do not load model definitions in *./appData/definitions*.
`--noAppDataTacticalGraphicsDefs`	Do not load the tactical graphics element definitions in *./appData/definitions*.
`--noAppDataTacticalGraphicsTypeMap`	Do not load tactical graphics type mappings in *./appData/definitions*.
`--noAppDataUnitDefs`	Do not load the unit element definitions in *./appData/definitions*.
`--noAppDataUnitTypeMap`	Do not load unit type mappings in *./appData/definitions*.
`--nonVrfObjectsUUIDScheme` *scheme*	Sets the UUID scheme for non VR-Forces objects. Values are entity-identifier, global-identifier, marking-text. Default: entity-identifier.
`(-O | --hasNTPSync)`	Enables synchronized ocean between different VR-Vantage applications, such as VR-Vantage Stealth and the VR-Forces front-end. System clocks must be synchronized.
`--plugin` *filename*	Load the specified plug-in. Can be used multiple times.
`--pseudoFullScreen`	Use pseudo full screen for full screen mode. This may improve performance when using dynamic ocean.
`--pvd`	Start the VR-Vantage front-end with only 2D modes available.
`--reportedHostAddressString` *address*	Specifies how VR-Forces reports its IP address to another VR-Forces application for the purpose of file transfers. It is only important when you have a GUI or remote controller on one side of a router and a simulation engine on the other, and there is some kind of network address translation occurring whereby they have to connect to one another using an IP address other than the one they recognize themselves at locally.

Table 5-1: vrfGui command-line options (Sheet 6 of 8)

Option	Description
`(-S | --suppressHatIntersectOnAttach)`	Suppress the periodic HAT intersection when attached. This setting can improve performance if you need to attach the observer to ground vehicles a lot, but do not need to attach to air vehicles.
`--scenarioFile` *`filename`*	Specifies a scenario to load.
`--stealth`	Start the VR-Vantage front-end with only 3D modes available.
`--stencilBits` *`bits`*	Sets the number of stencil bits (0 or 8). Default: 0.
`terrain_MTF_file`	Specifies a terrain (MTF) file to load. The path must be a relative path (from *./bin64*) to the file, for example, *../userData/terrains/MTF_file*.
`--unplug` *`filename`*	Do not load the specified plug-in. Can be used multiple times.
`--useAbsoluteTimeStamps`	Enables use of absolute time stamps. Otherwise uses relative time stamps.
! If you select `--useAbsoluteTimeStamps`, it is your responsibility to ensure that the time clocks of exercise participants are synchronized. If they are not synchronized, VR-Vantage may not work properly.	
`--useFileCommChannel` *`channel`*	The communication file to be used as a base for communicating text based commands from external applications.
`--userDataDir` *`directory`*	Specifies the user data directory.
`(-v | --version)`	Display version information and exit.
`(-z | --useReverseZBuffer)`	Specifies use of reverse Z buffering, which may reduce Z-fighting when viewing the terrain from a distance. (3D observer modes only.)
`(-Z | --ignoreZoomFor SpeedtreeLod`	Disable zoom for SpeedTree LOD.
HLA Only	These options are available if you specify --hla13, --hla1516, or --hla1516e.
`(-a | --appNumber)` *`number`*	Application number.
`(-f | --fomMapperLib)` *`libname`*	FOM Mapper library name.
`(-F | --fedFileName)` *`fedfile_name`*	FED file name.
`--fomMapperInitData` *`data`*	FOM Mapper initialization data.
`--fomModules` *`module`*	Specifies an HLA Evolved FOM module. Can be used multiple times.

Table 5-1: vrfGui command-line options (Sheet 7 of 8)

Option	Description
(-H | --hostAddressString) *address*	Specifies the host address. This is usually the same as the device address.
(-i | --sessionId) *ID*	Specifies the session ID for this instance of the front-end.
--ignoreAdvisories	Enables or disables HLA advisory messages. Default: true.
--localSettingsDesignator *designator*	Specifies a string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage.
(-N | --federateName) *federate_name*	Specifies a unique name that can be assigned to the federate. No two federates within the same federation can have the same name. If no name is specified, the RTI will automatically assign a unique name for the federate. Default: VR-Forces. (HLA Evolved only)
(-p | --federateType) *type*	Federate Type can be used to distinguish different categories of federates, and its only practical use is in save-and-restore.
--rprFomRevision	RPR FOM revision for RPR FOM 2.0, draft 17.
--rprFomVersion *version_number*	RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0)
(-s | --siteId *ID*	Site ID.
--useGeographicDdm	Specifies use of geographic DDM if you are using the MAK RTI. Default: false.
(-x | --execName) *exec_name*	Execution name. Default: MAK-RPR-2.0.
DIS Only	These options are available if you specify --dis.
(-a | --appNumber) *number*	Application number.
(-A | --destAddrString) *address*	Destination address.
--appIdRange	The application ID range for VR-Forces back-ends.
--defaultObjectTimeout *timeout*	Specifies the timeout interval for objects.
--deviceAddress *address*	Specifies the address of the network card to use for UDP traffic.
--disVersion	The DIS version. Default: 7.
--fileTransporterReceivePort *port*	The port for the file transporter.

Table 5-1: vrfGui command-line options (Sheet 8 of 8)

Option	Description
(-H | --hostAddressString) *address*	Specifies the host address. This is usually the same as the device address.
(-i | --sessionId) *ID*	Specifies the session ID for this instance of the front-end.
--mcastTtl	Specifies the multicast time-to-live. Default: -1.
--multicastAddresses *address1 ... addressn*	Specifies one or more multicast addresses.
(-P | --disPort) *port*	DIS port.
--recvBufferSize *size*	Receive buffer size.
(-s | --siteId *ID*	Site ID.
--sendBufferSize *size*	Send buffer size. Default: -1.
--subnetMask *mask*	Specifies a subnet mask.
--useAsyncIO	Use asynchronous I/O. Default: False.
--useIpv6	Use the IPV6 protocol, if available.
(-x | --exerciseId) *ID*	Exercise ID.

5.2. Command-Line Options for vrfSim

Table 5-2 describes the command-line options for vrfSim.

Table 5-2: vrfSim command-line options (Sheet 1 of 7)

Option	Description
(-a | --appNumber) *application_number*	Specifies the application number. For details, please see "Specifying the Site ID and Application Number," on page 5-18.
--additionalSystemScriptsDirectory *directory*	Specifies a directory where VR-Forces will look for system scripts when it populates the Scripted Task dialog box script list.
--appDataDir *directory*	Specifies the location of the appData directory.
--appIdRange *lower-upper*	Specifies a range of application IDs that represent objects generated by VR-Forces.
(-B | --batchScenarioFileName) *batch_file*	Runs the specified batch file. (For use with one back-end only.) For more information, please see *Running in Batch Mode from the Command Line*, under "Running VR-Forces in Batch Mode," on page 7-55.

Table 5-2: vrfSim command-line options (Sheet 2 of 7)

Option	Description
(-c | --frontEndPID) *PID*	The process ID for the front-end when running in combined mode.
--countryCodesMappingFile *file_name*	Specifies an alternative file that maps country codes to country names and DIS enumeration values.
(-d | --settingsFile) *file_name*	Specifies a configuration file to load instead of *vrfSimSettings.xml*.
--daemon	Starts that back-end as a daemon process (Linux only). If you start *vrfSim* as a daemon, it does not try to read characters from the console. It forks a background process at startup. It will exit cleanly if it receives a SITGTERM, SIGQUIT, or SIGINT signal.
--dataDir *directory*	Specifies the data directory for the back-end.
--defaultObjectTimeout *timeout*	Specifies the timeout interval for objects.
--diGuyAnimationsFile *file_name*	Specifies an additional configuration file for DI-Guy animations.
--diGuyCharacterDataFile *file_name*	Specifies an additional configuration file for DI-Guy character data.
--disableParallelTick	Execute all tick code serially.
--doNotLoadVrfPlugins	Do not load any VR-Forces plug-ins from the *./plugins* directory.
--enableChannel *channel*	Enables a specific channel for the channel-specific output, even if it is disabled in *channelSettings.mtl*.
--fileNotifyLevel *level*	Notify level in log file to use.
--fullyLoadTerrain	For paged terrains, load all pages.
(-h | --help)	Displays help information in a console window.
--hostAddressString *address*	The IP address of the host computer. Called Device Address in the Launcher.
(-i | --sessionId) *session_ID*	Specifies a session ID. For details, please see "Specifying a Session ID," on page 4-9.
(-L | --scenarioFileName) "*scenario_pathname*"	Loads the specified scenario. The pathname can be absolute or relative from the *./bin* directory. Mutually exclusive with the -T option. For details, please see "Loading a Scenario from the Command Line," on page 7-28.
--loadPlugin *plugin_file*	Specifies one or more plug-ins to load. (Accepted multiple times.)

Table 5-2: vrfSim command-line options (Sheet 3 of 7)

Option	Description
`--logFileName` *filename*	Log file to use.
`--logFrameRateStatistics`	Specifies that the back-end should log frame rate statistics.
`--msdlSIDCMappingFile` *file_name*	Specifies an alternative file for mapping SIDC codes to DIS enumerations.
`(-n | --notifyLevel)` *notification_level*	Sets the level for messages written to the console or the VR-Forces log file (*vrf.log*), where *notification_level* is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2. For details, please see "Setting the Notification Level," on page 5-18.
`--nonVrfObjectsUUIDScheme` *scheme*	Sets the UUID scheme for non vrf objects. Values are `entity-identifier`, `global-identifier`, `marking-text`.
`--numCallbackThreads` *threads*	Number of threads in the callback queue.
`--outputLogFile` *file_name*	The file to use to write out application output.
`(-q | --doNotUseConsole)`	Specifies that all *vrfSim* output go to the log file rather than the console (quiet mode). If you are using a high level of notification, sending output to the console can degrade performance. Running in quiet mode prevents this degradation of performance. This command only applies to back-ends running on Windows. To create a log file on Linux, redirect the output of vrfSim, for example: `vrfSim > mylog.txt` For more details about log files, please see "The VR-Forces Log Files," on page 4-44.
`(-r | --startInRunMode)`	Starts the back-end in run mode. If you start the back-end without specifying a scenario (with `-L`) or a terrain database (with `-T`), the back-end starts in pause mode.
`(-s | --siteId)` *site_id*	Specifies the site ID. For details, please see "Specifying the Site ID and Application Number," on page 5-18.
`--setMainThreadToHighPriority`	Sets the VR-Forces main thread to high priority. When enabled, other threads may be starved if there are insufficient processors available. If running on hardware with limited processor time, this may decrease overall system performance. Windows only.
`--simulationModelSet` *sms*	Specifies the simulation model set. It must be declared separately for front-end and back-end even in combined mode.

Table 5-2: vrfSim command-line options (Sheet 4 of 7)

Option	Description
`--startMinimized`	Minimizes the vrfSim console at startup.
`--suppressDataAvailableException`	Do not throw an exception when dataAvailable is not checked in non-blocking terrain calls.
`(-T | --terrainDatabase)` *terrain_database*	Specifies a terrain database to load in the back-end. Mutually exclusive with the `-L` option. For details, please see "Working with Multiple Back-ends," on page 3-8 and "Loading a Terrain Database at Startup," on page 4-24.
`--targetFrameRate` *rate*	Specifies the frame rate above which vrfSim will yield CPU time to other processes.
`--useAbsoluteTimeStamps`	Enables use of absolute time stamps. Otherwise uses relative time stamps.
`--userDataDir` *directory*	Specifies the user data directory for the back-end.
`(-v | --version)` *version*	Displays version information.
`--waitQueue`	Turn off main thread participation in parallel algorithms.
`(-- | --ignore_rest)]`	Ignore all of the command-line options that are listed after this argument. This lets you quickly limit the arguments applied if you want to edit a script or batch file without losing the original command line.
HLA Only	
`(-f | --fomMapperLib)` *libname*	FOM Mapper library name.
`(-F | --fedFileName)` *fedfile_name*	FED file name.
`--fomMapperInitData` *data*	FOM Mapper initialization data.
`--fomModules` *module*	Specifies an HLA Evolved FOM module. Can be used multiple times.
`--ignoreAdvisories`	Enables or disables HLA advisory messages. Default: true.
`--mimModule`	Specifies an HLA Evolved MIM module.

> **!** If you select `--useAbsoluteTimeStamps`, it is your responsibility to ensure that the time clocks of exercise participants are synchronized. If they are not synchronized, VR-Forces may not work properly.

Table 5-2: vrfSim command-line options (Sheet 5 of 7)

Option	Description
(-N | --federateName) *federate_name*	Specifies a unique name that can be assigned to the federate. No two federates within the same federation can have the same name. If no name is specified, the RTI will automatically assign a unique name for the federate. Default: VR-Forces. (HLA Evolved only)
--noRtiCompilerCheck	Disable RTI Compiler version check.
(-p | --federateType) *type*	Federate Type can be used to distinguish different categories of federates, and its only practical use is in save-and-restore.
--rprFomRevision	RPR FOM revision for RPR FOM 2.0, draft 17.
--rprFomVersion *version_number*	RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0)
(-S | --localSettingsDesignator) *designator*	Specifies a string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage.
--timeManagement	Enables time management for the back-end.
(-x | --execName) *exec_name*	Execution name. Default: VR-Link.
DIS Only	
(-A | --destAddrString) *address*	Destination address.

Table 5-2: vrfSim command-line options (Sheet 6 of 7)

Option	Description
`--attributePduStrategy`	This flag controls how VR-Forces sends DIS attribute PDUs. For the GUI to control VR-Forces Entities, the Simulation Engine must supply additional attributes through DIS. Some applications cannot correctly interpret coupled attribute PDUs in some DIS versions. In those cases, changing the default behavior may be required.The possible values are: ◆ 0: DtAttributePduStrategy_Unspecified ◆ 1: DtAttributePduStrategy_NoAttributePdu ◆ 2: DtAttributePduStrategy_Couple ◆ 3: DtAttributePduStrategy_DoNotCouple ◆ 4: DtAttributePduStrategy_DoNotCoupleAndSendAttributeFirst The default is 0, which means that VR-Forces chooses based on the selected version of DIS. For DIS 6 and lower, it uses 3, and for DIS 7 it uses 2. Mode 3 will not send any additional attributes, so VRF GUI will not recognize the published entities as VRF entities and will not be able to control them.
`--deviceAddress` *address*	Specifies the address of the network card to use for UDP traffic.
`--disVersion`	The DIS version. Default: 7.
`--mcastTtl`	Specifies the multicast time-to-live. Default: -1.
`(-S | --multicastAddresses)` *address1 ... addressn*	Specifies one or more multicast addresses.
`(-P | --disPort)` *port*	DIS port. If you specify a DIS port that is less than 3000 or greater than 4000, discovered objects might not be seen as VR-Forces objects. If you want to use a DIS port outside this range, specify it for the front-end and back-end with the `--appIdRange` command-line argument.
`--recvBufferSize` *size*	Receive buffer size.
`--sendBufferSize` *size*	Send buffer size. Default: -1.
`--subnetMask` *mask*	Specifies the subnet mask. VR-Forces can usually figure out the subnet mask from the IP address. However, if it does not do this correctly, you can explicitly specify the subnet mask.
`--suppressSelfReflect`	Represses self reflection of the DIS exercise connection.
`--useAsyncIO`	Use asynchronous I/O. Default: False.

Table 5-2: vrfSim command-line options (Sheet 7 of 7)

Option	Description
`--useIpv6`	Specifies that the DIS exercise connection use IPV6.
`(-x | --exerciseId)` *ID*	Exercise ID.

5.3. Command-Line Options for vrfLauncher

Table 5-3 describes the command-line options for vrfLauncher.

Table 5-3: vrfLauncher command-line options

Option	Description
`(-B | --backend)`	Start vrfLauncher for the back-end only.
`(-C | --config)`	Open the Simulation Connections Configuration tool.
`(-F | --frontend)`	Start vrfLauncher for the front-end only.
`(-G | --locale)` *locale*	Specifies the language to use.
`--guiArgs`	Arguments after `--guiArgs` are passed only to the front-end.
`(-h | --help)`	Displays help information in a console window.
`--simArgs`	Arguments after `--simArgs` are passed only to the back-end.
`--usePredefinedConnection` *connection*	Specify an already defined connection name, such as DIS localhost, so that the vrfLauncher can launch without needing to display the Simulation Connections dialog box.
`(-v | --version)`	Displays version information.
`--`	Pass all arguments after this to vrfGui and vrfSim, except as modified by `F--` or `B--`.
Special Commands for Arguments that Come After --	
`F--`	Arguments after `F--` are passed only to the front-end.
`B--`	Arguments after `B--` are passed only to the back-end.
`-debug`	Appends a "d" to vrfGui or vrfSim to run the debug version. Can only be used following `B--` or `F--`. (Windows only)

The use of the -- command line option is the opposite of how it is used for other MAK applications.

5.3.1. Running in Combined Mode from the Command Line

If you want to run VR-Forces in combined mode from the command line, there are no command-line arguments for vrfGui or vrfSim to support this. However, you can run vrfLauncher with the `--usePredefinedConnection` argument and VR-Forces will load the specified connection without displaying the Simulation Connections Configuration dialog box. This has the effect of running directly in combined mode from the command line. For example, suppose you want to start in combined mode using the DIS (7) connection. The command would be:

```
vrfLauncher --usePredefinedConnection "DIS (7)"
```

If a predefined connection has spaces in the name, you must surround the name with quotes. You must launch a predefined connection from the Launcher at least once so that VR-Forces can save the network address information it needs to start up.

5.3.2. Using the -- Command-line Argument

The -- command-line argument is used to send additional command-line arguments to the front-end and back-end. This is the opposite of how this argument is used in other MAK applications.

It has some special sub-arguments that allow you to direct some arguments only to the front-end or back-end. It also has an argument that lets you run vrfGui or vrfSim in debug mode. Figure 5-1 illustrates how these arguments are used.

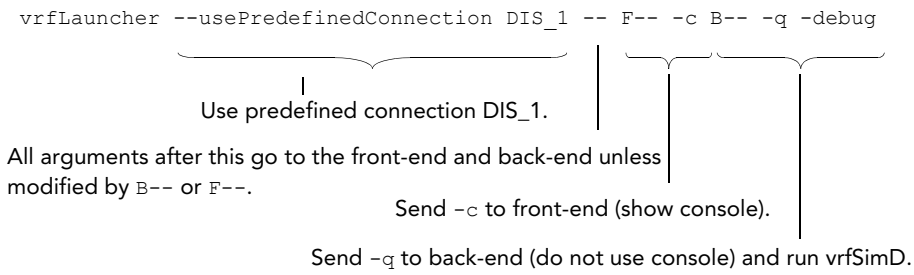


Figure 5-1. Using the -- command-line argument

5.4. Protocol-Independent Command-line Options

This section describes in greater detail some of the most commonly used command-line options that you can use with both DIS and HLA.

5.4.1. Specifying the Site ID and Application Number

Each front-end and back-end must have a unique site ID:application number pair. To specify the site ID, use the `-s` option. To specify the application number, use the `-a` option, for example:

```
vrfSimDIS -s 2 -a 3005
```

The default application number for vrfGui is 3101; for vrfSim it is 3001.

5.4.2. Specifying the Language to Use in the GUI

VR-Forces decides which translation files to use based on the language setting for your computer. However, you can override the language setting with the `-G` command-line option, for example, the following command sets the language to German:

```
vrfGui -s 1 -a 3000 -G de --dis
```

If you use this command-line option, you must have a file named *vrf_language_code* (for example, *vrf_de*) in the *./translations* directory. For more information, please see "Localizing the Graphical User Interface," on page 2-12.

5.4.3. Setting the Notification Level

You can control the types of messages sent to the console (and log file on a PC) by specifying a notification level.

➤ To specify a notification level, use the `-n` option, for example:

```
vrfSimDIS -s 1 -a 3001 -n 1
```

The notification levels you can specify are:

- 0 – Only fatal messages are printed.
- 1 – Warning messages and fatal messages are printed.
- 2 – Some diagnostic information is printed with warnings and fatal messages.
- 3 – Verbose information is printed.
- 4 – Debug information is printed.

The default notification level is 2.



The Linux version of VR-Forces does not have a log file. Informational messages are sent to the window from which you start VR-Forces. To capture informational messages, redirect output to a file.

5.4.4. Loading Plug-ins

You can load plug-ins from the command line or by specifying them in a configuration file. (For details about creating a configuration file, please see "Managing Plug-ins," on page 4-35.)

➤ To load a plug-in from the command line, use the `--loadPlugin` command-line option, for example:

```
vrfGui -s 1 -a 3000 --loadPlugin pluginA
   --loadPlugin pluginB --dis
vrfSimDIS -s 1 -a 3001 --loadPlugin pluginA
   --loadPlugin pluginB
```

! Do not specify the extension of the plug-in (for example, *.dll*). If the extension is included, the plug-in will not load.

VR-Forces can display a list of the plug-ins that are loaded. For details, please see "Viewing a List of Loaded Plug-ins," on page 4-42.

5.5. Command-Line Options for HLA Federations

VR-Forces listens to whatever port is specified in the RID file. For information about the RID file, consult your RTI documentation.

5.5.1. Specifying the Federation Execution

By default, VR-Forces uses the *MAK-RPR2-1-1.fed* file and the MAK-RPR-2.0 federation execution.

➤ To specify the HLA federation execution name, use the `-x` option, for example:

```
vrfSimHLA13 -s 1 -a 3001 -x MAK-RPR-2.0
   --rprFomVersion 2.0 -F MAK-RPR2-1-1.fed
```

! If you are using the MAK RTI and are running multiple, concurrent federation executions, you must run the rtiexec. In other words, you cannot use the MAK RTI in lightweight mode if you are running multiple federations. Please see the MAK RTI documentation for details about how to configure it to use the rtiexec.

5.5.2. Specifying the FED File

In combined mode, specifying `-F` in the command line also applies to the back-end. If you are running a back-end independently and want to specify a FED file name, you must use `-F`.

➤ To specify a FED file at startup, use the `-F` option.

5.5.3. Specifying the RPR FOM Version

VR-Forces has built-in support for the RPR FOM. The default version is 2.0.

You can specify a different RPR FOM version (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0). If you specify a different FOM version, you must also specify the correct FED file (with `-F`).

```
vrfSimHLA13 -s 1 -a 3001 --rprFomVersion 2.0
  -F MAK-RPR2-1-1.fed
```



Please see *VR-Forces Release Notes* for a list of the RPR FOM versions supported by your version of VR-Forces.

5.5.4. Specifying a FOM Mapper

VR-Forces supports alternative FOMs through the VR-Link FOM Mapper. You can specify which FOM Mapper to use with the `-f` command-line option:

```
vrfSimHLA13 -s 1 -a 3001 -x MAK-RPR-2.0
  --rprFomVersion 2.0 -F MAK-RPR2-1-1.fed
  -f myFomMapper
```

5.5.5. Specifying the RPR FOM Revision

MAK revises FED files when it discovers errors in how it has previously handled various aspects of the HLA standard. It maintains multiple revisions to ensure compatibility with applications that depend on previous revisions.

➤ To specify the RPR FOM revision, use the `--rprFomRevision` *`revision`* command-line option, for example:

```
vrfSimHLA13 -s 1 -a 3001 --rprFomVersion 2.0017
  -F MAK-RPR2-1-1.fed --rprFomRevision 2
```

5.5.6. Specifying FOM Mapper Initialization Data

Some FOM Mappers require initialization data. If you specify a FOM Mapper that requires such data, specify the data with the `--fomMapperInitData` option:

```
vrfSimHLA13 -s 1 -a 3001 -x MAK-RPR-2.0
  --rprFomVersion 2.0 -F MAK-RPR2-1-1.fed
  -f myFomMapper --fomMapperInitData ABC
```

The string "ABC" is passed to the FOM Mapper creation function in the shared library as an initialization parameter.

5.5.7. Specifying a FED File That is Appropriate for the FOM Mapper

The `-f` option only tells the application which FOM Mapper to use. It does not ensure that you are using the right FED file for the FOM Mapper. Make sure that you are using a FED file that is consistent with the FOM Mapper you have chosen.

You can specify a FED file using the `-F` option.

The following example instructs the application to use the RPR FOM 2.0, version 17 FOM mapping code, in the federation execution MAK-RPR20017-1-1, with the *MAK-RPR20017-1-1.fed* file:

```
vrfSimHLA13 -s 1 -a 3001 --rprFomVersion 2.0017
   -x MAK-RPR20017-1-1 -F MAK-RPR20017-1-1.fed
```

5.5.8. Enabling Time Management

You can enable time management for the back-end from the command line, as follows:

```
vrfSimHLA13 -s 1 -a 3001 --timeManagement
   -F MAK-RPR1-TimeManagement-1-1.fed
   -x MAK-RPR1-TimeManagement-1-1
```

5.6. Command-Line Options for DIS Exercises

This section describes command-line options that apply only to using VR-Forces with DIS exercises.

5.6.1. Specifying the Port Number

By default, VR-Forces listens to port 3000.

If you specify a DIS port that is less than 3000 or greater than 4000, discovered objects might not be seen as VR-Forces objects. If you want to use a DIS port outside this range, specify it for the front-end and back-end with the `--appIdRange` command-line argument.

➤ To change the port that VR-Forces listens to, use the `-P` option, for example:

```
vrfSimDIS -s 1 -a 3001 -P 3500
vrfGui -s 1 -a 3000 --dis -P 3500
```

5.6.2. Specifying Point-to-Point or Multicast Operation

You can specify that VR-Forces run in point-to-point mode or multicast.

➤ To specify point-to-point, use the `-A` option, for example:

```
vrfSimDIS -s 1 -a 3001 -A 123.123.12.1
```

➤ To specify multicast, use the `-S` option, for example:

```
vrfSimDIS -s 1 -a 3001 -S 233.233.22.3
```

You can specify more than one multicast address, as follows:

```
vrfSimDIS -s 1 -a 3001 -S address -S address ...
```



The default is broadcast mode.

5.6.3. Specifying the Multicast Time-to-Live

The multicast time-to-live specifies the number of routers that a message can pass through when PDUs are sent to multicast addresses.

➤ To specify the multicast time-to-live value, use the `--mcastTtl` option, for example:

```
vrfSimDIS -s 1 -a 3001 -S address -S address --mcastTtl 3
```

5.6.4. Using Asynchronous I/O

If you want to use asynchronous I/O for DIS exercises, start VR-Forces with the `--useAsyncIO` option. Using asynchronous I/O can help prevent dropped packets and the resulting loss of information. There is no comparable command-line option for HLA. To use asynchronous I/O in HLA, you set a RID file parameter.

5.6.5. Specifying the Exercise ID

➤ To set the VR-Forces exercise ID, use the `-x` option, for example:

```
vrfSimDIS -s 1 -a 3001 -x 6
```

5.6.6. Specifying the DIS Version

➤ To specify the DIS version, use the `--disVersion` command-line options, for example:

```
vrfSimDIS -s 1 -a 3001 --disVersion 6
```

Please see *VR-Forces Release Notes* for a list of supported versions.



6. Optimizing Performance

This chapter explains how you can view performance statistics for the front-end and back-end and how you can optimize GUI performance.

Introduction 6-3
- Optimizing Simulation Engine Performance 6-3
- Optimizing Front-End Performance 6-4

Monitoring the Back-end Frame Rate 6-6

Monitoring Script Frame Rates 6-6

Displaying Performance Statistics 6-7
- Displaying VR-Forces Performance Statistics 6-9
- Displaying OSG Statistics 6-9
- Displaying the VR-Forces Function Profiler 6-10

Filtering Simulation Objects Using Interest Management 6-10
- Enabling Interest Management 6-10
- Configuring Interest Management 6-11

Clearing the Model Instancing Cache 6-13

Miscellaneous Performance Configuration Options 6-14
- Limiting Use of Spot Reports 6-14
- Using Asynchronous I/O for DIS Exercises 6-14
- Using Incremental Compiling with Streamed Data 6-14

Setting the Tick Rate 6-15
- Specifying the Tick Rate for Components 6-15
- Tuning the Network Interface 6-16
- Tuning the State Repository Tick Rate 6-16

Configuring Graphics Quality 6-16
- Alpha to Coverage 6-17

Balancing Visual Quality Against Network Performance 6-19
- Enabling Configuration of Performance Settings 6-20

Trajectory Smoothing 6-21
Configuring Trajectory Smoothing 6-22
Tuning the Target Frame Rate 6-22
Coloring Draw Calls 6-23
Configuring VSync 6-24
Preloading Models 6-24
Command-Line Options for Preloading Models 6-25
DI-Guy Performance Settings 6-26
Using the Developer Debug Overlay 6-27

6.1. Introduction

This chapter describes ways that you can optimize performance through the GUI, the object parameter database, and command line options. You can also optimize performance programmatically.

VR-Forces's overall performance varies based on the following factors:

- Size of the terrain database.
- Amount of memory available on your computer.
- Computer speed and processing power.
- The number of objects (simulation objects and tactical graphics) in the scenario and how active the simulation objects are.
- The number of objects you try to simulate on a given back-end.
- The frequency of the tick rate.
- The number of front-ends and back-ends you run on your computer.

6.1.1. Optimizing Simulation Engine Performance

VR-Forces supports the following methods for optimizing the performance of the simulation engine:

- Adjusting the refresh rates. (For details, please see "Balancing Visual Quality Against Network Performance," on page 6-19.)
- Turning off processor-intensive features. (For details, please see "Miscellaneous Performance Configuration Options," on page 6-14.)
- Specifying the tick rate for a simulation object or component. (For details, please see "Setting the Tick Rate," on page 6-15.)
- Specifying a tick rate for the network interface. (For details, please see "Tuning the Network Interface," on page 6-16.)
- Specifying the VR-Forces sleep time. (For details, please see "Tuning the Target Frame Rate," on page 6-22.)
- Optimizing the terrain. (For details, please see various chapters in the Section XI, *Terrain*, particularly Chapter 54, *Introduction to Terrain Usage in VR-Forces* and Chapter 64, *Terrain Performance and Configuration*.)
- Using interest management. (For details, please see "Filtering Simulation Objects Using Interest Management," on page 6-10.)

6.1.2. Optimizing Front-End Performance

The performance of the front-end depends on:

- Processor speed.
- Video card. (Always use the latest drivers for your video card.)
- Available memory.
- Size of the terrain database.
- The number and complexity of models loaded and how well they have been optimized for performance. For information about optimizing models, please see "Best Practices for Creating Models," on page 90-34.
- Number of objects in the simulation and how active they are.
- Use of model instancing.

You can display statistics to help you evaluate VR-Forces's performance. For details, please see "Displaying Performance Statistics," on page 6-7.

The following display features can affect performance:

- Displaying performance statistics.
- Use of ground clamping.
- Displaying track histories.
- Displaying height-above-terrain lines.
- Displaying radio communications lines.
- Enabling advanced lighting and various lighting effects.
- Enabling dynamic ocean effects. When you are using Plan View observer mode, enabling wakes, spray effects, and the buoyancy model can seriously degrade performance. These features are disabled by default. If you inadvertently enable them and experience this problem, switching to Stealth mode and back to Plan View mode restores performance.
- Sound effects. Disabling sound effects can improve performance.
- Anti-aliasing settings. For details, please see "Configuring Graphics Quality," on page 6-16.

In addition to the sections in this chapter and the sections in Chapter 64, *Terrain Performance and Configuration*, the following sections describe options for optimizing front-end performance:

- "Configuring SpeedTree Trees," on page 90-30.
- "Clearing the Model Instancing Cache," on page 6-13.
- "Setting the Clipping Planes," on page 82-9.
- "Considerations and Limitations for Building Terrains," on page 57-2.
- "Filtering Simulation Objects Using Interest Management," on page 6-10.
- "Setting the Ocean LOD Elevation," on page 57-17.
- "Configuring the Ocean Height Map," on page 57-17.
- "Displaying Wakes and Spray Effects," on page 46-7.
- "Configuring the Skybox Cube Map," on page 45-15.
- "Improving Performance when Attaching to Ground Objects," on page 52-7.
- "Enabling Buoyancy for Surface Entities," on page 46-8.
- Chapter 45, *Lighting Effects*.

6.2. Monitoring the Back-end Frame Rate

You can monitor frame rate statistics for each back-end in your session. The Frame Rate Monitor Panel lists statistics for each back-end (Figure 6-1).

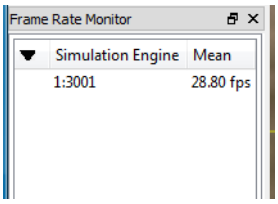


Figure 6-1. Frame Rate Monitor

➤ To view the Frame Rate Monitor Panel, choose **View** → **Frame Rate Monitor Panel**.

6.3. Monitoring Script Frame Rates

VR-Forces can display the average time per tick for scripted tasks and other scripts. When this is enabled, the Scripts window shows the frame rate for all active scripts (Figure 6-2).

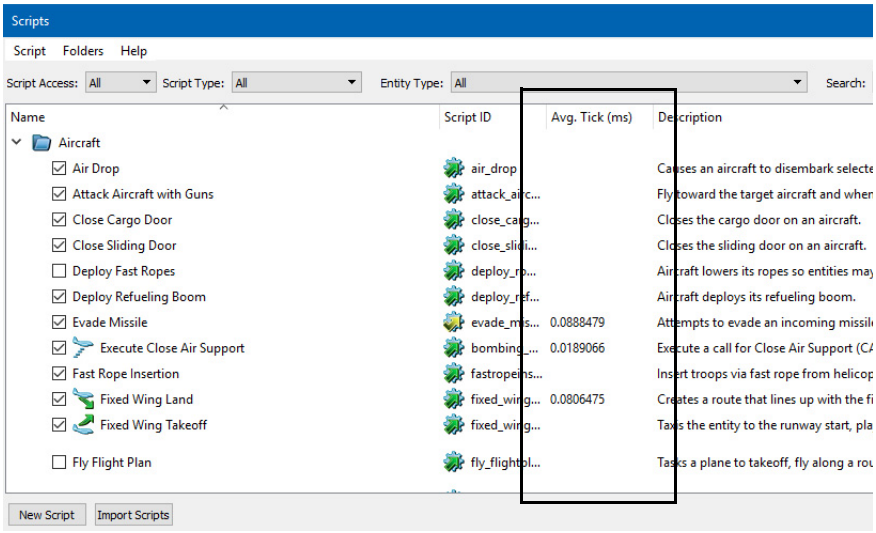


Figure 6-2. Script frame rate

For information about scripts and the scripts window, please see "Creating Scripted Tasks and Sets," on page 31-1

To display the frame rate for scripts:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Script Settings page (Figure 6-3).

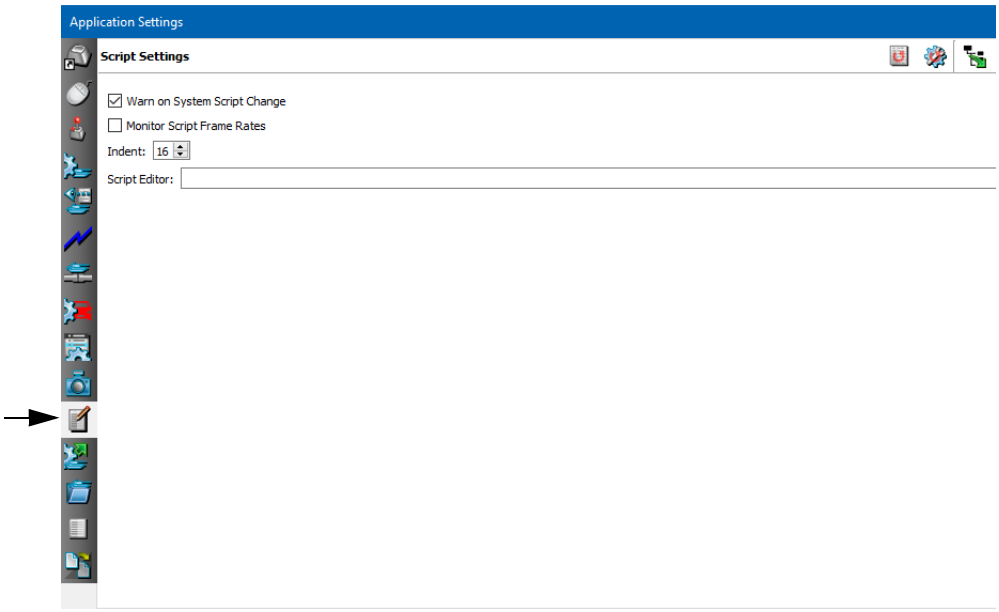


Figure 6-3. Script Settings page

3. Select Monitor Script Frame Rates.
4. Choose **Simulation** → **Scripts**. The Scripts window opens. A column called Avg Tick is added. It displays the tick rate for active scripts.

6.4. Displaying Performance Statistics

VR-Forces can display two kinds of performance statistics — OSG statistics (Figure 6-4), and VR-Forces statistics (Figure 6-5). VR-Forces also has a function profiler that developers can use to assess the performance of new code.

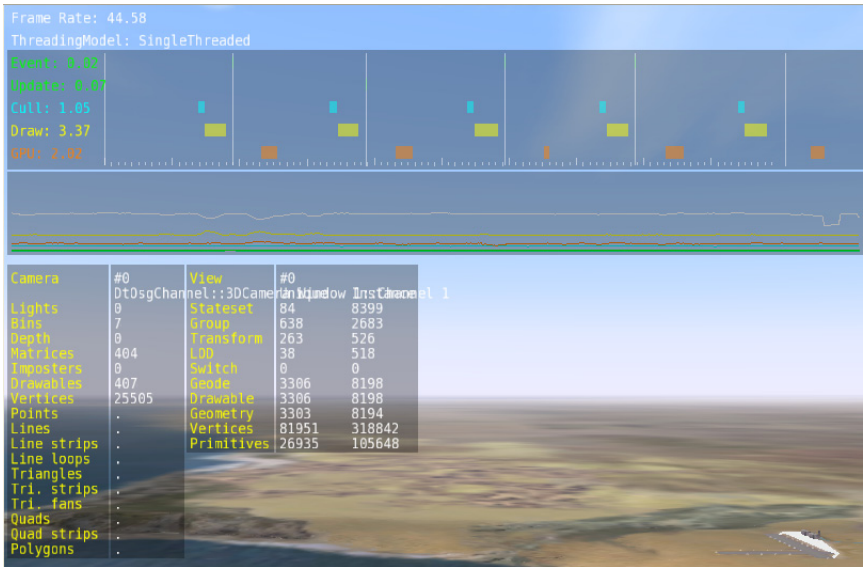


Figure 6-4. OSG performance statistics

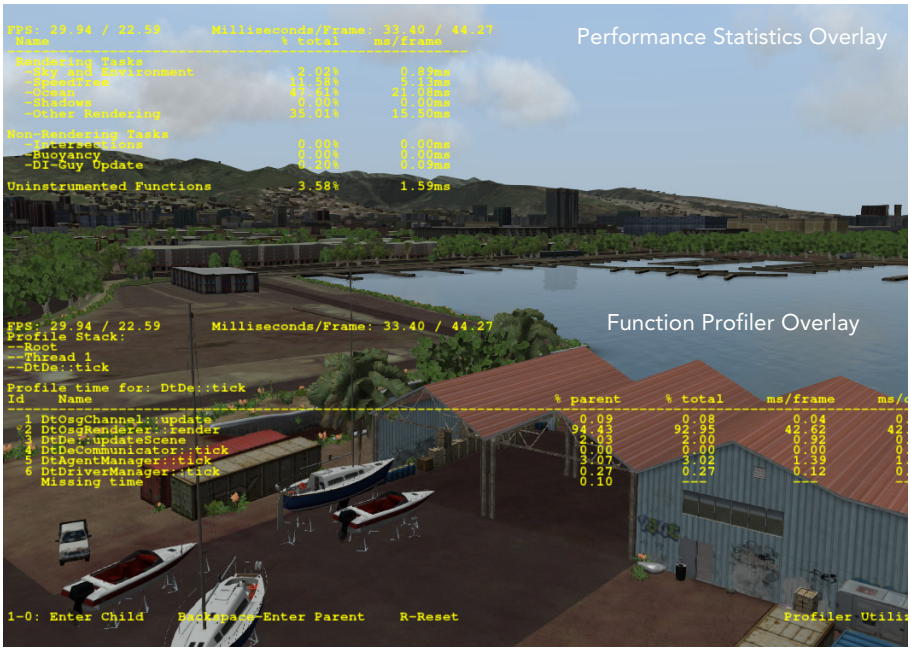


Figure 6-5. VR-Forces performance statistics and function profiler



Displaying performance statistics may affect performance. Displaying OSG statistics may affect performance more than VR-Forces statistics.

6.4.1. Displaying VR-Forces Performance Statistics

To display VR-Forces performance statistics:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-6).

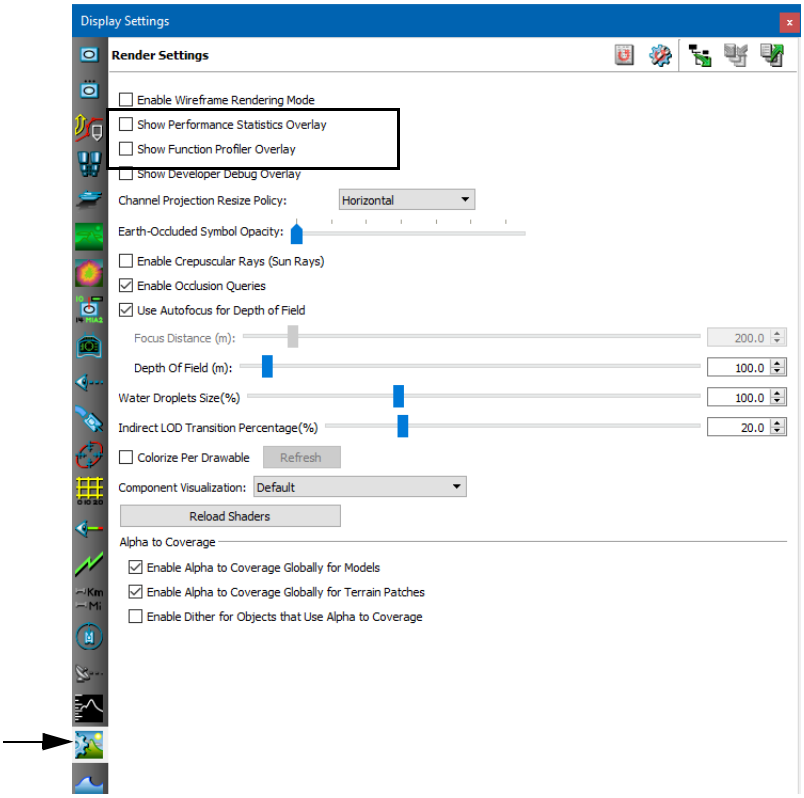


Figure 6-6. Render Settings page

3. Select the Show Performance Statistics Overlay check box. A statistics display is added to the window (Figure 6-5).

6.4.2. Displaying OSG Statistics



If you display OSG statistics and VR-Forces statistics at the same time, they overlay each other.

To display OSG performance statistics:

1. Press **F2**. The frame rate is displayed.
2. Continue pressing **F2**. Additional statistics are displayed. When all statistics are displayed, pressing **F2** closes the display.

6.4.3. Displaying the VR-Forces Function Profiler

The VR-Forces Function Profiler is primarily for the use of developers who are debugging VR-Forces applications. For information about how to integrate the profiler into your code, please see Chapter 11, Using the Function Profiler in *VR-Forces Developers Guide*.



For information about another debugging utility, please see "Using the Developer Debug Overlay," on page 6-27.

To run the VR-Forces Profiler:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-6).
3. Select the Show Function Profiler Overlay check box. A statistics display is added to the window (Figure 6-5).

6.5. Filtering Simulation Objects Using Interest Management

VR-Forces can use interest management to improve front-end performance in simulations that have high simulation object counts. Interest management is an implementation of HLA data distribution management (DDM). When interest management is enabled, VR-Forces filters out all simulation objects that are more than a specified distance from the observer.

6.5.1. Enabling Interest Management

To use interest management:

- You must use HLA (DIS does not support interest management).
- If you are using the MAK RTI, you must use MAK RTI 3.4a or later.
- You must configure your RTI to support DDM. (If you are using the MAK RTI, you can configure your system to use *ddm-rid.mtl*, which has DDM enabled.)

➤ To enable interest management, start vrfGui from the command line and include the `--useGeographicDdm` command line option with the HLA configuration commands, for example:

```
vrfGui -s 1 -a 3004 --hla -x MAK-RPR-2.0
  --rprFomVersion 2.0 -F MAK-RPR2-1-1.fed
  --useGeographicDdm
```

6.5.2. Configuring Interest Management

Interest management works by filtering out all simulation objects beyond a specified distance from the observer. You can configure this distance (DDM region size) and you can configure the threshold for observer movement (region update threshold) before VR-Forces recalculates which simulation objects should be displayed.

VR-Forces uses a geographic DDM scheme. The earth is divided into two dimensional regions, with X pointing north and Y pointing east. When you specify a DDM region size, you specify the length of one side of a region.

The region update threshold is a percentage of the distance from the center of the region to a side. When the observer moves beyond the threshold, the region is recalculated. For example, if the region size is 1000 meters, the distance from the center to a side is 500 meters. If the region update threshold is 50%, when the observer moves more than 250 meters from the center of the region, VR-Forces recalculates the display region.

DDM regions use a two-dimensional grid, so the altitude of the observer is not considered when filtering simulation objects. However, in 3D observer modes, you usually cannot see entity models once the observer is a certain distance above the terrain (although you can see labels). Therefore, you may want to filter out simulation objects based on the observer's altitude, thereby increasing performance.

To configure interest management:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Interest Management Settings page (Figure 6-7).

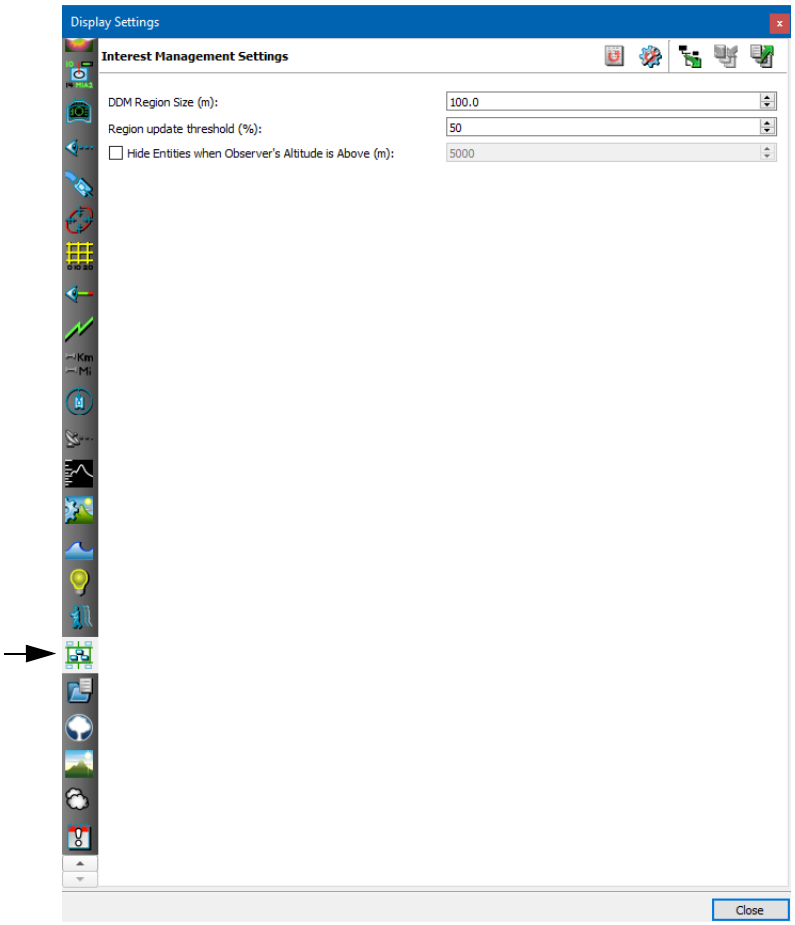


Figure 6-7. Interest Management Settings page

3. In the DDM Region Size box, type a value for the region size.
4. In the Region Update Threshold box, type a value, as a percentage, for the update threshold.
5. Optionally, filter simulation objects based on the observer's altitude, as follows:
   a. Select the Hide Entities When Observer's Altitude is Above check box.
   b. Specify the observer altitude at which to hide simulation objects.
6. Click OK.

6.6. Clearing the Model Instancing Cache

VR-Forces has a feature called model instancing that lets models share resources, such as geometry and textures. These shared resources are stored in a cache. Use of model instance caching decreases memory use and improves performance.

You can clear the model instancing cache if you need the disk space.

To clear the model instancing cache:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 6-8).

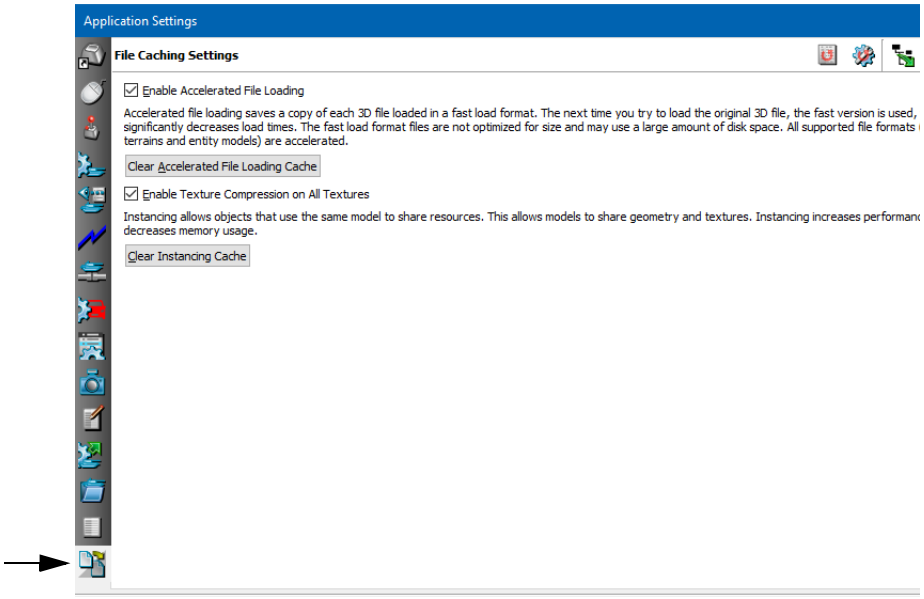


Figure 6-8. File Caching Settings page

3. Click Clear Instancing Cache.

6.7. Miscellaneous Performance Configuration Options

You can improve front-end performance by limiting use of display features, as follows:

- Turn off track histories.
- Limit use of object transparency.
- Limit use of spot reports.
- In DIS, use asynchronous I/O.
- For streamed data, use incremental compiling.

6.7.1. Limiting Use of Spot Reports

Use of spot reports can degrade performance. You may want to limit the number of simulation objects for which they are generated or disable spot reports entirely. You can also specify that spot reports be sent only to the front-end. For details about using spot reports, please see "Displaying Spot Reports," on page 9-2.

6.7.2. Using Asynchronous I/O for DIS Exercises

By its nature, DIS generates a lot of network traffic. Therefore, you may experience problems with dropped packets and simulation objects may time out. You can mitigate this problem by using asynchronous I/O. To use asynchronous I/O, start with the `--useAsyncIO` command line option or select Use Asynchronous I/O in the Simulation Connections Configuration dialog box.

6.7.3. Using Incremental Compiling with Streamed Data

Users of streamed terrains might want to use incremental compiling. This is an OSG option that allows the processing of streamed data to be spread across multiple frames, reducing performance spikes. To use it, create the environment variable OSG_ICO. (It does not need a value. It just needs to exist.)



Using OSG_ICO may cause LOD issues with other paged formats.

6.8. Setting the Tick Rate

VR-Forces lets you tune performance by varying the tick rate of components, state repositories, and network interfaces. By decreasing the frequency with which some simulation object components are ticked, you make more CPU time available to other simulation objects. For example you could decrease the tick rate for slow moving entities such as dismounted infantry or command posts, making more CPU available for entities such as aircraft or missiles.

Variable tick rates are determined by a minimum tick period and a minimum tick period variance. The minimum tick period parameter specifies the shortest period of time, in seconds, that can elapse between ticks of a simulation object's components. VR-Forces may tick the components less frequently (due to heavy CPU load), but it will never tick the entity's components more frequently than specified.

To reduce the frequency that a component can be ticked, increase the minimum tick period value. To increase the frequency that a component can be ticked, reduce the minimum tick period value.

The minimum tick period variance argument specifies an amount of variance allowed around the minimum tick period. Specifying a minimum tick period variance value helps prevent the spikes in CPU usage that would result if all components tried to tick at the same time. Calculation of the time to tick a component using minimum tick period variance is as follows (the example uses the entity component arguments):

If the current time is t_0 and the next scheduled tick is t_{next} then:

$$t_{next} = t_0 + \text{min-tick-period} +/- \text{rand}(\text{min-tick-period-variance}/2)$$

If you insert some sample values (in seconds), you get:

$$t_{next} = .1 + .05 +/- \text{rand}(.05/2)$$

This means that the next tick will be no sooner than the current time plus .15 seconds plus or minus a random value between 0 and .0125 seconds.



If the next tick time calculated by this formula is less than the current tick time, it is set to the current tick time, which results in the component being ticked on the next tick.

6.8.1. Specifying the Tick Rate for Components

In the object parameter database, you can specify a min-tick-period and min-tick-period-variance for each simulation object and for individual simulation object components. The component-level settings, if any, override the simulation object-level settings.

6.8.2. Tuning the Network Interface

You can tune the local and remote network interface subcomponents. Specify the net-interface-min-tick-period and net-interface-min-tick-period-variance parameters in the local and remote subcomponents blocks. If you reduce the rate at which the network interfaces are ticked, you can limit how often VR-Forces needs to do coordinate system conversion from network (geocentric) to database coordinates.



Reducing the frequency with which you tick network interface components could delay the discovery of objects in remote front-ends and back-ends.

6.8.3. Tuning the State Repository Tick Rate

You can tune the frequency with which state repositories get ticked. Specify the state-repository-min-tick-period and state-repository-min-tick-period-variance parameters in the local and remote subcomponents blocks. These parameters control how often VR-Forces recomputes the bounding volume of an object based on object geometry.

6.9. Configuring Graphics Quality

VR-Forces has command-line options that let you configure graphics quality. In particular, depending on the capabilities of your graphics card, you may need to adjust the default settings for anti-aliasing using `--antiAliasing` *`level`*. Anti-aliasing is a technique used to reduce the jagged edges of digital graphics. You can set the level to 0, 2, 4, 8, or 16. Default: 4.

On older graphics cards, you may need to set anti-aliasing to 0 to improve performance.

6.9.1. Alpha to Coverage

Alpha to coverage is a way of handling objects with transparency that improves rendering of objects that have a sharp transition between opaque and transparent. Grass, trees, and fences are a few examples of when alpha to coverage is a good option, as shown in Figure 6-9. In addition to improving visualization of some objects, there is a slight performance improvement when alpha to coverage is enabled.



Off On

Figure 6-9. Alpha to coverage

You can set alpha to coverage for models and terrains. For models, you can enable it globally or in specific model definitions. For terrains, you can enable it globally or for specific terrain patches.

By default, alpha to coverage is enabled globally for models and terrains.

If alpha to coverage is enabled, you can enable a dither shader that modifies the alpha with a noise pattern so that there are no hard edges between different alpha levels. The trade-off for this setting is that it adds a small amount of noise as the object or camera moves.

VR-Forces uses render bin 1 for the alpha to coverage render bin. You can change the render bin with the VRV_ALPHA_TO_COVERAGE_RENDERBIN environment variable. (Render bins are an OpenSceneGraph concept that affects the order in which scene objects are rendered. Do not set the render bin environment variable unless instructed to do so by a plug-in developer.)

Enabling Alpha to Coverage for Models

You can enable alpha to coverage globally for models and you can specify it for specific models. When you enable alpha to coverage globally, it affects all models that do not have it specified specifically in their model definition.

To enable or disable alpha to coverage for a specific model, add the `alphaToCoverageEnabled` parameter to the model definition and set it to true or false. For information about editing model definitions, please see "Editing a Model Definition," on page 87-8.

To enable or disable alpha to coverage globally for models:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-6).
3. Select or clear the Enable Alpha to Coverage Globally for Models check box.

Enabling Alpha to Coverage for Terrains

You can enable alpha to coverage globally for terrains or for specific terrain patches. The global setting affects all terrain patches that do not have an alpha to coverage option specified in their MTF file.

You enable alpha to coverage for a terrain patch when you add it. For details, please see "Adding Elevation Data (Terrain Patches) to a Terrain," on page 57-4.

To enable or disable alpha to coverage globally for terrains:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-6).
3. Select or clear the Enable Alpha to Coverage Globally for Terrain Patches check box.

Enabling the Dither Shader for Alpha to Coverage

To enable or disable the dither shader for alpha to coverage:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-6).
3. Select or clear the Enable Dither for Objects that Use Alpha to Coverage check box.

6.10. Balancing Visual Quality Against Network Performance

VR-Forces allows you to tune performance by adjusting dead-reckoning thresholds. The primary benefit of this feature is to improve the appearance of entities in 3D visualization programs such as VR-Vantage Stealth and the VR-Forces Stealth observer mode. Reducing thresholds improves the accuracy of entity movement and reduces jumpy movement. However it does so at the expense of network performance, because a lower dead-reckoning threshold results in more packets being sent for each entity.

When you change thresholds at runtime, VR-Forces sends a message to all back-ends in the scenario to change their default thresholds for translation and rotation. The new thresholds remain in effect until the back-ends are closed. They apply to any scenario run during this time and they override thresholds set in *./appData/settings/vrfSim/vrfSimSettings.xml*. Changes to the default thresholds get saved in *vrfSimSettings.xml*.

If your exercise sends periodic updates, such as a DIS heartbeat, the thresholds evaluate changes since the most recent heartbeat.

Changes made at runtime have no effect in the following cases:

- If a back-end joins the scenario after the thresholds are sent. It uses the default VR-Link thresholds, or the values in the *vrfSimSettings.xml* file, if specified.
- If an object was instantiated with a customized VR-Link *DtThresholder*.

To configure performance settings:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Performance Settings page (Figure 6-10).

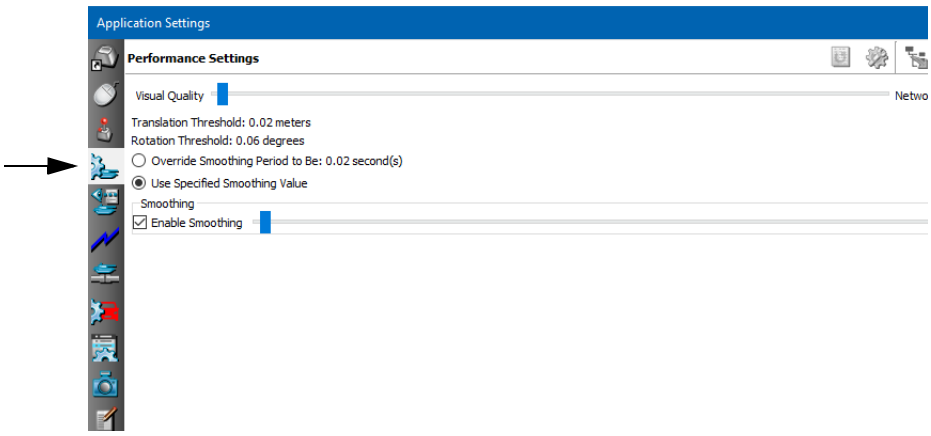


Figure 6-10. Performance Settings page

3. Adjust the Visual Quality slider to your desired preference for visual quality versus network performance.

4. To have VR-Forces calculate the smoothing period, select Override Smoothing Period. To specify the smoothing period yourself, select Use Specified Smoothing Value and use the slider to set the smoothing period you want to use.

6.10.1. Enabling Configuration of Performance Settings

The dr-allow-gui-overrides configuration parameter, specified in each OPE file, determines whether a given entity type can have its translation and rotation thresholds overridden by the value set by the Performance Settings page.

6.11. Trajectory Smoothing

From the time VR-Forces receives a state update message for a simulation object until it receives the next update, it calculates the location of the simulation object based on the heading and acceleration in the most recent state update. This process is called dead-reckoning. When the next state update arrives for the simulation object, VR-Forces moves it to the new position. If the actual position is significantly different from the dead-reckoned position, the simulation object could appear to jump from the dead-reckoned position to the actual position. VR-Forces can eliminate these discontinuities by smoothing trajectories and rotation paths.

Figure 6-11 illustrates dead-reckoning and smoothing.

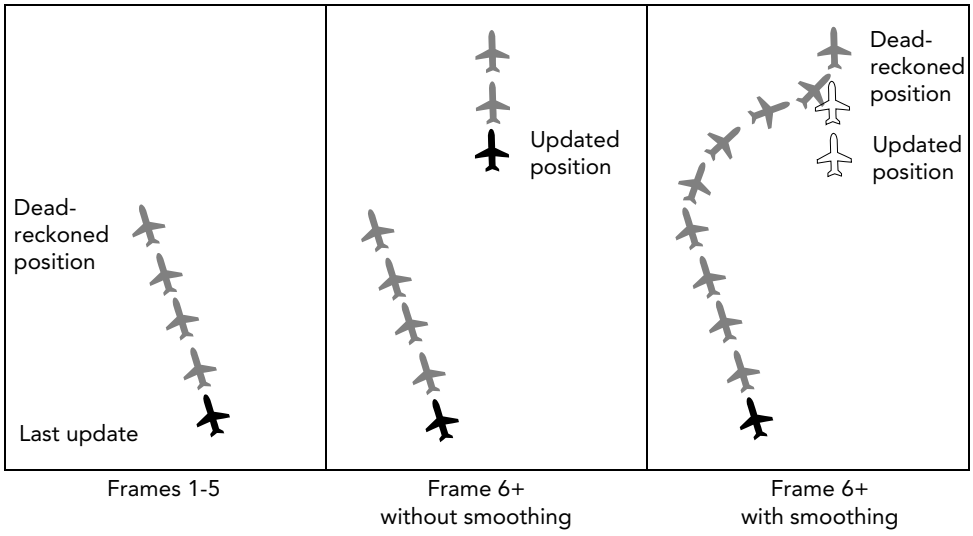


Figure 6-11. Dead-reckoning and smoothing

The smooth period is the period of time over which smoothing takes place. The default period is 1/10 second. If course corrections appear abrupt with this setting, you may want to increase the smooth period.



Smoothing makes fast-moving entities, such as fixed wing aircraft, look better. Slow moving entities, such as dismounted infantry, may look worse. If your simulation is mostly one or the other of these types of entities, enable or disable smoothing as appropriate for the best visualization experience.

6.11.1. Configuring Trajectory Smoothing

To configure trajectory smoothing:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Performance Settings page (Figure 6-10).
3. Select the Use Specified Smoothing Value option. The Enable Smoothing check box becomes enabled.
4. To enable or disable smoothing, select or clear the Enable Smoothing check box.
5. To change the smooth period, adjust the Smoothing Period slider or type a value, in seconds, in the box.

6.12. Tuning the Target Frame Rate

The `--targetFrameRate` command line option and *vrfSim.mtl* parameter allow you to specify the frame rate above which the VR-Forces back-end should begin to sleep in order to give up CPU time to other processes. This parameter is only applicable to variable frame mode. The fixed frame modes already have a built in sleep that gives up unused time. Increasing the target frame rate prevents the back-end from sleeping only if it cannot achieve that rate – ultimately it is up to the operating system to determine how much CPU time each process receives.

6.13. Coloring Draw Calls

One of the factors that affects performance is the number of draw calls it takes to render a scene. VR-Forces lets you color the draw calls in a scene so that you can see which objects may be creating performance problems.

When you color draw calls, all the drawables in the scene are randomly colorized. The colors themselves do not convey any meaning. The important information is the number of draw calls required for a model. For example, if a vehicle has 25 different colored parts, you know that it requires 25 draw calls and you may want to edit the model to use fewer of them.

Figure 6-12 illustrates a tank with its drawables colorized.

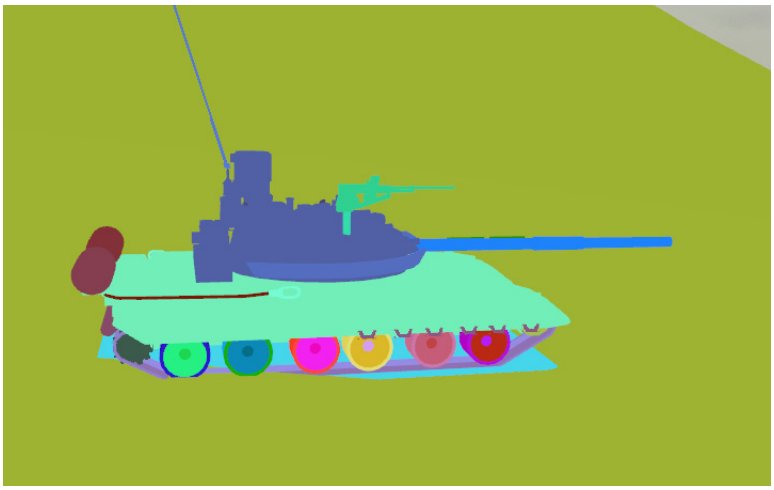


Figure 6-12. Colorized drawables on tank

The colors do not update as new information comes into the scene. Therefore, you can refresh the colorization.

To visualize draw calls:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-6).
3. Select the Colorize Per Drawable check box.
4. To refresh the display, click Refresh.

6.14. Configuring VSync

VSync (vertical synchronization) synchronizes VR-Forces's frame rate with your monitor's refresh rate. This prevents a visual anomaly called tearing, in which parts of two frames of data can be visible at the same time, but it caps the maximum render frame rate based on the display. Typically this cap is 60hz.

In some cases, if the frame rate cannot quite keep up with the refresh rate, VSync can force the frame rate to a noticeably lower rate. To avoid this problem, we recommend that you enable triple buffering for your video card. You must do this in the video card's configuration application; VR-Forces cannot do this for you.

By default, VSync is disabled. If you want to enable it, start up with the `--enableVSync` command-line option, or you can configure your video card to force it to be on.



For `--enableVSync` to work, your graphics card must be set to "Use application setting".

6.15. Preloading Models

When VR-Forces discovers simulation objects on the network, it must load the visual models for these objects. This can affect performance.You can reduce the likelihood of this problem by loading models before you connect to an exercise.

To preload models, you create a list of the models that you expect to need. VR-Forces loads these models when it starts up. You can also load them immediately.

While VR-Forces is participating in an exercise, you can capture a list of models that have not been preloaded. You can then add this list of models to the default preload list or specify the list in a command-line argument.

By default, the list of models to be loaded is saved to *./appData/settings/vrfGui/default_ModelPreload.mpsx*. You can export the list to a different file name. You can also import and merge lists.

To configure the list of models for preloading:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Model Preload Configuration page (Figure 6-13).

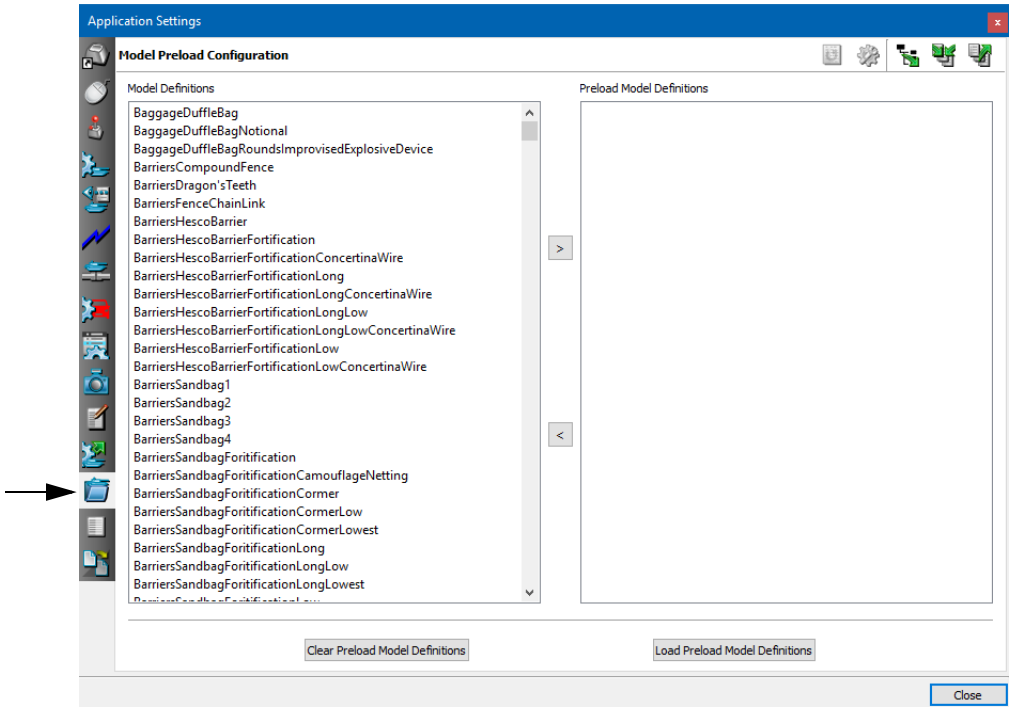


Figure 6-13. Model Preload Configuration page

3. In the Model Definitions list, select the models that you want to preload. Use the standard methods of multiple object selection.
4. Click the right-facing arrow to move the models into the Preload Model Definitions list.
5. To load the models immediately, click Load Preload Model Definitions.

6.15.1. Command-Line Options for Preloading Models

By default, when VR-Forces starts, it loads the models listed in *default_ModelPreload.mpsx*. You can specify that it load a different model list with the `--modelPreloadFileName` command-line option. The syntax is:

```
--modelPreloadFileName filename
```

where *`filename`* is the name of the file to load.

While VR-Forces is participating in an exercise, it can capture a list of models that were not preloaded. You can then merge this list with the default preload file or specify it as an alternative to the default file on the command line. To capture a list of models, use the following command:

```
--discoveredModelPreloadFileName filename
```

where *`filename`* is the name of the file to save the list to.

6.16. DI-Guy Performance Settings

The following settings affect performance of DI-Guy human characters:

- Maximum draw distance. VR-Forces will not render characters that are farther away from the observer than this distance.
- Use multi-threaded update. Developers who are debugging DI-Guy may want to disable multi-threaded update to make it easier to debug.
- Enable instancing. Enabling instancing improves performance by reusing model data for similar characters. You can set the LOD beyond which instancing is used. You can visualize instanced characters.

To specify DI-Guy performance settings:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the DI-Guy Settings page (Figure 6-14).

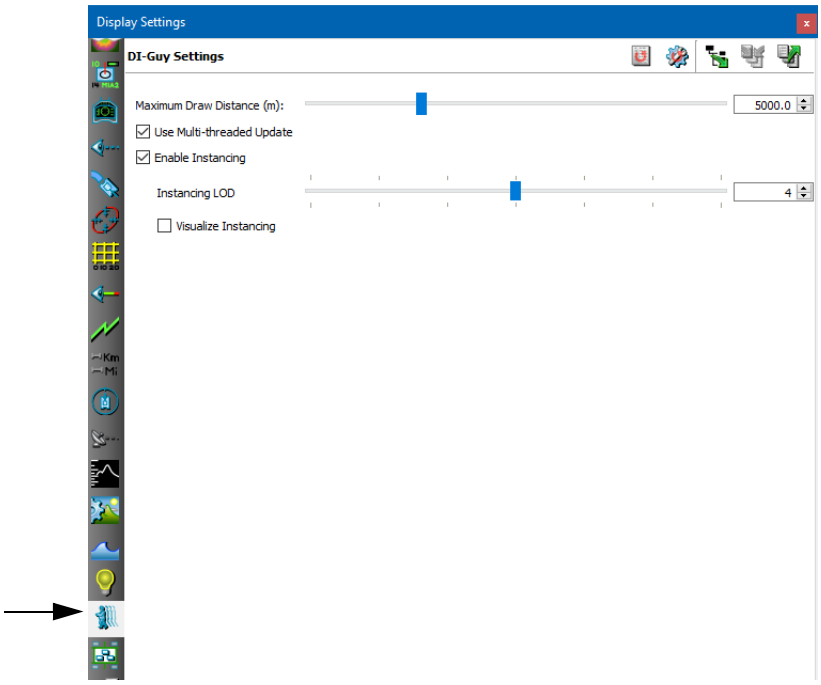


Figure 6-14. DI-Guy Settings page

3. To set the maximum draw distance, adjust the Max Draw Distance slider or change the value in the box.
4. Optionally, select or clear the Use Multi-threaded Update check box.
5. Optionally, select or clear the Enable Instancing check box.
6. If instancing is enabled, adjust the Instancing LOD slider or change the value in the box.
7. Optionally, select or clear the Visualize Instancing check box.

6.17. Using the Developer Debug Overlay

VR-Forces uses the ImGui API to create an overlay that developers can use to test changes to some of its visual features. VR-Forces includes an overlay for testing ocean settings, shadow settings, and other visual features. It also includes an ImGui test user interface that demonstrates some of its features.



Figure 6-15. Developer debug overlay



This feature is provided “as is” for the use of interested developers. For complete information about how to use ImGui, please see documentation at https://github.com/ocornut/imgui and other web pages.

To enable the developer debug overlay:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-6).
3. Select the Show Developer Debug Overlay check box. The Developer Debug Overlay opens. It has a menu for Ocean Settings.
4. To view shadow settings:
   a. In the Display Settings dialog box, select the Shadow Settings page.
   b. Select the Enable Shadow check box.
5. To view the ImGui Test UI, select the Show Imgui Test UI check box. (This check box is only available if you have enabled Shadow Settings.)



II. Scenarios

Whenever you use VR-Forces, you work in the context of a scenario. A scenario consists of everything that needs to be specified to run a VR-Forces simulation. Each scenario has a scenario file (*.scn*) and the following supporting files:

- A scenario objects file (*.oob*).
- A plan file (*.pln*).
- An object map file (*.omp*).
- A terrain database for the front-end and back-end.
- An overlay file (*.ovl*).
- A scripts file (*.spt*). This file contains the scenario scripts, if any.
- A selection groups file (*.sgr*).
- A scenario "extras" file (*.xtr*). This file records hostility relationships and stores spawn pattern templates.
- A saved views file (*.osrx*). If a scenario has captured observer views, they get saved automatically. You can also specify a startup view.
- A settings file that records scenario-specific GUI settings (*.gui_settings*).

By default, scenarios are saved in a compressed zip archive with the extension *.scnx*. If you examine a compressed scenario file in an unzipping application, you will see that it contains the individual files described in the previous paragraphs.

A terrain database does not have to be stored in the same directory as a scenario. It is referenced by the scenario. This lets you use a terrain database, which can be quite large, in more than one scenario, without needing to make multiple copies of it. For more information about terrain databases, please see Section XI, "Terrain".

For details about scenario files, please see Chapter 12, *Scenario Files*.



7. Creating and Running Scenarios

This chapter explains how to create, load, run, and save scenarios.

Creating a Scenario 7-3
- Specifying Multiple Simulation Model Sets 7-8
- Creating Simulation Model Set Configurations 7-9
- Setting the Scenario Starting Date and Time 7-12

Importing Objects into Scenarios 7-14
- Importing (Merging) Scenarios 7-14
- Importing and Exporting MSDL 7-16
- Importing Scenario Objects from CSV Files 7-18
- Importing Airspace Control Orders 7-20

Saving a Scenario 7-21
- Saving a Previously Saved Scenario 7-22
- Saving a Scenario to a New Name or a Different Format 7-24

Loading a Scenario 7-24
- Loading a Recently Loaded Scenario 7-26
- Loading a Scenario from the Command Line 7-26
- Editing Scenario Properties 7-26
- Editing the Scenario Description 7-29
- Sample Scenarios 7-29

Load Balancing Scenarios 7-30
- Load Balancing an Existing Scenario 7-30
- Loading a Load-Balanced Scenario 7-32

Running a Scenario 7-34
- Changing the Simulation Speed 7-34
- Pausing a Scenario 7-36
- Rewinding a Scenario 7-36
- Closing a Scenario 7-37

Reinitializing a Scenario 7-38

Checkpoints and Snapshots 7-38
Saving Checkpoints 7-39
Deleting Checkpoints 7-42
Creating Snapshots 7-42
Clearing the Snapshot List 7-43
Pausing a Scenario Automatically 7-44
Setting the Scenario End Time for a New Scenario 7-44
Setting the Scenario End Time for an Open Scenario 7-44
Running VR-Forces in Batch Mode 7-45
The Batch File 7-45
Creating a Batch File 7-46
Editing a Batch File 7-46
Running VR-Forces in Batch Mode 7-48
Recording Batch Scenarios 7-49
Sending Run and Pause Messages to Simulation Participants 7-50
Recording VR-Forces Simulations with the MAK Data Logger 7-50

7.1. Creating a Scenario

To use VR-Forces to simulate simulation objects, you must load a scenario or create a new one. When you start VR-Forces, the Scenario Startup screen asks you if you want to create or load a scenario (unless you have disabled this feature). Thereafter, to create a scenario, use the procedure in this section. (For details about creating a scenario from the Scenario Startup screen, please see "Starting VR-Forces from the VR-Forces Launcher," on page 4-4.)



If you are using multiple front-ends in the same session, when one of them creates a scenario, the terrains in all other front-ends are closed (unless the terrain is the same as that for the new scenario) and the terrain for the new scenario is loaded.

To create a scenario:

1. Choose **File** → **New Scenario**, or click the New Scenario button () on the File Toolbar. The Initial Scenario Information dialog box opens (Figure 7-1). By default, it opens in *./userData/terrains*, the directory that contains the terrain files shipped with VR-Forces. (For a brief description of the databases supplied with VR-Forces, please see "Terrain Databases Provided with VR-Forces," on page 55-21. For details about terrain files, please see Section XI, "Terrain".)

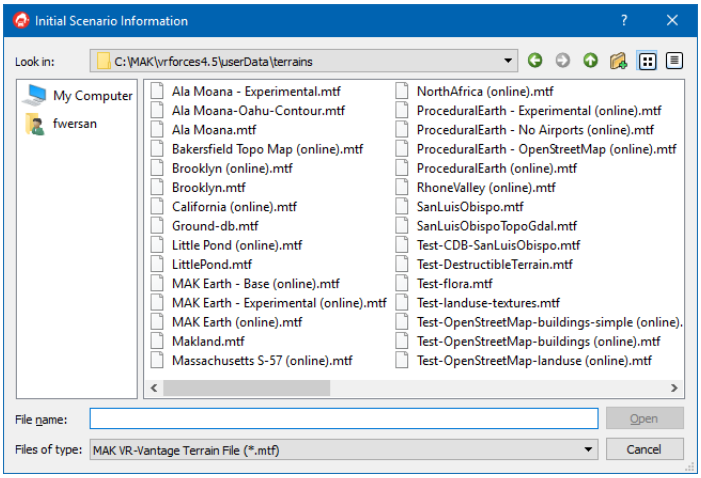


Figure 7-1. Initial Scenario Information dialog box

2. In the file list, select a terrain database. (MTF is the preferred file type.) A database must be a flat earth, geocentric, UTM, or Lambert Conformal Conic database. By default, this terrain is used for both the front-end and back-end.



If you choose an MTF file and an MTD file with the same name exists, the back-end loads the MTD file. (MTD is a legacy format that may be present for older terrains.)

3. Click Open. The New Scenario dialog box opens (Figure 7-2).

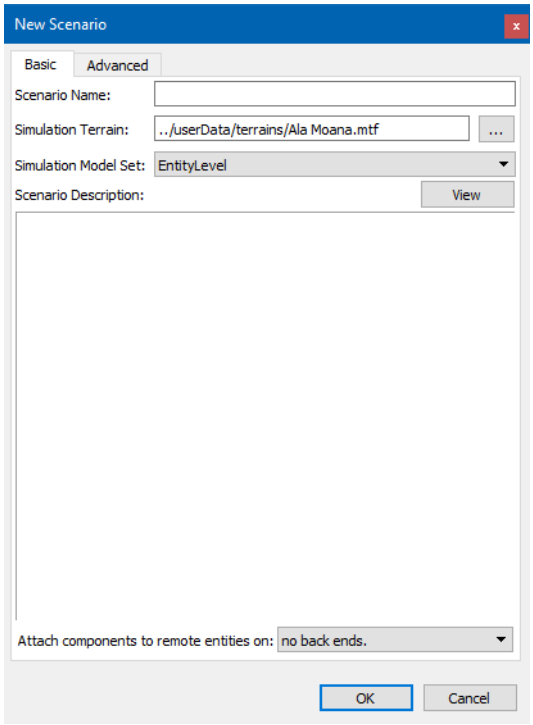


Figure 7-2. New Scenario dialog box

4. Select the simulation model set (SMS) for the type of scenario you want to run – entity-level or aggregate-level.

5. Optionally, set other basic scenario parameters. Table 7-1 describes the basic parameters.

Table 7-1: Basic scenario parameters

Parameter	Description
Scenario Name	Specifies the name of the scenario. This is the name used to identify the scenario when you join a session. If you do not specify a name, the scenario file name is used.
Simulation Terrain	Specifies the terrain to be used by the back-end. By default, this is the terrain you selected in the Initial Scenario Information dialog box. However, you can change it if you want to.
Simulation Model Set	The simulation model set to use. For entity-level scenarios, choose *EntityLevel.sms* or an SMS similar to it. For aggregate-level scenarios, choose *AggregateLevel.sms* or a similar SMS.
Scenario Description	The scenario description lets you provide a description of the scenario. This may be useful to remind you or other VR-Forces users of its purpose or special details. You can enter the description as plain text or as HTML. If you use HTML, when the description is displayed in the Scenario Properties dialog box, it is displayed using the HTML formatting.
Attach Components to Remote Entities On	Specifies whether or not to attach components to remote entities, and if so, the back-ends on which to manage the components that are being attached to remote entities. Unlike other scenario parameters, you can change this value when you load a scenario. For details about configuring components for attachment to remote entities, please see "Attaching Components to Remote Entities," on page 81-2. ! This value is not saved in the scenario file. To attach components in future runs of the scenario, you must specify the appropriate setting each time you load the scenario.

For more information about these parameters, please see "Scenario Parameters," on page 12-4.

6. Optionally, select the Advanced tab (Figure 7-3) and set advanced scenario parameters (overriding the default settings).

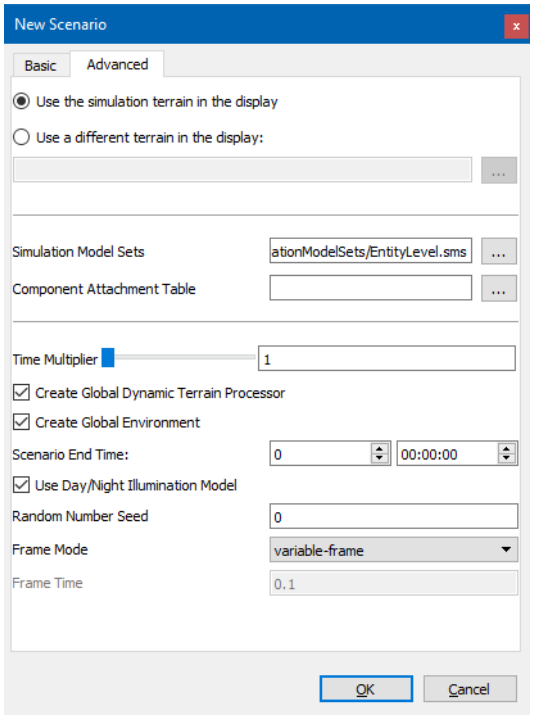


Figure 7-3. New Scenario dialog box, Advanced tab

Table 7-2 describes the parameters you can set.

Table 7-2: Advanced scenario parameters

Parameter	Description
Use the Simulation Terrain in the Display	If selected (the default), the scenario uses the same terrain for the front-end and back-end.
Use a Different Terrain for the Display	If selected, lets you specify the terrain to be used by the front-end.
Simulation Model Sets	Lets you specify simulation model sets other than or in addition to the default SMSs listed on the Basic tab. A scenario must specify at least one simulation model set. For more information, please see "Specifying Multiple Simulation Model Sets," on page 7-8. Default: *EntityLevel.sms*.
Component Attachment Table	If you plan to attach components to remote entities, specify the component attachment table that you want to use. For details about configuring components for attachment to remote entities, please see "Attaching Components to Remote Entities," on page 81-2.
Time Multiplier	Specifies a time multiplier for the scenario. For HLA time-managed federations, this parameter is ignored. Default: 1.0 (real-time).

Table 7-2: Advanced scenario parameters

Parameter	Description
Create Global Dynamic Terrain Processor	If selected, the global dynamic terrain processor monitors the entire terrain for dynamic terrain changes. If disabled, dynamic terrain is not supported unless you create a dynamic terrain area.
Create Global Environment	If selected, global weather is supported. If cleared, the settings on the Environment Conditions dialog box, Weather page have no effect on the scenario. (You can still create weather zones.)
Scenario End Time	The simulation time at which to automatically pause the scenario. For more information, please see "Setting the Scenario End Time for a New Scenario," on page 7-44.
Use Day/Night Illumination Model	If selected, use an ephemeris model in the back-end to calculate illumination. This affects sensors. If cleared, the entire world is fully illuminated regardless of the time of day. For more information, please see "Choosing the Illumination Model," on page 11-5.
Random Number Seed	Specify a random number seed as an integer > = 0.
Frame Mode	Select a frame mode from the list.
Frame Time	Specifies the length of a frame, in seconds. This parameter is not enabled if Frame Mode is variable-frame.

For more information about these parameters, please see:

- "Simulation Time," on page 3-10.
- "Exercise Clock Modes," on page 3-12.
- "Scenario Parameters," on page 12-4.

7. Click OK. The scenario information is displayed and the terrain is displayed in the Plan View (2D) observer mode.
8. Add the simulation objects and tactical graphics that you need for your simulation. For information about adding simulation objects and tactical graphics, please see Chapter 14, *Creating and Placing Objects*.
9. Save the scenario. For details about saving scenarios, please see "Saving a Scenario," on page 7-21.

7.1.1. Specifying Multiple Simulation Model Sets

You can specify more than one simulation model set for a scenario. One strategy for extending VR-Forces is to use an SMS provided with VR-Forces for the majority of objects and to create a separate SMS for customized simulation objects. As you upgrade to newer versions of VR-Forces, you can more easily migrate your customized SMS to the new version because your changes have not been integrated into the default SMS. (You can also include the SMSs provided by VR-Forces in your custom SMS. In that case you would just load the custom SMS and not multiple SMSs. For more information, please see "Simulation Model Sets," on page 67-6.)

If you routinely use more than one SMS, consider creating an SMS configuration. For details, please see "Creating Simulation Model Set Configurations," on page 7-9.

When you load multiple SMSs, if there is any overlap in the objects they specify, the last SMS loaded overrides previous SMSs.

! Do not use an entity-level SMS and an aggregate-level SMS together. The simulation objects from the two SMSs will not be able to interact.

To specify multiple simulation model sets for a scenario:

1. In the New Scenario dialog box, click the Browse button on the Simulation Model Set line. The Simulation Model Set Files dialog box opens. It lists the SMS files to be loaded for this scenario.

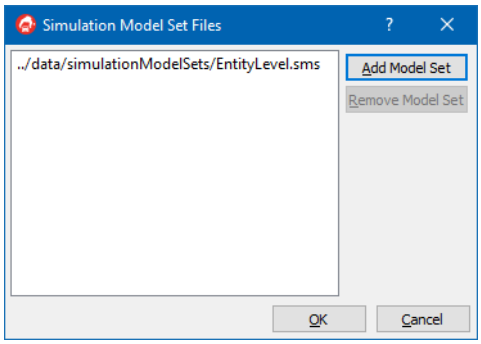


Figure 7-4. Simulation Model Set Files dialog box

2. In the Simulation Model Set Files dialog box, click Add Model Set. A blank line is added to the window (Figure 7-5).

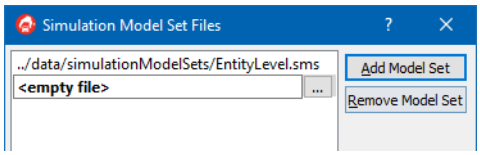


Figure 7-5. Add simulation model set

3. Click the Browse button next to the blank line. The Choose File dialog box opens.
4. Select a simulation model set and click Open. The SMS is added to the text box.
5. Click OK.
6. Complete the New Scenario dialog box.

7.1.2. Creating Simulation Model Set Configurations

When you create a new scenario, you specify an SMS in the Initial Scenario Information dialog box (please see "Creating a Scenario," on page 7-3 for details). The Simulation Model Set list lists all SMSs in *./data/simulationModelSets*. The SMSs are listed in alphabetical order. If you want a particular SMS to be the default SMS you can create an SMS configuration and specify that it be the default. You can create as many SMS configurations as you want. An SMS configuration can have more than one SMS in it. So if you routinely create scenarios that use multiple SMSs, creating a configuration can save you from the necessity of always specifying the additional SMSs on the Advanced tab of the New Scenario dialog box (as described in "Specifying Multiple Simulation Model Sets," on page 7-8).



The Simulation Model Set list in the New Scenario dialog box lists all SMS configurations and all SMSs. It does not duplicate names, so if an SMS configuration has the same name as an SMS, only the SMS configuration is listed.

To create a simulation model set configuration:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 7-6).

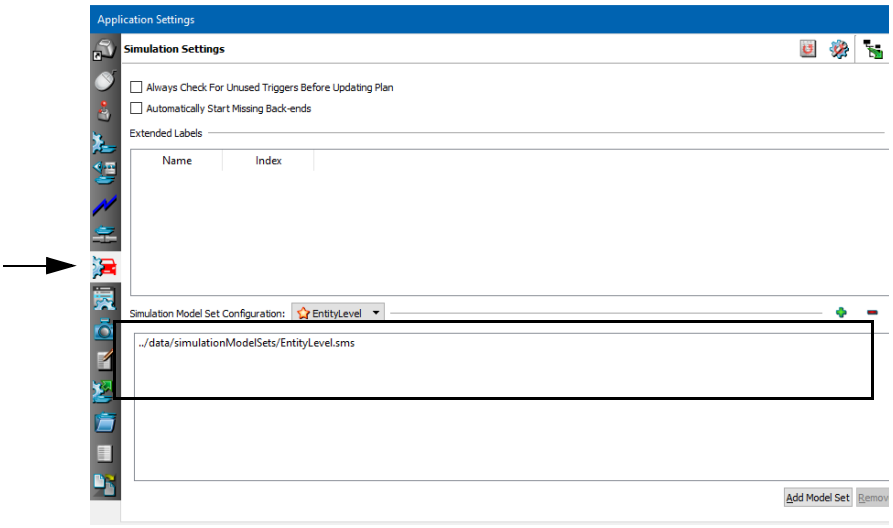


Figure 7-6. Simulation Settings page

3. Click the Add button (+) on the Simulation Model Set Configuration line. The Configuration dialog box opens.
4. Type a name for the configuration.
5. Click OK. The new configuration is added to the list.
6. Click Add Model Set. A line is added to the window.
7. Click the Browse button at the end of the new line. A file chooser dialog box opens.
8. Select the SMS you want to put in this configuration.
9. Click Open. The SMS is added to the list.
10. Optionally, add additional SMSs to this configuration.
11. Click Close.

Specifying the Default SMS Configuration

You can specify that a particular SMS configuration be selected in the Initial Scenario Information dialog box whenever you create a new scenario.



Specifying an SMS as the default SMS automatically removes that designation from any other SMS.

To specify the default SMS configuration:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 7-6).
3. In the Simulation Model Set Configuration list, select the configuration that you want to be the default.
4. Click the Star button (). A star is displayed next to the configuration name in the list, indicating that it is the default configuration.

7.1.3. Setting the Scenario Starting Date and Time

By default, when you create a scenario, the starting simulation date and time is the current system date at 10:00 A.M. local time, where the local time zone is determined by the longitude of the first simulation object that gets created in the scenario.

You can change the date and time at which a scenario starts in either of the following ways:

- A date and time other than the default, but still at the location where the first simulation object is created.
- Use a date, time, and time zone without regard to where the first simulation object is created.

By default, before you create the first simulation object, as you move the observer through the terrain, the time shown on the Time of Day toolbar and in the Date and Time page changes to show you the time at the location of the observer relative to the selected time zone. (By default, the Time of Day toolbar and the Date and Time page show Greenwich Mean Time (GMT).) Furthermore, the sun moves with the observer so that the observer location is always in daylight.



The time zone is an approximation. It does not take into account unusual time zone boundaries or daylight savings time. The time zone is set once. If you create a simulation object and then delete it, the location of that entity still determines the time zone. Furthermore, if for some reason you were to create a scenario and then run it without creating any simulation objects, if at any point you created a simulation object, the starting date and time would be reset to the default value based on the location of that entity.

You can change the starting date and time when you create a scenario.

For more information about simulation time, please see "Setting the Date and Time of Day," on page 11-3.

To set the scenario start date and time using the time zone of the first simulation object:

1. Begin the scenario creation process, as described in "Creating a Scenario," on page 7-3.
2. Before you create the first simulation object, choose **Simulation** → **Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
3. Select the Date and Time page (Figure 7-7).

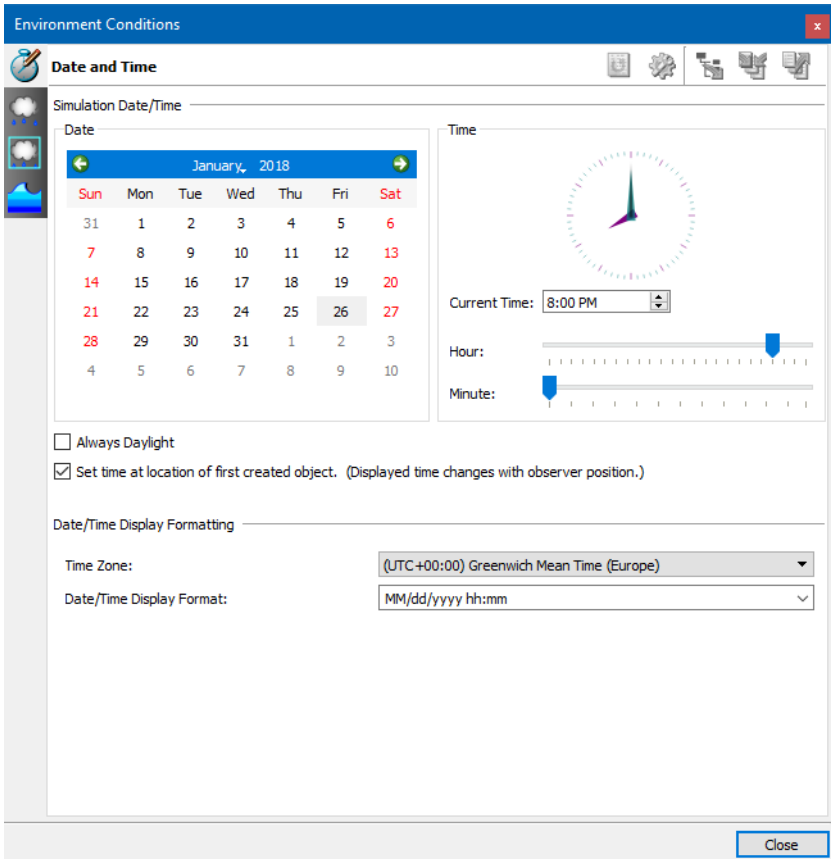


Figure 7-7. Date and Time page

4. Change the date and current time to the desired date and current time.
5. Create a simulation object in the desired location.

To set the scenario start time and time zone without respect to the first simulation object:

1. Begin the scenario creation process, as described in "Creating a Scenario," on page 7-3.
2. Before you create the first simulation object, choose **Simulation → Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
3. Select the Date and Time page (Figure 7-7).
4. Clear the Set Time at Location of First Created Object check box.
5. Set the date, time, and time zone that you want the scenario to start at.
6. Continue the scenario creation process.

7.2. Importing Objects into Scenarios

If you frequently need to create scenarios with the same simulation objects, scenario-level scripts, or other scenario components, consider using one or more of the following strategies:

- Create simulation object groups. Simulation objects groups are heterogeneous groups of simulation objects and tactical graphics that you can add to any scenario. Like the individual simulation objects that you can add to a scenario, they are not tied to a terrain location until you add them. Use them if you want to create a repeatable set of objects that do not fit into a unit hierarchy. For more information, please see "Simulation Object Groups," on page 14-32.
- Use scenario import. If you have a base scenario that you want to use repeatedly, create the scenario, then open it and rename it to preserve the original, or open a new scenario and import the base scenario. You can do this with a fully realized scenario or a scenario layout that you want to add to other scenarios. For example, you might want to create a town with background traffic for repeated use with different training problems added to it. Since this is a scenario, it is tied to the terrain location you use when you create it. Alternatively, you might have a set of scripts that you do not want to promote to system scripts. You can save them in a scenario and import the scenario into the scenarios for which you want to use the tasks. For more information, please see "Importing (Merging) Scenarios," on page 7-14.
- Import MSDL. If you have a MSDL entity laydown, you can import that into different scenarios as needed. For more information, please see "Importing and Exporting MSDL," on page 7-16.
- Import point, linear, or areal objects from a CSV file. For details, please see "Importing Scenario Objects from CSV Files," on page 7-18.
- Export an Order of Battle and import it into other scenarios. For details, please see Chapter 20, *Creating and Using an Order of Battle*.
- Export tactical graphics to shapefiles. For details, please see "Exporting Tactical Graphics as Shapefiles," on page 39-7.

7.2.1. Importing (Merging) Scenarios

VR-Forces users often assign scenario development to multiple persons or teams. Then they merge the separately-developed scenarios into a master scenario. You can easily merge scenarios by importing them. When you import a scenario, you can choose to rename simulation objects to avoid duplicate names, to import scenario events, and to create a selection group for the imported simulation objects.

! If you import a scenario that was saved (checkpointed) with a simulation time that is different from the current time of the scenario into which you are importing it, there may be unexpected behaviors due to timing issues. For example, an entity executing a Wait Duration task may wait for a longer or shorter time than you expect.

To import a scenario:

1. Open a scenario.
2. Choose **File** → **Import VR-Forces Scenario**. The Import Scenario dialog box opens (Figure 7-8).

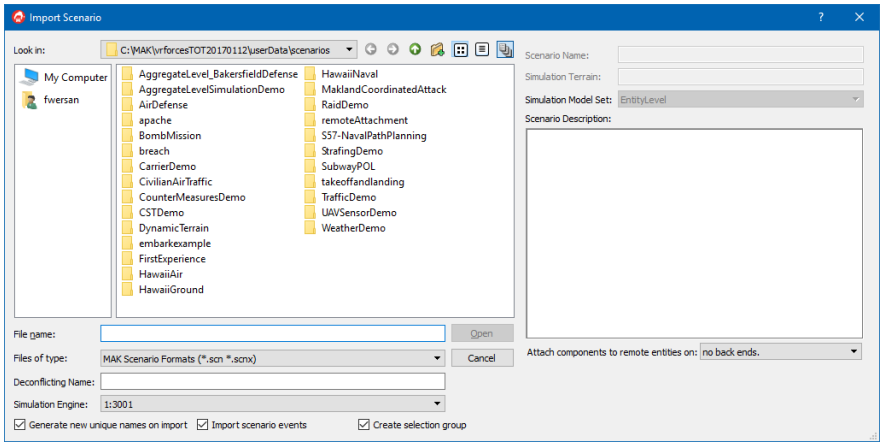


Figure 7-8. Import Scenario dialog box

3. Select the scenario that you want to import.
4. Optionally, on the Deconflicting Name box, specify a text string that VR-Forces should add to object names in the imported scenario that match names in the base scenario.
5. Optionally, to allow duplicate names, clear the Generate New Unique Names on Import check box.
6. Optionally, specify whether or not to import scenario events.
7. Optionally, specify whether or not to create a selection group.

7.2.2. Importing and Exporting MSDL

Military Scenario Definition Language (MSDL) is a SISO standard for describing military scenarios using an XML format. It allows for interchange of scenario descriptions between simulation applications and systems that use different internal formats. You can import MSDL files into a VR-Forces scenario.

VR-Forces does not import all of the information that can be in an MSDL file. It imports and exports:

- Simulation objects and their locations. Locations are always exported in geocentric coordinates.
- Force relationships and hostility relationships.
- Unit formations.
- Tactical graphics.

To import MSDL:

1. Create or load a scenario.
2. Choose **File** → **Import Scenario Objects**. The Import Scenario File dialog box opens.
3. In the Files of Type box, set the filter to MSDL Scenario File (*.xml).
4. Select the MSDL file that you want to import.
5. Click Open.

Exporting MSDL

To export a scenario to MSDL:

1. Create or load a scenario.
2. Choose **File** → **Export Scenario Laydown**. The Export Scenario dialog box opens.
3. In the Save As Type box, set the filter to MSDL Scenario File (*.xml).
4. Select the directory in which you want to save the file.
5. Type a name for the file.
6. Click Save. If VR-Forces is not sure how to export an entity type to MSDL, the Export MSDL Scenario dialog box opens (Figure 7-9).

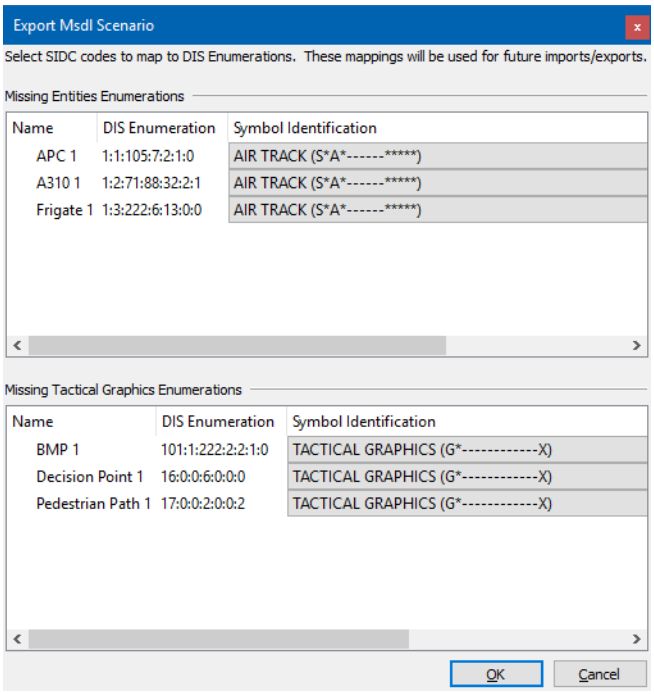


Figure 7-9. Export MSDL Scenario dialog box

7. For each object type listed in the Export MSDL Scenario dialog box:
   a. In the Symbol Identification column, click the value. A list of possible identification codes is displayed. (These codes are from the MSDL specification. You will need to familiarize yourself with the specification to fully understand them.)
   b. Select the proper identification.
8. Click OK.

7.2.3. Importing Scenario Objects from CSV Files

Some VR-Forces users like to plot tactical graphics and simulation object locations off line and then import them into a scenario. VR-Forces supports importation of points, lines, and areas from comma separated values (CSV) files. A CSV file can contain only one kind of object, as follows:

- One area.
- One route.
- One or more point objects. Points can represent tactical graphics, such as waypoints, or individual entities.

VR-Forces includes some example CSV files in *./userData/csvLaydown*.



You cannot export simulation objects to a CSV file. To make the objects in a scenario available to other scenarios, try some of the strategies in "Importing Objects into Scenarios," on page 7-14.

To import scenario objects from a CSV file:

1. Choose **File** → **Import Scenario Objects**. The Import Scenario File dialog box opens.
2. In the Files of Type box, set the filter to CSV Scenario File (*.csv).
3. Select the CSV file that you want to import.
4. Click Open. The Import Object Options dialog box opens. If you selected a CSV file that has point data, the points are listed in the window

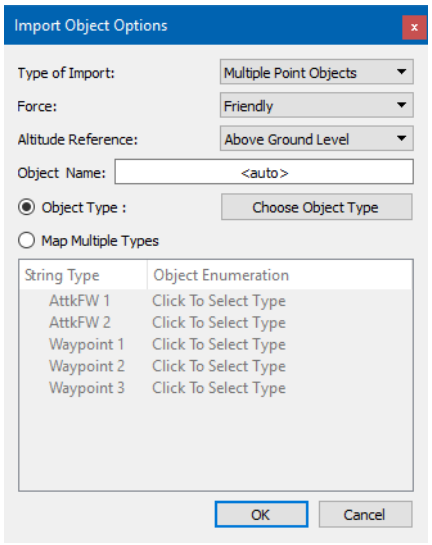


Figure 7-10. Import Object Options dialog box

5. In the Type of Import, select the type of data you are importing.

6. Optionally, specify a force. All imported objects are imported with the same force.
7. Optionally specify the altitude of objects as AGL (above ground level) or ASL (above sea level).
8. Optionally, in the Object Name box specify a name for the object. The default is to use VR-Forces naming conventions to automatically assign a name.
9. If you are importing an area or route:
   a. Select the Object Type option.
   b. Click Choose Object Type. The Entity Type dialog box opens. It lists objects of the type you selected in the Type of Import list. (The layout of the dialog box is similar to the object creation palettes.)
   c. Select the specific type of area or route that you want to use for the imported object. The Entity Type dialog box closes.

   If you are importing a list of waypoints:
   a. Select the Map Multiple Types option.
   b. Click a point in the list. The Entity Type dialog box opens.
   c. Select the specific type of point or entity that you want to use for the imported object. The Entity Type dialog box closes. The Import Object Options lists the object type enumeration for the object.

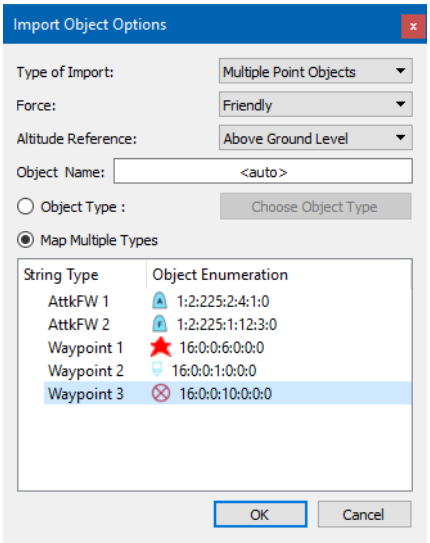


Figure 7-11. Import Object Options dialog box

   d. Repeat this process for each object in the list.
10. Click OK.

7.2.4. Importing Airspace Control Orders

An airspace control order (ACO) defines and establishes an airspace for military operations. It is can be part of an air tasking order (ATO) or a separate document. VR-Forces can import ACOs that are separate documents.

VR-Forces includes an example ACO file, *./userData/aco/exampleMontereyACO.aco*. The file's format is documented in *./userData/aco/acoFormattingReadme.txt*.

To import an ACO:

1. Choose **File** → **Import Scenario Objects**. The Import Scenario File dialog box opens.
2. In the Files of Type box, set the filter to ACO Scenario File (*.aco).
3. Select the file that you want to import.
4. Click Open.

7.3. Saving a Scenario

The first time you save a scenario, VR-Forces saves the path of the terrain database and other scenario configuration data to a scenario file (extension *.scn*). (Paths are saved relative to the *./bin64* directory.) This information usually does not change between runs of a scenario. Every time you save a scenario, VR-Forces saves the list of simulation objects and tactical graphics, the plans, and object mapping information to individual files that have the same name as the scenario, but with file extensions that identify the information in the file (*.oob*, *.pln*, *.xtr*, *.omp*, and so on). By default, all these files are saved to a compressed archive with the extension *.scnx*. However, you can save them as individual files, as explained in this procedure.

- If you want to keep the initial state of a scenario (simulation time = 0 and no simulation has taken place), so that you can repeatedly run a simulation from the beginning, save the scenario when you have finished creating it, but before you run it. Then, do not save the scenario again under the same name (unless you specifically want to change the original state of the scenario). (If you forget to save a new scenario, you can rewind to the initial snapshot and save it.)
- If you load an alternate terrain in a front-end when it joins a session, the alternate terrain is not saved as part of the scenario.

When you save a scenario, whether during the course of setting it up or during a simulation, all object state information gets saved. Essentially, every save is a checkpoint. When you run a saved simulation, all simulation objects resume whatever they were doing when the scenario was saved.



If simulation objects are executing scripted tasks, some information regarding the tasks is not saved. For details, please see "Limitations for Checkpointing Scripts," on page 32-11.

Save behavior depends on when you save a scenario, as follows:

- If you save a previously saved scenario after it has started (whether it is paused or running), it is not saved to the original scenario file. It is saved to a checkpoint that is named using the convention *scenario_name*_at_time_-*time*.scn. This ensures that you cannot inadvertently overwrite the original scenario file. To overwrite the original scenario, you must use Save Scenario As. For more information, please see "Saving Checkpoints," on page 7-39 and "Saving a Previously Saved Scenario," on page 7-22.
- If you run a new scenario without saving it and save it at some simulation time greater than zero, it is saved using whatever name you provide. The time is not included in the scenario name.

To save a new scenario:

1. Choose **File** → **Save Scenario** or click the Save button (💾) on the File Toolbar. The Save Scenario dialog box opens.
2. Select the directory in which you want to save the scenario.

> **i** If you create a folder in the Save dialog box and it has a space at the end of the name, Windows does not recognize the existence of the folder. If you try to delete it, the system may blue screen. Running scandisk can solve this problem.

3. Give the scenario a name.
4. By default, scenarios are saved in a compressed archive (**.scnx*). If you want to save a scenario as individual files, in the File of Type list, select MAK Scenario Format (*.scn).
5. Click Save. All scenario files are saved with the new name.

> **i** If you save a remapped scenario, you are prompted to save the scenario with the original mappings or with the remappings.

7.3.1. Saving a Previously Saved Scenario

In most computer applications, when you save a file that has been saved previously, the current version of the file simply overwrites the previously saved version. However, when VR-Forces saves a scenario, the save action depends on the status of the scenario. VR-Forces tracks the simulation time of the scenario. When you save a scenario:

- If a version of the scenario exists with the current simulation time, VR-Forces displays a message indicating that a version of the scenario at that simulation time exists and asks if you want to overwrite it.
- If there is no version with the current simulation time, VR-Forces creates a new checkpoint at the current simulation time, using the checkpoint naming convention.

This behavior ensures that if you change a scenario after you have started a simulation, you will never inadvertently overwrite the original version or another checkpoint.

➤ To save a previously saved scenario, choose **File** → **Save Scenario**.

Saving a Load Balanced Scenario

When you load a load-balanced scenario and do not have all of the required back-ends available, VR-Forces prompts you to allow it to remap objects to the available back-ends. Then it loads the scenario.

When you save a scenario that was originally load balanced, VR-Forces prompts you to either save the original mappings or to save the scenario with the back-end mappings used to load it in the current session.

If you save the original mappings, all objects that were part of the scenario when it was loaded maintain their previous mappings. Any objects that you created during the current session are mapped to the back-ends that they were created on in the current session.

7.3.2. Saving a Scenario to a New Name or a Different Format

You can save a scenario to a different name. You can also save it to a different format. That is, you can save a scenario that was originally saved as a compressed archive to individual files or save individual files to a compressed archive.



If you save a scenario to a different format and you use the same name, VR-Forces does not replace the previously saved files with the new format. The two sets of files live side-by-side. Therefore we recommend that you give the newly saved scenario a different name or remove the old files to avoid confusion between different versions of the same scenario.

To save a scenario to a new name:

1. Choose **File** → **Save Scenario As**. The Save Scenario dialog box opens.
2. Select the directory in which you want to save the scenario.
3. Give the scenario a name.
4. If you are changing the format, select an option in the Files of Type list.
5. Click Save. The scenario is saved to the new name.

7.4. Loading a Scenario

After you have created and saved a scenario, you can load it and run it on any platform supported by the version of VR-Forces in which you created it. Most scenarios are forward compatible. Please see *VR-Forces Release Notes* for compatibility issues.



- If you are running multiple front-ends in the same session, when you load a scenario, any terrains that are open on other front-ends are closed and the terrain for the newly loaded scenario is opened.
- When you load a scenario that uses multiple back-ends, if any of the required back-ends are not running, VR-Forces prompts you to remap the objects in the scenario. The Remap Missing Back-ends dialog box lists the unavailable back-ends and the available back-ends. You can specify which back-ends to map objects to or you can select "map all to" and map all objects to one back-end. For more information about remapping back-ends, please see "Remapping Back-ends," on page 3-8.

To load a scenario:

1. Choose **File** → **Load Scenario** or click the Open Scenario button ( ) on the File Toolbar. The Load Scenario dialog box opens (Figure 7-12).

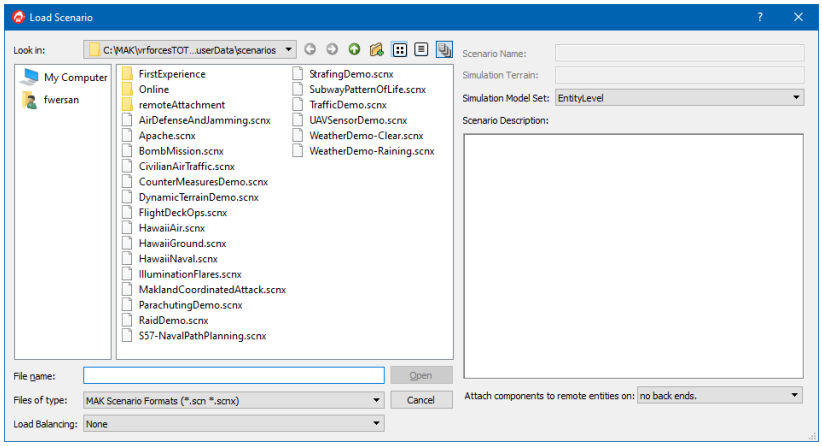


Figure 7-12. Load Scenario dialog box

2. Select a scenario to load. Basic information about the scenario is displayed (Figure 7-13). Scenarios are files with the suffix *.scn* or *.scnx*. Sample scenarios are located in the *./userData/scenarios* directory. (Batch scenarios, described in "Running VR-Forces in Batch Mode," on page 7-45, have the suffix *.bsn*.)

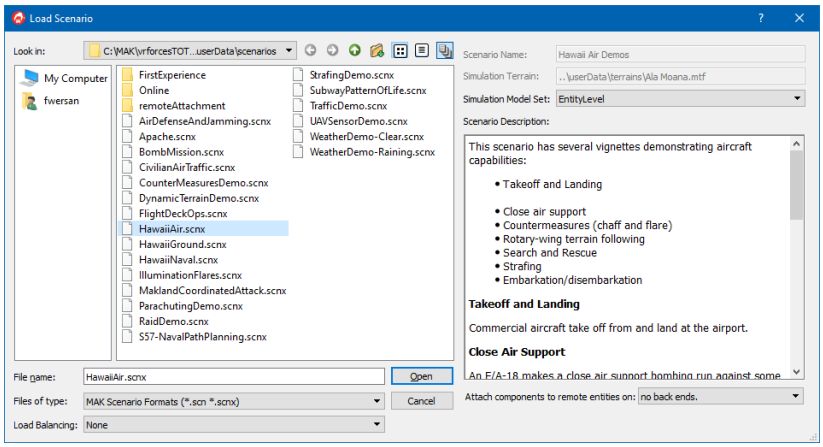


Figure 7-13. Load Scenario dialog box

3. Optionally, specify whether or not to allow management of components on remote entities by selecting an option in the Attach Components to Remote Entities On list. For details about configuring components on remote entities, please see "Attaching Components to Remote Entities," on page 81-2.
4. Optionally, select a load balancing option.

5. Click Open.

7.4.1. Loading a Recently Loaded Scenario

VR-Forces keeps a list of the scenarios you open. You can quickly open a scenario from the list.

➤ To open a recently used scenario, choose **File** → **Recent Scenarios** → *scenario_name*.

7.4.2. Loading a Scenario from the Command Line

If you start VR-Forces from the command line, you can specify a scenario to load. You can specify the scenario in the back-end or front-end. The other executable will load it as soon as it starts up.

To load a scenario when you start the back-end, use the `-L` command-line option, for example:

```
vrfSimDIS -s 1 -a 3001 -L "../userData/scenarios/mySce-
  nario.scnx"
```

To load a scenario when you start the front-end, use the `--scenarioFile` command-line option, for example:

```
vrfGui --scenarioFile "../userData/scenarios/mySce-
  nario.scnx" --dis -s 1 -a 3000
```

7.4.3. Editing Scenario Properties

When you create a scenario, you specify a variety of parameters (or accept default values) and, optionally, specify a name and description for the scenario. If a scenario has a description, when you load the scenario, the Scenario Information dialog box opens automatically and displays the scenario properties. (You can disable this behavior.) You cannot edit properties in the Scenario Information dialog box.

You can display scenario properties at any time while a scenario is loaded. You can edit the scenario properties. The following changes take place immediately:

- Scenario description.
- Scenario name.
- Time multiplier.
- Create global environment.
- Create global dynamic terrain processor.
- Scenario end time.

The following changes do not take effect until you save the scenario and reload it. If you edit one of these properties and have not yet saved the scenario, if you open the dialog box again, an icon is displayed to alert you to save the scenario.

- Simulation terrain.
- Use a different terrain in the display.
- Simulation model sets.
- Component attachment table.
- Random number seed.
- Frame mode.
- Frame time.

To edit scenario properties:

1. Choose **File** → **Scenario Properties**. The Scenario Properties dialog box opens (Figure 7-14).

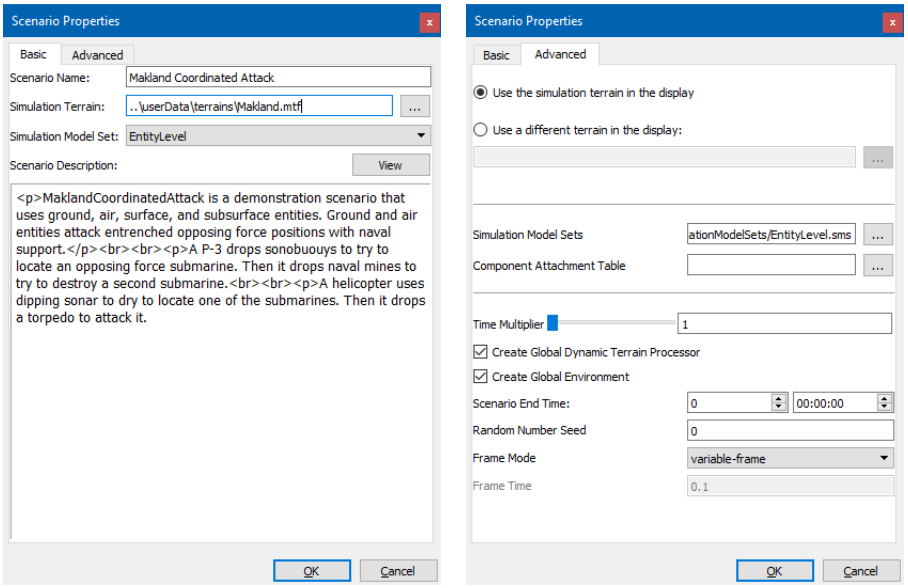


Figure 7-14. Scenario Properties dialog box

2. Edit properties as desired. (For descriptions of the properties, please see Table 7-1 and Table 7-2.)

Enabling and Disabling Display of the Scenario Description

By default, the scenario description gets displayed when you open a scenario. You can turn it off in the Scenario Information window and you can enable and disable its display in the Application Settings dialog box.

To disable display of scenario information when a scenario is loaded, use one of the following methods:

- When the Scenario Information window opens, select the Never Show Again check box.
- On the Application Settings dialog box, GUI Settings page, clear the Show Scenario Information Dialog check box (Figure 7-15).

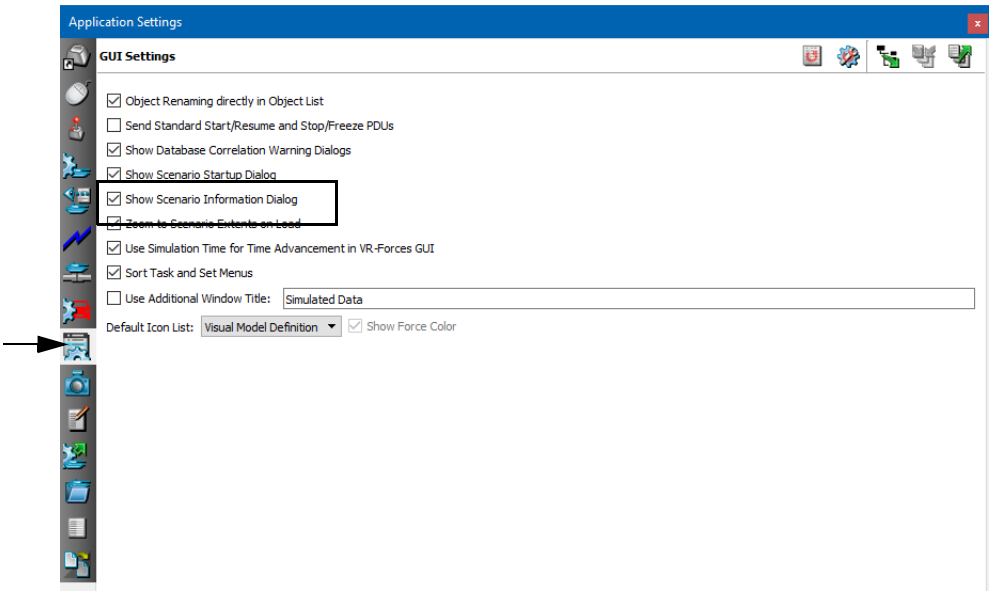


Figure 7-15. GUI Settings page

➤ To enable the Scenario Information dialog box, on the Application Settings dialog box, GUI Settings page, select the Show Scenario Information Dialog check box.

7.4.4. Editing the Scenario Description

You can edit a scenario's description in the Scenario Properties dialog box.

To edit a scenario's description:

1. Choose **File** → **Scenario Properties**. The Scenario Properties dialog box opens (Figure 7-16).

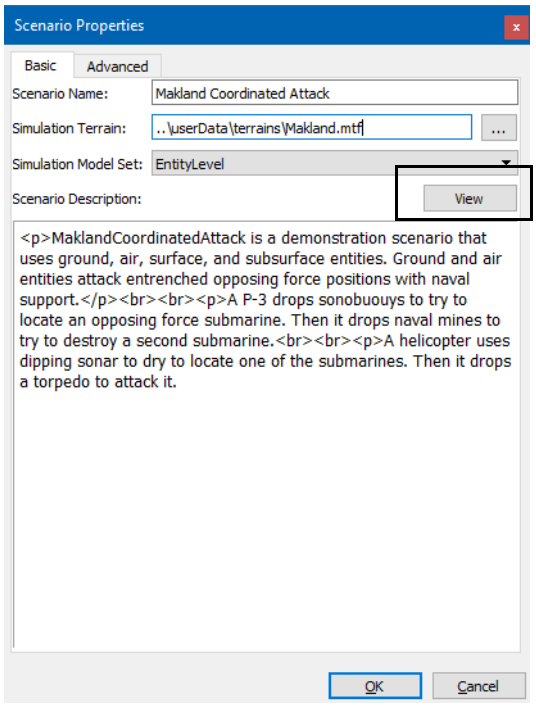


Figure 7-16. Scenario Properties

2. Edit the text.
3. To see how the text will be displayed, click View.
4. Click OK.

7.4.5. Sample Scenarios

VR-Forces includes several sample scenarios. *VR-Forces First Experience Guide* has three tutorials for scenario creation.

7.5. Load Balancing Scenarios

One way to improve the performance of a scenario is to spread simulation of objects across multiple back-ends. This is called load balancing. VR-Forces supports even distribution (done automatically by VR-Forces) and manual load balancing.

You can load balance a scenario when you create it. You can also load balance a scenario after you create it.

! If you are using interest management to support large simulation object counts, do not enable load balancing. It will negate the assignment of simulation objects to specific back-ends that is part of designing a scenario for interest management.

To load-balance a new scenario:

1. Start as many back-ends as you want to use in the scenario, as described in "Starting VR-Forces from the VR-Forces Launcher," on page 4-4 and "Starting Independent VR-Forces Executables," on page 4-9.
2. Create the new scenario, as described in "Creating a Scenario," on page 7-3.
3. Before you create an object, select the back-end on which you want it to be simulated, as described in "Selecting the Object to Create on an Object Palette," on page 14-4 and "Selecting the Object to Create on the Create Menu," on page 14-7.

7.5.1. Load Balancing an Existing Scenario

You can change the object mappings for a scenario (load balancing) when you load it. You cannot change mapping after the scenario loads.

To load-balance an existing scenario:

1. Start as many back-ends as you want to use in the scenario.
2. Choose **File** → **Load Scenario**. The Load Scenario dialog box opens.
3. In the Load Scenario dialog box, select an option from the Load Balancing list (Figure 7-17).

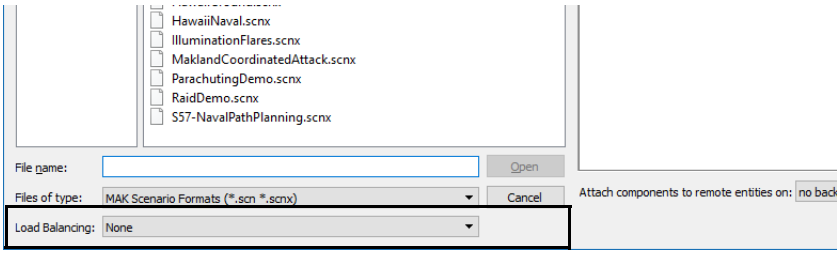


Figure 7-17. Load Balancing list

If you select Even Distribution, VR-Forces automatically distributes objects across the available back-ends. If you choose Manual load balancing, the Object Mapping dialog box opens (Figure 7-18).

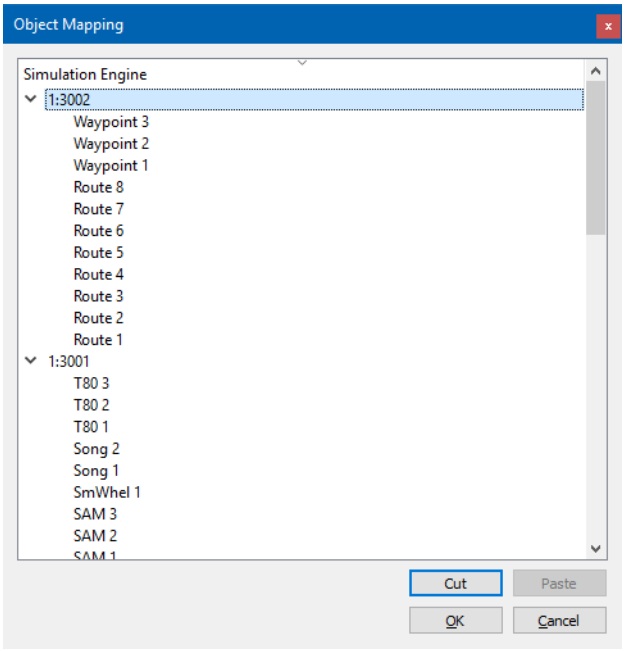


Figure 7-18. Object Mapping dialog box

To load balance the scenario:

a. In the Object Mapping dialog box, expand the list for the back-end from which you want to distribute objects.

b. Redistribute objects by dragging them from a source back-end to a target back-end, or by cutting them from the source back-end and pasting them onto the target back-end. You can view the new distribution of objects by expanding the list for the target back-end.

c. Click OK.

4. In the Load Scenario dialog box, click OK. The scenario loads. If you open the Information dialog box for an object, the title bar displays the back-end on which it is simulated.

5. To preserve the new object mappings, save the scenario. When you save the scenario, you are prompted to save the original mappings. Click No to save the new load balanced mappings.

7.5.2. Loading a Load-Balanced Scenario

When you load-balance a scenario, it saves the mapping between objects and simulation engines in the scenario's object mapping file. When you load the scenario, VR-Forces looks for the listed simulation engines based on their application numbers. If it does not find the required simulation engines, it prompts you for instructions on how to proceed (Figure 7-19). You can start the missing simulation engines, remap objects to available simulation engines, or if you are using more than two simulation engines, do a combination of both.

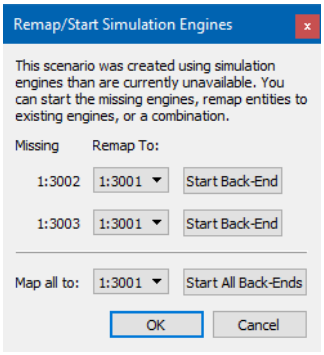


Figure 7-19. Remap/Start Simulation Engines dialog box

If you start a simulation engine as part of the process of loading a scenario, it is started in combined mode and will automatically be closed when you shut down the front-end.

To load a scenario that uses multiple simulation engines:

1. Optionally, start all the required simulation engines and at least one front-end. One simulation engine can be started in combined mode with the front-end.
2. Load the scenario. If the scenario cannot find all of the required simulation engines, it prompts you to start them or remap objects to the available back-ends (Figure 7-19). For each unavailable simulation engine, you can remap objects to an available simulation engine or start the simulation engine. You have the following options:
   - To start a simulation engine, click the Start Back-End button for the Missing simulation engine.

     If there is more than one unavailable simulation engine, as you start them individually, the dialog box redisplays to show only those that are still unavailable. If you start all of the required simulation engines, the dialog box closes and the scenario loads.
   - To start all missing simulation engines, click the Start All Back-Ends button.
   - To remap objects for a missing simulation engine, select an available simulation engine from the Remap To list for the Missing simulation engine.
   - To remap all objects to an available simulation engine, select an available simulation engine from the Map All To list.

     When you select a simulation engine in the Map All To list, it sets the same simulation engine in the lists for each specific missing simulation engine. Put another way, when you click OK to close the dialog box, VR-Forces uses the remapping specified for each missing simulation engine. It ignores the setting for Map All To. Map All To is just a quick way to set all missing simulation engines to the same available one.
3. If you decided to remap objects to an available simulation engine, click OK. VR-Forces remaps the objects and the scenario loads.

7.6. Running a Scenario

If you want to use VR-Forces to simply observe a remote simulation, you can start it, open a terrain database, and wait for it to pick up messages from the network. However, you will probably use VR-Forces to run scenarios that have locally simulated simulation objects.

When you load a scenario, it opens in the paused state unless you start with the `-L` and `-r` back-end command line options.



When you run a scenario, VR-Forces saves a snapshot of the state of the scenario in memory. However, it does not save the scenario to disk. We recommend that you save new or edited scenarios before you run them. If you choose not to, it is your responsibility to remember to rewind to the initial scenario snapshot if you want to save the starting point of the scenario.

➤ To run a scenario, choose **Simulation** → **Run Scenario**, or click the Run button (▶) on the Simulation Control Toolbar (Figure 7-20).

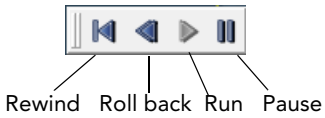


Figure 7-20. Simulation Control Toolbar

7.6.1. Changing the Simulation Speed

You can set the speed of a simulation to run faster or slower than real-time. If you are running VR-Forces with time management, the Simulation Time Scale Toolbar has no effect on simulation speed.

If you increase the speed at which a scenario runs, the frame rate is reduced and performance of models may degrade.

➤ To change the speed of a simulation, click an option on the Simulation Time Scale Toolbar (Figure 7-21).

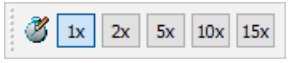


Figure 7-21. Simulation Time Scale Toolbar

Configuring the Simulation Time Scale Toolbar

By default, the range for the Simulation Time Scale Toolbar is 1 to 15. You can change the range. To change the range, edit the following lines in *./appData/settings/vrfGui/default_GuiSettings.grsx*:

```
<myTimescaleLow>1</myTimescaleLow>
<myTimescaleHigh>15</myTimescaleHigh>
```

To change the options on the time scale buttons, edit the myTimescaleButtons elements:

```
<myTimescaleButtons>1;2;5;10;15</myTimescaleButtons>
```

If you want the Simulation Time Scale Toolbar to use a slider as in VR-Forces 4.6.1 and earlier, instead of discrete buttons, set the myTimescaleButtonMode to 0:

```
<myTimescaleButtonMode>0</myTimescaleButtonMode>
```

Changing Simulation Parameters when Increasing the Time Scale

When you increase the time scale in scenarios that have many entities, particularly if they are fast moving entities such as aircraft, simulation quality may degrade. VR-Forces lets you change the dead-reckoning algorithm, frame mode, and frame time automatically to settings that may improve simulation quality.

These settings are configured in *./appData/settings/vrfSim/fastForwardSettings.mtl*. The comments in the file explain the acceptable values for each parameter. The following is an example configuration:

```
(fast-forward-settings
  (fast-forward-entry
    (play-speed 3)
    (settings
      (dead-reckon-algorithm Static)
      (frame-mode 2)            ;; fixed frame
      (frame-time 1.0)
    )
  )
  (fast-forward-entry
    (play-speed 6)
    (settings
      (dead-reckon-algorithm Static)
      (frame-mode 2)            ;; fixed frame
      (frame-time 2.0)
    )
  )
)
```

Each fast-forward entry specifies a time scale speed at which it takes effect. When you change the time scale, VR-Forces finds the highest play-speed specified in this file that is less than the current speed and updates the settings accordingly. If the current speed is less than the lowest value, default settings are used.

7.6.2. Pausing a Scenario

When you pause a simulation, local simulation objects stop executing plans and tasks. Display of remote entity activity continues as it is received over the network.

➤ To pause a simulation, choose **Simulation** → **Pause Scenario**, or click the Pause button ( ) on the Simulation Control Toolbar (Figure 7-20).

7.6.3. Rewinding a Scenario

If you are running a scenario and want to start it over again, you can rewind it. You can rewind a scenario to its initial state, which is captured as a snapshot, and you can roll back to any snapshot or checkpoint (if snapshots and checkpoints are enabled).

When you rewind a scenario, all back-ends are set to the paused state.



An initial snapshot is taken of a scenario when you play it even if periodic snapshots is not enabled. The initial snapshot is used for rewinding.

➤ To rewind a scenario, choose **Simulation** → **Rewind to Scenario Start**, or click the Rewind Scenario to the Beginning button ( ) on the Simulation Control Toolbar (Figure 7-20).

If you do not pause a scenario before you rewind it, it starts playing again immediately.

Rolling Back to a Snapshot or Checkpoint

You can roll back to any of the available snapshots or checkpoints. When you roll back to a snapshot, VR-Forces loads the scenario as it existed at the point of that snapshot. When you run the snapshot, all snapshots from a later time than the one you rolled back to are deleted. Older snapshots are kept in memory. New snapshots are taken starting from the loaded snapshot.

Once snapshots are deleted based on the maximum number of snapshots being reached, you cannot roll back to them. If you have saved periodic checkpoints, you can load them using the Load Scenario command on the File menu.

For details about checkpoints and snapshots, please see "Checkpoints and Snapshots," on page 7-38.

To roll back to a snapshot or checkpoint:

1. Choose **File** → **Scenario Snapshots**, or on the Checkpoint Settings page, click Show Snapshots. The Scenario Snapshots dialog box opens (Figure 7-22).
2. Select the snapshot or checkpoint you want to roll back to.

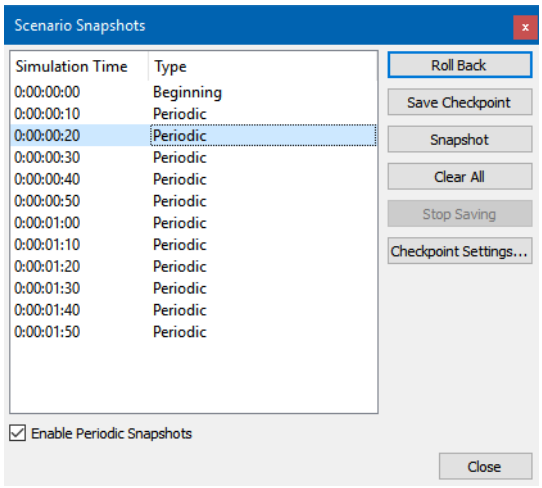


Figure 7-22. Scenario Snapshots dialog box

3. Click Roll Back.

You can also display a list of snapshots and checkpoints by clicking the Snapshots button (◀) on the Simulation Control Toolbar (Figure 7-23).

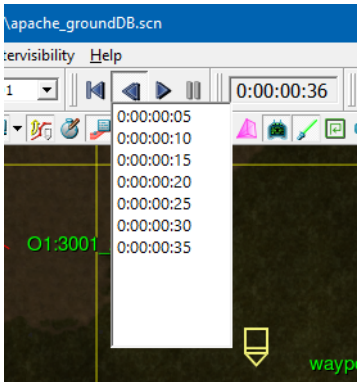


Figure 7-23. Snapshot list

7.6.4. Closing a Scenario

When you close a scenario, the terrain stays loaded. Checkpoints and snapshots remain in memory until a new scenario is loaded. Therefore, if you click Rewind after closing a scenario, the scenario opens again.

➤ To close a scenario, choose **File** → **Close Scenario**, or click the Close Scenario button (✖) on the File Toolbar.

7.7. Reinitializing a Scenario

Sometimes when you load a scenario that was created in a previous version of VR-Forces, entities do not behave as they did previously due to changes to systems. Usually, the way to fix this problem is to delete the objects, create new instances of the object type, and reconstruct their plans or tasks. Unfortunately, when you create a new object, it is assigned a unique ID. Since the unique ID for the newly created objects will be different from those of the deleted objects, any plans or tasks that referenced the deleted objects will not work, because they depend on the unique ID to identify the object that they are referencing. If you need to recreate an object and maintain its unique ID, you can reinitialize a scenario.

When you reinitialize a scenario, VR-Forces deletes all simulation objects and creates them again with the same unique IDs they originally had. This preserves task and plan relationships. However, this process does not preserve the entire state of entities. If you reinitialize a checkpointed scenario, it may lose information that is saved as part of a system. Plans are restarted, so the current task might be cleared. So there may be cases where reinitializing causes more problems than it solves.

Given this consideration, you should only use this option for scenarios that are at zero simulation time and only if you are sure that it is the only option. You may want to save a backup of the scenario in case you want to revert to it and try fixing it by manually deleting the offending object and adjusting other aspects of the scenario as needed.

To reinitialize a scenario:

1. Optionally, save a backup of the scenario in case you want to revert to the original state.
2. Choose **File** → **Reinitialize Scenario**.

7.8. Checkpoints and Snapshots

As noted in "Saving a Scenario," on page 7-21, each time you save a scenario, you are essentially creating a checkpoint. A checkpoint saves the state of a scenario to disk and allows you to run the scenario starting from the point at which it was saved. Since a checkpoint is saved to disk, you can return to it at any time. However, the time spent saving to disk may briefly interrupt scenario progress.

Sometimes VR-Forces users want to be able to quickly return to an earlier point in a simulation without the processing overhead required to save checkpoints. As an alternative or complement to checkpoints, you can save snapshots.

Snapshots save the state of a scenario to memory. You can quickly roll back to any saved snapshot and start running the scenario from that point. When you roll back to a snapshot, all snapshots that were taken at a later time than the rollback point get discarded and new snapshots get saved.

When you load a scenario, any snapshots in memory get discarded.

You can configure the frequency with which snapshots are saved and how many to keep in memory at a time. You can save snapshots as checkpoints.

7.8.1. Saving Checkpoints

You can save checkpoints in the following ways:

- Save the scenario.
- Save a snapshot as a checkpoint.
- Enable periodic checkpointing.

Checkpoints are saved to the same directory as the base scenario unless you specify otherwise. If periodic checkpoints are enabled, the Status bar displays the time until the next checkpoint.



If simulation objects are executing scripted tasks, some information regarding the tasks is not saved. For details, please see "Limitations for Checkpointing Scripts," on page 32-11.

When you run a checkpoint, simulation objects resume the tasks that they were executing at the time the checkpoint was saved. If detonations were scheduled or if missiles were in flight, they resume their former state.

Saving an Individual Checkpoint

Whenever you save a scenario, it creates a checkpoint.

➤ To save an individual checkpoint, choose **File** → **Save Scenario**.

Saving Snapshots as Checkpoints

If you enable periodic snapshots, you can save the snapshots as checkpoints.

To save a snapshot as a checkpoint:

1. Choose **File** → **Scenario Snapshots**. The Scenario Snapshots dialog box opens (Figure 7-24). It lists all of the snapshots available.

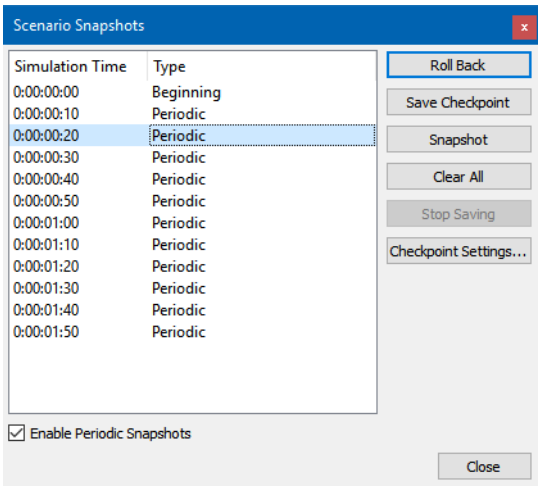


Figure 7-24. Scenario Snapshots dialog box

2. Select the snapshots that you want to save as checkpoints.
3. Click Save Checkpoint. The snapshots are saved to checkpoints.

Scheduling Periodic Checkpoints

You can configure VR-Forces to save checkpoints at regular intervals. When VR-Forces saves a checkpoint, it is actually saving the snapshot taken at the checkpointing interval. Therefore, to schedule periodic checkpoints, you must enable periodic snapshots.

To schedule checkpoints:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Checkpoint Settings page (Figure 7-25).

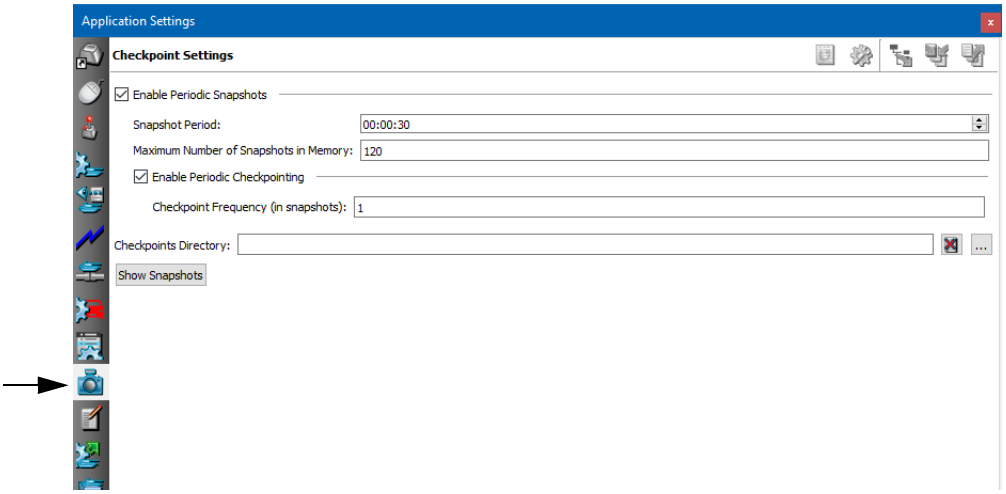


Figure 7-25. Checkpoint Settings page

3. Select the Enable Periodic Snapshots check box.
4. In the Snapshot Period box, specify how frequently VR-Forces should take snapshots.
5. Select the Enable Periodic Checkpointing check box.
6. In the Checkpoint Frequency box, specify how frequently to save checkpoints as a function of the number of snapshots taken. For example, 4 means save every fourth snapshot as a checkpoint. If you save snapshots every 30 seconds, then you would save a checkpoint every two minutes.
7. Optionally, specify the directory in which to save the checkpoints. If you leave this box blank, they are saved to the directory where the base scenario is located.

Disabling Periodic Checkpointing

To disable periodic checkpointing:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Checkpoint Settings page (Figure 7-25).
3. Clear the Enable Periodic Checkpointing check box.

7.8.2. Deleting Checkpoints

You can delete checkpoints manually in a file browser, like you can delete any other files on your computer. You can also easily delete them from the front-end.

To delete checkpoints:

1. Choose **File** → **Delete Scenario Checkpoints**. The Delete Checkpoints dialog box opens (Figure 7-26).

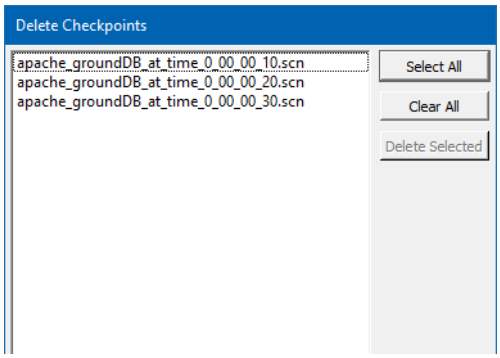


Figure 7-26. Delete Checkpoints dialog box

2. Select the checkpoints that you want to delete.
3. Click Delete Selected.

7.8.3. Creating Snapshots

You can create snapshots manually and you can specify that they be created automatically at a specified interval.

Creating Snapshots Manually

To create a snapshot:

1. Choose **File** → **Scenario Snapshots**, or on the Checkpoint Settings page, click Show Snapshots. The Scenario Snapshots dialog box opens (Figure 7-24).
2. Click Snapshot.

You can also create a snapshot by clicking the Snapshot button ( ) on the Scenario Toolbar.

Enabling Periodic Snapshots

When you enable periodic snapshots, you specify how frequently to take them, in simulation time, and the maximum number to store in memory at a time. If the number of snapshots reaches the maximum, VR-Forces drops the oldest snapshot and adds the newest.

To enable periodic snapshots:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Checkpoint Settings page (Figure 7-25).
3. Select the Enable Periodic Snapshots check box.
4. In the Snapshot Period box, specify how frequently VR-Forces should take snapshots.
5. In the Maximum Number of Snapshots in Memory box, specify how many snapshots to keep in memory.

7.8.4. Clearing the Snapshot List

Snapshots are stored in memory and listed in the Scenario Snapshots dialog box. They get cleared automatically when you load a scenario or rewind a scenario and run it again. If you roll back to a snapshot and run the scenario, all snapshots later than the rollback point are deleted. You can also clear snapshots manually.

When you clear the snapshot list, VR-Forces keeps the initial snapshot so that you can still rewind the scenario.

To clear the snapshot list:

1. Choose **File** → **Scenario Snapshots**, or on the Checkpoint Settings page, click Show Snapshots. The Scenario Snapshots dialog box opens (Figure 7-24).
2. Click Clear All.

7.9. Pausing a Scenario Automatically

You can specify a scenario end time (in simulation time). As the scenario progresses, the amount of time remaining is displayed in the Simulation Time Remaining Toolbar (Figure 7-27). When the scenario reaches this time it stops. You can resume the scenario by clicking the Play button.

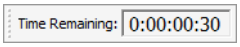


Figure 7-27. Simulation Time Remaining Toolbar

You can set the scenario end time when you create a scenario or after you have loaded it.

7.9.1. Setting the Scenario End Time for a New Scenario

The default scenario end time for new scenarios is 0 – do not pause the scenario.

To set the scenario end time for a new scenario:

1. Create a new scenario as described in "Creating a Scenario," on page 7-3.
2. In the New Scenario dialog box, select the Advanced tab.
3. In the Scenario End Time boxes, specify an end time in days, hours, minutes, and seconds.
4. Continue the scenario creation procedure.

7.9.2. Setting the Scenario End Time for an Open Scenario

You can set the scenario end time for a scenario after you open it. If you set a time that is sooner than the current elapsed time, the scenario stops immediately.

To set the scenario end time for an open scenario:

1. Choose **File** → **Scenario Properties**. The Scenario Properties dialog box opens.
2. Select the Advanced tab
3. In the Scenario End Time boxes, specify an end time in days, hours, minutes, and seconds.
4. Click OK.
5. Optionally, save the scenario. Remember that if the scenario is already running or has advanced and then paused, the end time is saved to a checkpoint, not the original scenario.

7.10. Running VR-Forces in Batch Mode

Batch mode allows you to run one or more scenarios multiple times, without direct action through the graphical user interface. A scenario run in batch mode runs with the parameters in the scenario file, except that you can override the random number seed.

Batch mode is "read only." You cannot save a scenario, create simulation objects or objects, pause the scenario, or otherwise change it, while a batch is running. However, you can view the Objects List Panel, plan windows, and open informational dialog boxes. You can close a scenario that is running in batch mode and resume control of VR-Forces.

7.10.1. The Batch File

A batch mode file contains the information that VR-Forces needs to run scenarios in batch mode. Batch mode offers the following options:

- Run a particular scenario multiple times.
- Run a series of different scenarios (batches), each of which can be run multiple times.
- Run from the command line using one back-end. If a scenario was created using multiple back-ends, you cannot run it in batch mode using this method.
- Run from the VR-Forces front-end using one or more back-ends.
- Generate Logger Control messages to have the MAK Logger record each simulation.

For each batch (one scenario run one or more times) within the batch file, you can:

- Specify the length of time the simulation is to run (in real time or simulation time).
- Specify the object parameter database to use.
- Specify that the random number seed used be the value specified in the scenario file, a value specified in the batch file, or be computer generated.

A batch file has the file extension *.bsn*.

7.10.2. Creating a Batch File

To run VR-Forces in batch mode, you create a batch file (*.bsn*). VR-Forces supplies a sample batch file. Use it as a model for creating your own batch files.

To create a batch file:

1. Create scenarios as you would any scenario. (For details, please see "Creating a Scenario," on page 7-3.)
2. In a text editor, open the sample batch file provided with VR-Forces.
3. Save the batch file under a new name, with the file extension *.bsn*.
4. Replace the sample values with your own, as described in "Editing a Batch File," on page 7-46.
5. Save the new batch file.

7.10.3. Editing a Batch File

A batch file is written in MAK Technologies Lisp format (MTL). (For information about the MTL format, please see "MAK Technologies Lisp (MTL) Files," on page A-3.) A batch file starts with parameters that affect the entire batch. Then it has a batch list, which contains one or more batches. Each batch specifies a scenario, the number of times it is to be run, and other parameters. Table 7-3 describes the parameters.

The following is an example of a batch file:

```
(scenario-batch
  (use-logger-control True)
  (logger-files-path "..\logger_tapes") ;; path is relative to Logger
  executable.
  (batch-list
    (batch
      (number-of-runs  2)
      (random-number-seed 4)
      (scenario-filename  "..\data\scenarios\Raid.scnx")
      (sms-filename
        "..\data\simulationModelSets\EntityLevel.sms")
      (simulation-run-duration 0 : 0 : 0 : 30.000000) ;; simulation time
    )
    (batch
      (number-of-runs  3)
      (scenario-filename
        "..\data\scenarios\Apache.scnx")
      (sms-filename
        "..\data\simulationModelSets\EntityLevel.sms")
      (run-duration 0 : 0 : 0 : 20.000000) ;; real time
    )
  )
)
```

Table 7-3: Batch file parameters

Parameter	Description
Scenario-batch parameters	
use-logger-control	Specifies whether or not to generate Logger Control messages. Options: ◆ False – Do not generate messages. ◆ True – Generate messages to create Logger files for each run of each batch.
logger-files-path	If use-logger-control is True, specifies the directory in which to store Logger files. The path can be absolute or relative to the Logger executable.
Batch-list parameters	
number-of-runs	Specifies how many times to run the scenario in this batch run entry.
random-number-seed	Optional. Overrides the random number seed in the scenario file. It specifies a random number seed as follows: ◆ -1 – use a value generated by the computer. Values are different for each run of this batch. ◆ Any non-negative integer – use the number specified for each run of the batch.
scenario-filename	Specifies the scenario to be run in this batch run entry. The scenario can be specified as an absolute path or a path relative to the directory in which the executable is located.
sms-filename	Optional. Overrides the simulation model set specified by the scenario and specifies the simulation model set file to use.
simulation-run-duration	Specifies the duration in simulation time that the simulation will run. It is mutually exclusive with run-duration. If both are specified, VR-Forces uses simulation-run-duration. ! Specify hours, minutes, and days as integers, for example 15. Specify seconds as real numbers, for example, 15.0.
run-duration	Specifies the duration in real time that the simulation will run. For example, if a scenario has a time scale set to 2 (twice real time), and a run-duration of 10 minutes, 20 minutes of simulation time will elapse before the batch stops running this scenario. It is mutually exclusive with simulation-run-duration. ! Specify hours, minutes, and days as integers, for example 15. Specify seconds as real numbers, for example, 15.0.

7.10.4. Running VR-Forces in Batch Mode

To run VR-Forces in batch mode:

1. Start VR-Forces. If any of the scenarios you plan to run in batch mode require multiple back-ends, start the back-ends with the required site and application IDs.
2. If you are going to generate Logger control messages, start the MAK Data Logger. (Make sure to use the same protocol as you are using for VR-Forces.)
3. Choose **File** → **Load Batch File**, or click the Load Batch File button () on the File Toolbar. The Load Batch File dialog box opens.
4. Select the batch file you want to run.
5. Click Open.

VR-Forces begins running the first scenario in the batch file. When VR-Forces finishes running the batch file, it closes the final scenario.



- When VR-Forces runs a batch file, if it cannot load a scenario specified in the file, it skips the scenario and goes to the next batch entry in the file.
- Automatic remapping is always enabled for batch mode.

Running in Batch Mode from the Command Line

If your scenarios only need one back-end, you can run a batch file from the command line.

➤ To run a batch from the command line, use the `-B` parameter, with the back-end in separate mode, for example:

```
vrfSimDIS -s 1 -a 3001
  -B "../userData/scenarios/batchfile.bsn"
```

The scenario begins to run automatically.

7.10.5. Recording Batch Scenarios

You can generate Logger Control messages and save scenario output to a Logger file. You must start the Logger before you run the batch scenario.

If you generate Logger Control messages, each simulation is saved individually by the Logger, with a filename in the format:

```
<batch_scenario_name>_<scenario_run_number>_<random_number_seed>.lgr
```

You can specify the directory in which you want the Logger file to be saved.

The MAK Data Logger does not record batch file runs if a previous recording exists. By default, the Logger does not allow you to overwrite existing files using the remote control API. So, once you record a batch run, you cannot record a subsequent run of the same batch. You can work around this problem by editing the *lgrConfig.xml* file to permit overwriting of Logger files. Make the following edits in the configuration file for each protocol for which you want to record batch runs:

Add the following lines to the <defaults> section:

```
<var name="processRemoteControl" type="bool" value="true"/>
<var name="recordRemoteControl" type="bool" value="true"/>
<var name="allowDeleteRemoteControl" type="bool" value="true"/>
```

Add the following lines to the <commands name="startup"> section:

```
<command domain="Simulation" protocol="DIS"
  command="SetRemoteControlSettings">
  <param name="process" var="processRemoteControl"></param>
  <param name="record" var="recordRemoteControl"></param>
  <param name="allowDelete" var="allowDeleteRemoteControl"></param>
</command>
```

Use the appropriate value for the protocol attribute. For DIS, set listenWhenIdle to false:

```
<var name="listenWhenIdle" type="bool" value="false"/>
```

7.11. Sending Run and Pause Messages to Simulation Participants

By default, when you run or pause a scenario, VR-Forces sends scenario control messages that are meaningful only to VR-Forces applications that are part of the current session. Sometimes VR-Forces users want all of the participants in a simulation, which may include non-VR-Forces applications, to know when VR-Forces runs or pauses a scenario. You can achieve this by telling VR-Forces to send standard Start/Resume and Stop/Freeze messages instead of the VR-Forces-specific messages. All of the applications that are listening for them will receive them and can respond as needed.



If you are running multiple VR-Forces sessions and you use Start/Freeze messages, all of the sessions receive them, which may defeat the purpose of having separate sessions.

To enable use of Start/Freeze messages:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 7-6).
3. Select the Send Standard Start/Resume and Stop/Freeze PDUs check box.

7.12. Recording VR-Forces Simulations with the MAK Data Logger

You can use the MAK Logger to record VR-Forces simulations and then play them back for review. If you want to view a Logger file in VR-Forces (as opposed to some other MAK viewer, such as VR-Vantage Stealth), note the following considerations:

- Do not load the original scenario in VR-Forces while you view the recording. Just load the terrain database that the simulation requires.
- If you are using HLA 1516, do not record a simulation and then immediately play it back. Exit the applications and start a new federation. Trying to play back the file in the federation from which it was recorded can result in name reservation conflicts and applications may crash.

If a scenario includes plans that have view control messages, the Logger records them. When you play the recording, VR-Forces and other VR-Vantage applications will change view modes in response to them. This feature lets you easily create demonstrations that change views.



8. Scenario Events

Scenario events let you provide non-simulated input to scenario participants.

Introduction to Scenario Events 8-2
Creating a Scenario Event 8-3
Adding Content to a Scenario Event 8-5
Adding Linked Events 8-6
Adding Intelligence Objects 8-8
Starting a Scenario Event 8-10
Controlling Scenario Events from a Plan 8-12
Ending a Scenario Event 8-13
Enabling and Disabling Scenario Events 8-14
Sorting the Event List 8-15
Deleting a Scenario Event 8-15
Showing and Hiding Scenario Events 8-16

8.1. Introduction to Scenario Events

A scenario event represents an incident or condition that exercise participants can respond to, for example a political event, a natural disaster, or some other novel event that might require a participant to respond. VR-Forces does not simulate scenario events. Objects in a simulation do not interact with scenario events and scenario events do not directly affect simulation objects (plans can include conditional statements that are contingent on scenario events). Scenario events are placeholders that allow you to provide a richer simulation experience.

Events can include rich text, audio files (WAV), video (AVI) and raster files (JPG, PNG, SVG, or BMP). Use of these media allow you to expand the scope and quality of a training experience without needing to build a complicated simulation to support it. The flexibility of scenario events allows you to use them in any way that adds value to an exercise, for example, mission briefings, and video teaching points.

Unless you disable it, when an event gets triggered, VR-Forces displays a dialog box that displays the event information. If a scenario has events that are triggered by simulation time or time of day, the Scenario Event Toolbar displays the amount of time until the next event (Figure 8-1).

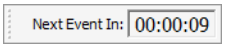


Figure 8-1. Scenario Event Toolbar



If you want to play MP4 video for scenario events, you must install an appropriate driver on your computer.

8.2. Creating a Scenario Event

To create a scenario event:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens (Figure 8-2).

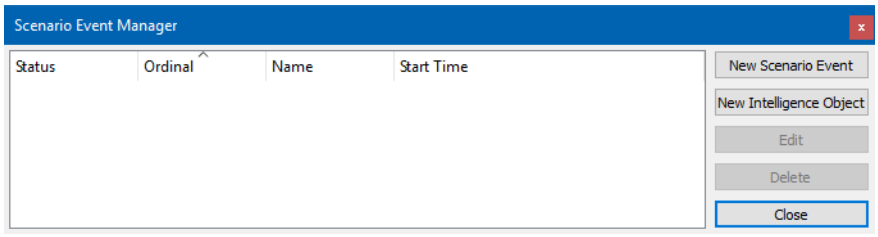


Figure 8-2. Scenario Event Manager

2. Click New Scenario Event. The Create Scenario Event dialog box opens (Figure 8-3).

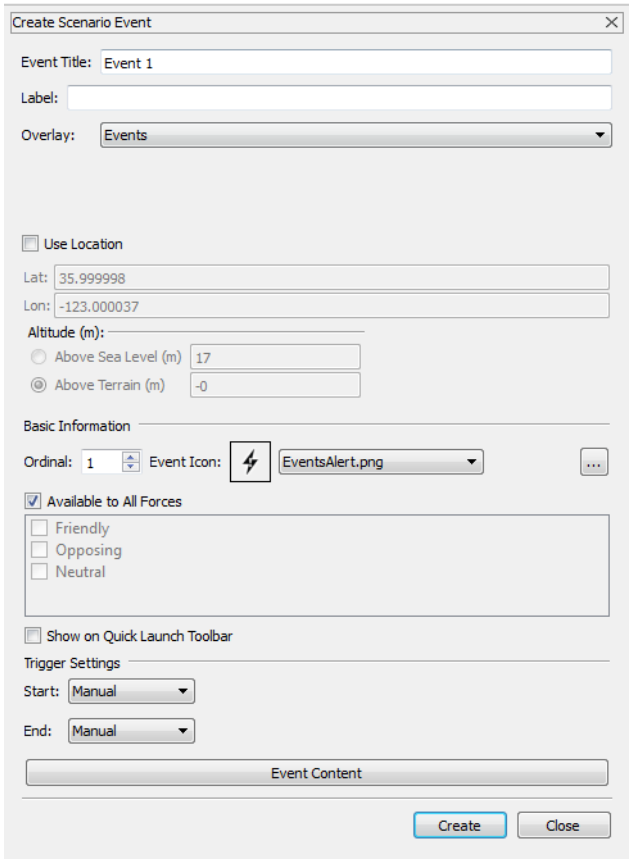


Figure 8-3. Create Scenario Event dialog box

3. Table 8-1 describes the parameters you can specify for a scenario event:

Table 8-1: Scenario event parameters

Parameter	Description
Event Title	A short descriptive name for the event.
Label	The label to be displayed. (optional)
Extended labels	If present, fill out extended labels. (optional)
Overlay	The overlay to which the event is assigned. Default: Events.
Use Location	You can specify a location for an event icon to appear when the event is triggered. You can click on the map where you want the event to occur or enter a location in any of the location formats. (optional)
Ordinal	Events are numbered in the sequence they are created. You can change the numbers to suit your purposes.
Event Icon	Select an appropriate icon to appear on the map at the location and time of the event. This icon is also used on the Quick Launch toolbar. (optional)
Forces	Select the forces you want to get the event when it occurs. You can select multiple forces.
Show on Quick Launch Toolbar	Adds the event to the Quick Launch toolbar for one-click activation.
Start	Specify one of the options for how the event gets triggered: ◆ Manual. Event will be triggered manually by the scenario administrator. ◆ Time of Day. Specify the time of day that should trigger the event. ◆ Linked. Specify the linking events as described in "Adding Linked Events," on page 8-6 that should trigger the event.. ◆ Simulation Time. Specify the simulation time that should trigger the event.. ◆ Immediate. The event starts as soon as it is created.
End	Specify how the end of the event is triggered, as follows: ◆ Manual. Ended by the scenario administrator. ◆ Time of Day. Specify the time of day at which it should end. ◆ Duration. Specify how long the event lasts. ◆ Permanent. The event does not end until you manually change the end type and end it. ! If you trigger an event by time and the specified time is prior to the start time of the scenario, the event is triggered when the scenario starts.
Event Content	Add content as described in "Adding Content to a Scenario Event," on page 8-5.

4. Click Create. Your new event appears in the event list. You can sort events by any of the column headings.

8.2.1. Adding Content to a Scenario Event

The content for a scenario event can be text, audio, video, graphics (BMP, JPG, SVG, or PNG), or some combination of them.

For sound and video files, you can specify files in any format that your computer supports.

To add content to a scenario event:

1. Create an event as described in "Creating a Scenario Event," on page 8-3.
2. In the Create Scenario Event dialog box, click Event Content. The Event Contents dialog box opens (Figure 8-4).

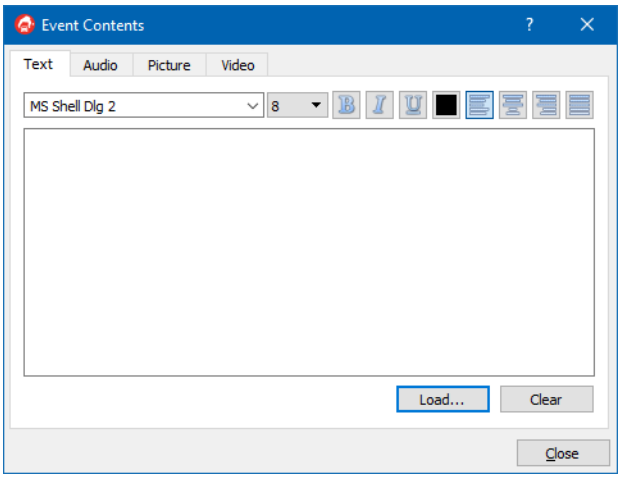


Figure 8-4. Event Contents dialog box

3. Select the tab for the content type that you want for this event. You can have more than one type of content.
4. For video, picture, or audio:
   a. Click Choose. A file selection dialog box opens.
   b. Select the file you want to add.
   c. Click Open.

   For text:

   a. Type the text you want in the dialog box or click Load and select a text file to load.
   b. Format the text as desired.
5. Optionally, test video and audio files by clicking the Play button.
6. Click Close.

8.2.2. Adding Linked Events

You can link multiple events to a single preceding event (many to one) or link one event to another linked event if your scenario requires it (one to one). You can create a linked event to trigger at any time after the preceding event it is linked to. For example, an event of a bridge being washed away could be linked to a preceding event such as reporting heavy rains, flooding, or rising river levels.

To add a link to a scenario event:

1. Create an event as described in "Creating a Scenario Event," on page 8-3.
2. For the Start trigger, select Linked. The Edit Links button is added to the dialog box.
3. Click Edit Links. The Edit Links dialog box opens (Figure 8-5).

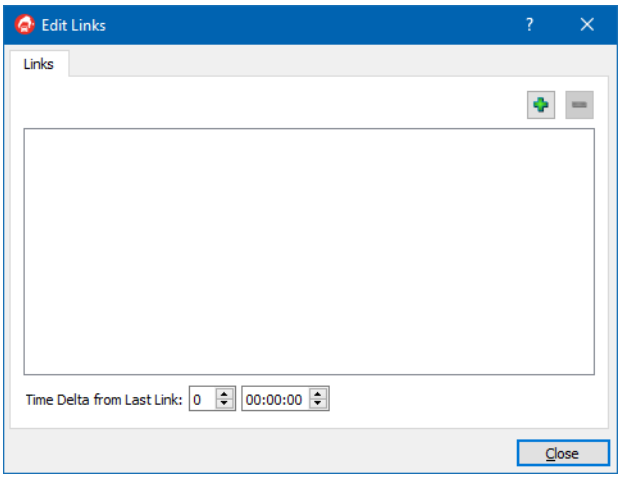


Figure 8-5. Edit Links dialog box

4. Click the Add button (+). The Link to Scenario Event dialog box opens (Figure 8-6).

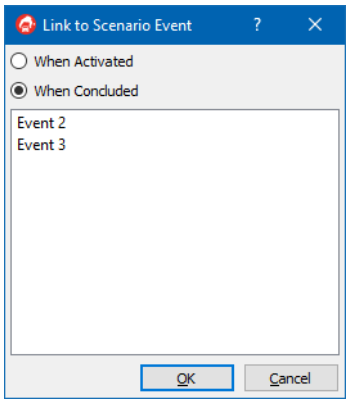


Figure 8-6. Link to Scenario Event dialog box

5. Select When Activated or When Concluded to indicate the relationship of the event to its linked event.
6. Select the event you want to link to.
7. Click OK.
8. Optionally, add additional linked events.
9. Specify the time delta. This is the amount of time to wait after the preceding event before triggering the linked event. If you specify multiple linked events, this is the time after all event conditions have been met.

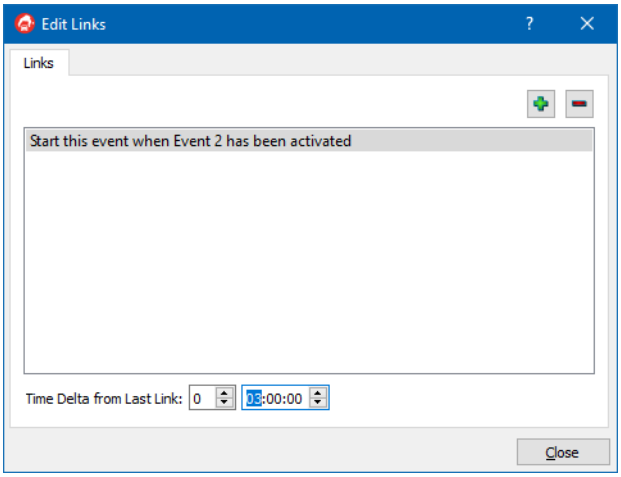


Figure 8-7. Edit Links dialog box with links

10. Click OK.

8.3. Adding Intelligence Objects

Intelligence objects are tactical graphics that trigger scenario events when they are detected by a simulation object. By default, intelligence objects are detected by visual sensors. However, you can configure them to be detectable by other sensors.

To add an intelligence object:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens (Figure 8-2).
2. Click New Intelligence Object. The cursor changes to draw mode and the Create Intelligence Object dialog box opens (Figure 8-8).

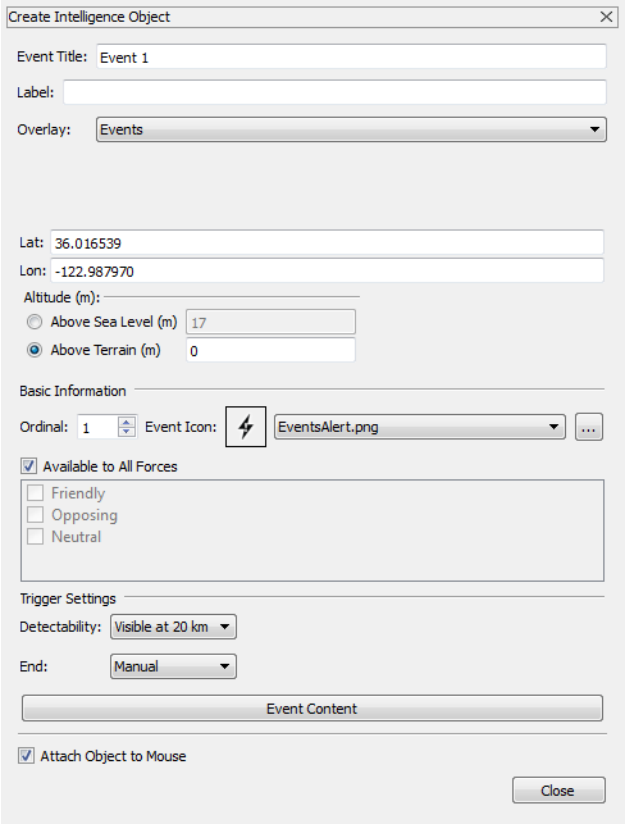


Figure 8-8. Create Intelligence Object dialog box

3. Click where you want the intelligence object to be located, or type its location in the dialog box.

4. Specify basic information and trigger settings as you would for a scenario event ("Creating a Scenario Event," on page 8-3), except that instead of configuring a start event, you specify detectability.

   To use only a visual sensor for detectability, select a distance in the Detectability list.

   To use other sensors, specify detectability as follows:

   a. In the Detectability list, select User Defined. The Edit Detectability button is added to the dialog box.
   b. Click Edit Detectability. The Object Visible to Sensor At dialog box opens (Figure 8-9).

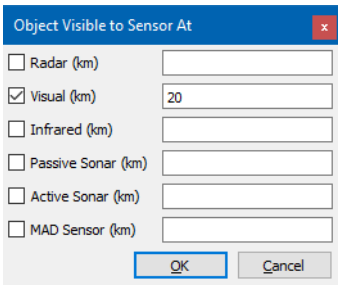


Figure 8-9. Set Sensor Signatures

   c. Select each sensor that you want to use to detect the intelligence object.
   d. For each sensor specify the sensor signature. (Although sensor signatures are technically unit-less, as a practical matter the sensor signature represents the distance in kilometers at which the object can be sensed.)
   e. Click OK.
   f. Click Create.

8.4. Starting a Scenario Event

A scenario event starts based on the Start Event trigger, by a command in a plan, or manually. Regardless of how a scenario event's trigger is set up, you can manually start an event at any time.



If an event has been added to the Quick Launch toolbar, you can start it by clicking its icon on the toolbar.

To start a scenario event manually:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens (Figure 8-10).

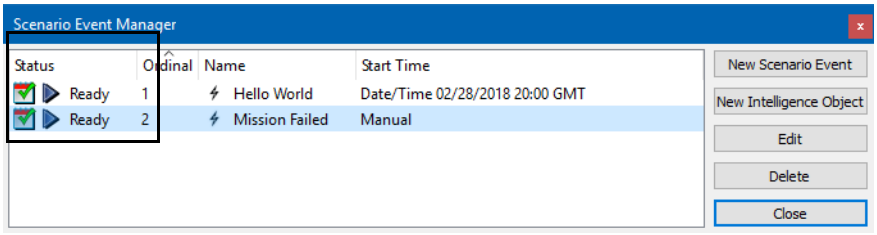


Figure 8-10. Scenario Event Manager, Play button

2. In the Status column, click the Play button (▶) for the event you want to start (Figure 8-10). The Event window opens (Figure 8-11).



Figure 8-11. Event window

8.4.1. Controlling Scenario Events from a Plan

You can start, stop, enable, and disable scenario events from plans. For example, a simulation object's plan could have an Entity In Area condition and when that condition is satisfied, you could trigger an event such as, "The enemy has seized your headquarters!" or "Congratulations, you have seized the enemy headquarters!".

Scenario events can also be used as the condition in a trigger. For details, please see Chapter 36, *Plan Concepts* and Chapter 37, *Writing Plans*.

To send a scenario event in a plan:

1. In a plan window, choose **Commands** → **Scenario Event**. The Scenario Event dialog box opens (Figure 8-12).

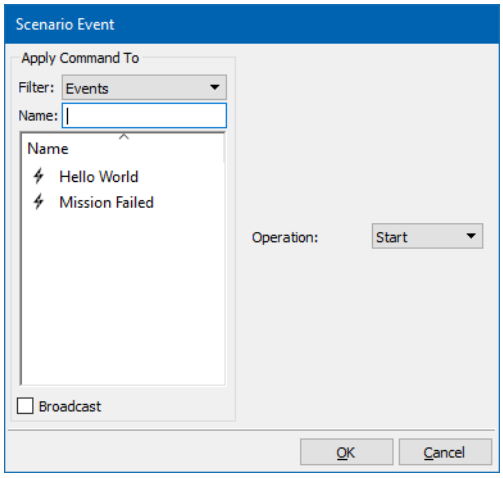


Figure 8-12. Scenario Event dialog box

2. Select the event you want to send.
3. Select an option in the Operation list.
4. Click OK.

8.5. Ending a Scenario Event

An event ends based on the End Event trigger setting, by a command in a plan, or manually.



When a scenario event ends, the Event window does not close automatically. The title bar indicates that the event is over by changing the text from Active to Inactive. You must close the window manually.

To end a scenario event manually:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens.
2. In the Status column, click the Stop (■) button for the event you want to stop (Figure 8-13).

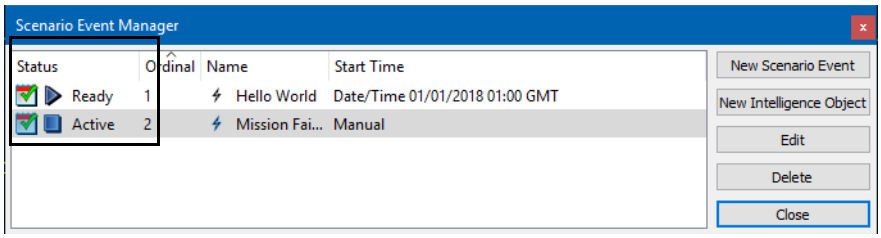


Figure 8-13. Scenario Event Manager, Stop button



If the Status column says Active, but does not have a Stop button, this is a permanent event. To end it, you must edit the event and change the End mode to Manual. A Stop button will be displayed.

3. Click Close.

8.6. Enabling and Disabling Scenario Events

You can enable and disable scenario events. You can change their status from the Scenario Event Manager, the Edit Scenario Event dialog box, and from plans. For details about enabling and disabling a scenario event from a plan, please see "Controlling Scenario Events from a Plan," on page 8-12.

To enable or disable a scenario event in the Scenario Event Manager:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens (Figure 8-13).
2. In the Status column, click the Enable/Disable ( ) button for the event you want to stop (Figure 8-13).

To enable or disable a scenario event in the Edit Scenario Event dialog box:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens (Figure 8-13).
2. Select the event you want to edit.
3. Click Edit. The Edit Scenario Event dialog box opens (similar to Figure 8-3).
4. In the Enabled State list, select the option you want.
5. Click Update.

8.7. Sorting the Event List

Scenario events have ID numbers. The numbers have no effect on when events get sent. They are simply a convenience for instructors and players. You can sort the list by number or by any other column listed.

➤ To change the sort order of scenario events, click the heading of the column that you want to sort on.

8.8. Deleting a Scenario Event

To delete a scenario event:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens (Figure 8-13).
2. Select the event you want to delete.
3. Click Delete.

8.9. Showing and Hiding Scenario Events

You can specify which events, if any, are displayed and whether or not the Event dialog box is displayed when an event gets triggered. The visible to force settings refer to visibility according to spot report settings and are based on the force availability specified when a scenario event is created or edited.

To show or hide scenario events:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Scenario Event Settings page (Figure 8-14).

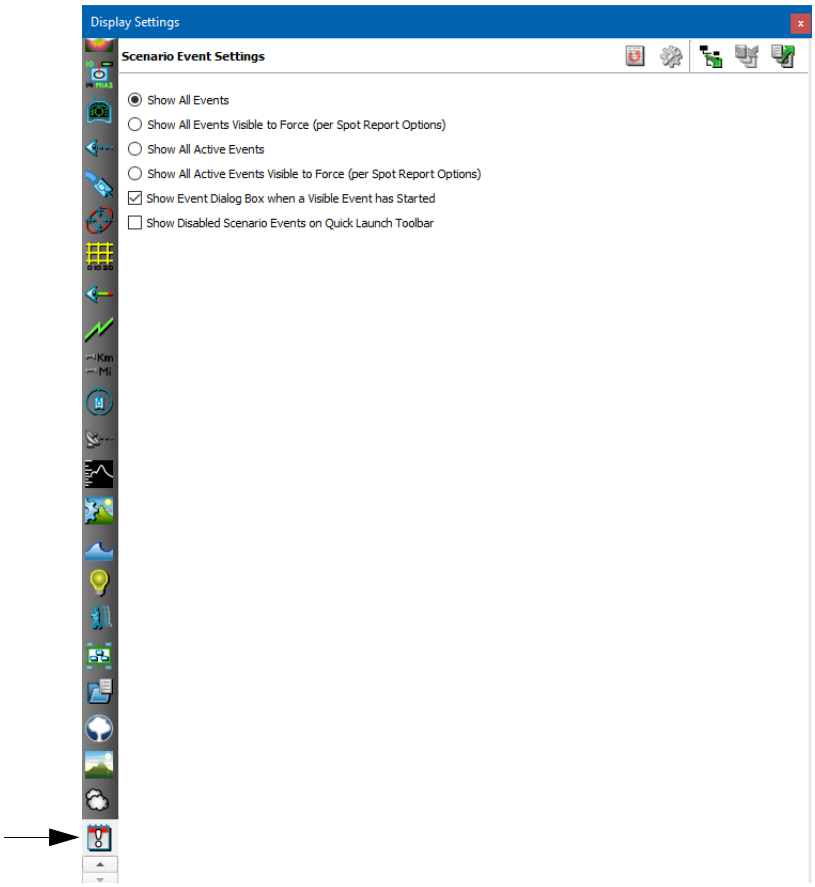


Figure 8-14. Scenario Event Settings page

3. Select an option for viewing events.



9. Target Detection and Combat Features

This chapter describes VR-Forces features related to target detection and acquisition, such as spot reports, hostility relationships, lasing, and munition damage.

Displaying Spot Reports 9-2
- Enabling or Disabling Spot Reports 9-3
- Configuring the Spot Reports Viewpoint 9-4
- Configuring the Spot Reports Certainty Level 9-7
- Applying Spot Reports to Tactical Graphics 9-8
- Displaying Labels for Spot Reports 9-8
- Using Spot Reports in Tasks 9-8
- Displaying Track Histories for Spot Reports 9-9

Displaying Tactical Data Link Messages 9-10
- Configuring the Display of TDL Messages 9-11
- Displaying Track Histories for TDL Messages 9-12

Managing Force Hostility Relationships 9-13
- Changing a Force's Hostility in a Plan 9-15

Detecting Targets 9-16
- Target Detection and Spot Reports 9-18
- Lasing Targets 9-18

Engagement Zones 9-19

Electronic Warfare 9-20

Radar Coverage 9-20
- Enabling the Display of Radar Coverage 9-22
- Displaying Radar Coverage 9-22
- Configuring Radar Coverage Displays 9-23

Using Sonar 9-24
- Propulsion Noise and Sonar 9-25

Launching Counter Measures (Chaff and Flare) 9-26

Modeling Artillery Munitions 9-28

9.1. Displaying Spot Reports

A spot report is a report by a simulation object to members of its force that it has detected a member of another force.

By default, VR-Forces displays simulation objects based on complete ground truth. That is, it shows all simulation objects in their exact location. However in real-world combat and training situations, participants rarely know where all other participants are, including those in their own force. This is called the fog-of-war. VR-Forces can simulate a semblance of the fog-of-war by displaying simulation objects based on spot reports, rather than ground truth.

When you display simulation objects based on spot reports, you do not see all members of forces other than your own. You only see spot reports for other-force simulation objects that have been detected by members of your force. Spot reports are represented by icons that are similar to simulation object icons, but they do not move as simulation object icons do. Furthermore, since spot reports lose value over time as the detected entities move to different locations, VR-Forces maintains a certainty level for the location of spot reports. If a detected entity is not detected again for a period of time, its certainty decreases and its icon becomes increasingly transparent to show that it is less certain that the entity is at the reported location.



- Generating spot reports can reduce the performance of the simulation engine.
- When spot reports are enabled, VR-Forces does not display icons for dead entities.
- When you rewind a scenario, the global spot report setting persists. Spot report settings for individual entities revert to the settings saved with the scenario.
- If you enable spot reports and you use Stealth observer mode, 2D icons are used for the entities that are identified by spot reports. You will not see any 3D models.

You can find links to videos that demonstrate simulation object classification and spot reports at https://www.mak.com/support/tutorials, or click the camera icon next to the procedure whose video you want to view.

9.1.1. Enabling or Disabling Spot Reports

You can enable or disable spot reports globally or per simulation object. To set spot reports for a simulation object, use the Spot Reports set data request, as described in "Spot Reports," on page 34-50.



By default, when you enable spot reports they are only sent to the front-end (for display purposes). They do not get sent to other simulation objects. This reduces the performance impact of spot reports on the back-end, but it means that entities will not receive that information for targeting purposes. Their spot report receivers will not show any spot reports in the Information dialog box. If you want spot reports to be sent to simulation objects, you must specify that using the Spot Report set data request. For more information, please see "Spot Reports," on page 34-50.

Enabling or Disabling Spot Reports Globally

When you enable or disable spot reports globally, the setting is applied to all simulation objects that have not explicitly set spot reports on or off with the Spot Reports set data request. The setting persists, which allows you to set a simulation object to use the global spot report setting.

To enable or disable spot reports for all simulation objects:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 9-1).

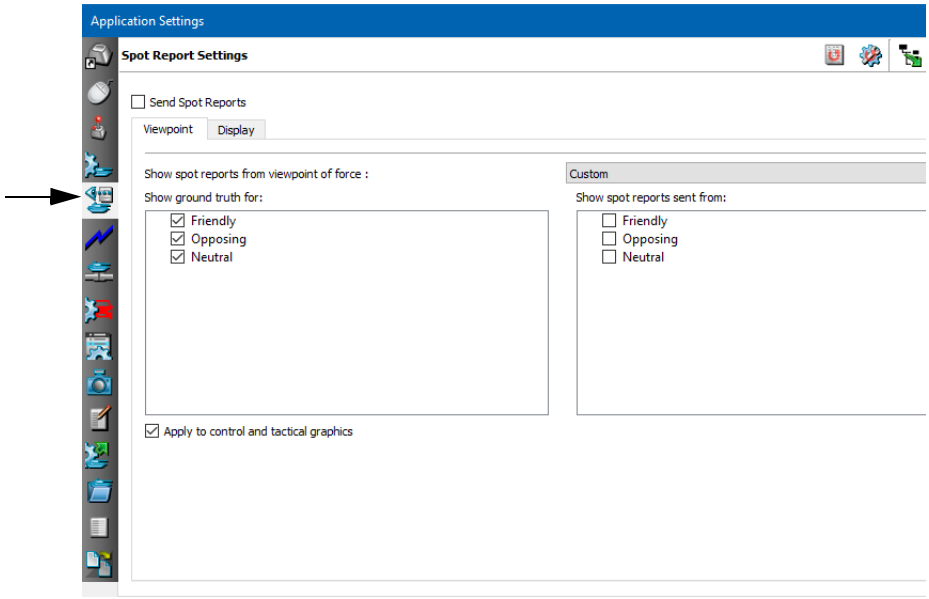


Figure 9-1. Spot Report Settings page, Viewpoint tab

4. Select or clear the Send Spot Reports check box. The change takes effect immediately.

9.1.2. Configuring the Spot Reports Viewpoint

The spot report viewpoint determines which simulation objects and tactical graphics are shown using ground truth and which are shown using only spot reports. You can select preconfigured viewpoints or create custom viewpoints.

- If you create a simulation object for a force that is not visible based on the viewpoint setting, after you create the simulation object, it will disappear. Therefore, it is recommended that you show ground truth for all forces when you are creating simulation objects and tactical graphics.
- When you rewind a scenario, the viewpoint configuration stays the same as in the previous run of the scenario. The global spot report setting stays the same. Entity-specific spot report settings revert to the settings saved with the scenario.

Selecting a Preconfigured Spot Report Viewpoint

VR-Forces provides viewpoints for the friendly, opposing, and neutral forces. In each of these viewpoints, VR-Forces displays ground truth for the viewpoint force and spot reports for the other forces. For example, if you select the Friendly force viewpoint, all simulation objects in Force 1 are displayed showing ground truth. All other simulation objects are displayed only if they have been detected by a friendly simulation object that is configured to send spot reports.

When you select a preconfigured viewpoint, the check boxes in the windows in the Map View group box are automatically set to match the selected viewpoint.

To configure spot reports:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 9-1).
4. Select an option in the Show Spot Reports from Viewpoint of Force list.

Configuring Custom Spot Report Viewpoints

If you do not want to use a preconfigured viewpoint, you can create a custom viewpoint. This option allows you to see ground truth and spot reports from a mix of forces. When you create a custom viewpoint, the Show Spot Reports from Viewpoint of Force list displays the viewpoint that you have created (usually Custom).

It is possible to create a custom configuration in which no objects are displayed, so think carefully about your choices.

To configure a custom viewpoint:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 9-1).
4. In the Show Ground Truth For window, select the forces for whom you want ground truth to be displayed. You would usually select the same forces as the ones selected in the Show Spot Reports Sent From window.
5. In the Show Spot Reports Sent From window, select the forces whose spot reports you want to display. In other words, if you are not showing ground truth for the opposing force and only want to see the opposing force simulation objects that are detected by friendly and neutral simulation objects, then select the Friendly and Neutral check boxes.

Configuring Same-Side Fog-of-War

There may be cases when you want to simulate same-side fog-of-war. That is, members of a force do not know where other members of their force are unless detected by a simulation object and reported using a spot report.

To simulate same-side fog-of-war, you must edit the *spot-report-generator.sysdef* file. You might want to create a customized file that you can assign only to the simulation objects that you want to simulate same-side fog-of-war. (You can use the Simulation Object Editor to change the spot report systems that simulation objects use.) In the system definition file change the **send-spot-reports-on-own-force** parameter to True, as follows:

```
(spot-report-generator-system
  (controllers
    (spot-report-generator
      (component-descriptor-type
        "spot-report-generator-controller-descriptor")
      ...
      (object-types-to-spot-report
        (object-type  1 (1 -1 -1 -1 -1 -1 -1))
        (object-type  1 (3 -1 -1 -1 -1 -1 -1))
        (object-type  1 (5 -1 -1 -1 -1 -1 -1))
      )
      (send-spot-reports True)
      (send-spot-reports-on-own-force True)
      (send-spot-reports-on-hostile-forces True)
      (send-spot-reports-on-neutral-forces True)
    )
  )
  ...
)
```

9.1.3. Configuring the Spot Reports Certainty Level

It is assumed that as time passes the validity, or certainty, of a spot report declines. To simulate this decline in certainty, VR-Forces can display the icons that represent spot reports at increasing levels of transparency. Once the initial validity period has expired, the spot report degrades over a configurable amount of time. The icon's transparency decreases in four steps, with each step being 25% of the total time allotted for degradation.



The validity period is specified by the **report-resend-period** parameter in the *spot-report-generator.sysdef* file in the SMS you are using. The default is 60 seconds.

To configure the amount of time, in seconds, over which a spot report degrades:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Display tab (Figure 9-2).
4. Specify a value, in seconds, in the Time After Spot Report Expiration Before Spot Report Is Removed box.

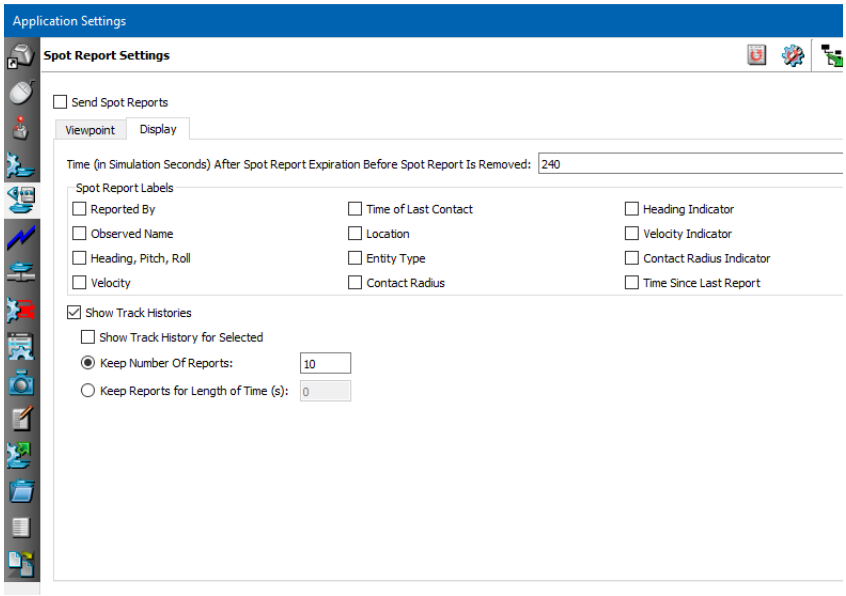


Figure 9-2. Spot Reports Option, Display tab

9.1.4. Applying Spot Reports to Tactical Graphics

If a tactical graphic is assigned to a force, you can display it based on spot reports rather than ground truth. The same criteria are applied to their display as are applied to the display of simulation objects. For information about how to assign objects to a force, please see "Creating Tactical Graphics," on page 40-2.

To apply spot reports to tactical graphics:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 9-1).
4. Select the Apply to Control and Tactical Graphics check box.

9.1.5. Displaying Labels for Spot Reports

You can display labels (symbol decorations) for spot reports with information similar to that for simulation object icons.

To display spot report labels:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Display tab (Figure 9-2).
4. In the Spot Report Labels group box, select the labels that you want to display.



You can also enable labels for spot reports on the Display Settings dialog box, Symbol Decoration Settings page.

9.1.6. Using Spot Reports in Tasks

Spot reports are listed in the Objects List Panel and other simulation object lists. Although the spot report icon in the display window may not show information about a spot report if it has not been identified, the entry in the list includes the object name. You can filter the list to only show spot reports.

You can select a spot report as a parameter in tasks that require you to select a simulation object. If you use a spot report in a task, VR-Forces carries out the task based on the ground truth of the reported simulation object, not the location of the spot report. For example, if you specify a spot report in the Follow Entity task, and the target simulation object moves after the spot report was received, the following simulation object will follow the actual location of the reported simulation object, not its location when the spot report was made.

9.1.7. Displaying Track Histories for Spot Reports

You can display track histories for spot reports (Figure 9-3). When you enable display of track histories for spot reports, it does not actually cause them to be displayed. You must also enable them for the selected spot report and select a spot report.

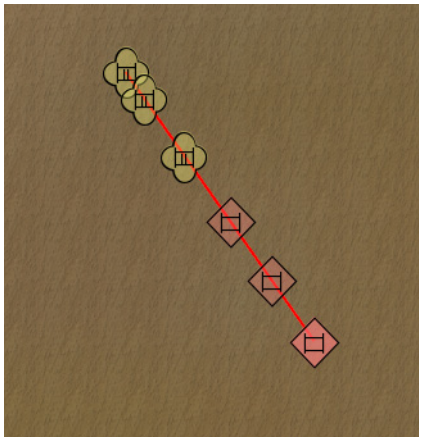


Figure 9-3. Spot report track history



The Track Histories setting on the Observer menu and the Observer toolbar do not affect the display of track histories for spot reports.

To display track histories for a spot report:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Display tab (Figure 9-2).
4. Select the Show Track Histories check box.
5. Select the Show Track History for Selected check box.
6. Select the spot report whose track history you want to show.

Pinning a Spot Report Track History

If you want to see more than one spot report track history at a time, you can pin track histories. If a track history is pinned, you do not have to select the spot report to see its track history.

To pin the track history for a spot report:

1. Select the spot report.
2. Choose **Objects** → **Decorations** → **Pin Track History**.

9.2. Displaying Tactical Data Link Messages

VR-Forces can display information about tactical data link (TDL) messages received over the network or sent by simulation objects that have a tactical data link message generator system. Systems are provided for each major platform type. TDL support is provided by an ITAR-controlled plug-in. However, if you do not have the plug-in, the base VR-Forces product simulates the effect of generating TDL messages.

When VR-Forces receives a TDL message, it displays:

- An icon similar to a spot report icon.
- Optionally, the time since the last report.
- Optionally, the data that accompanies the message, in up to 9 lines. The number of lines to display is configurable. There is no particular type of information or ordering information associated with the lines. It is simply based on the data in the TDL message.
- Optionally, track histories for the TDL reports.

For information about configuring the display of TDL messages, please see "Configuring the Display of TDL Messages," on page 9-11.

You can enable TDL messaging globally or for individual simulation objects. To enable TDL messages for a specific simulation object, use the TDL Messaging set data request. For details, please see "TDL Messaging," on page 34-54.

To enable TDL messaging globally:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Tactical Data Link Settings page (Figure 9-4).

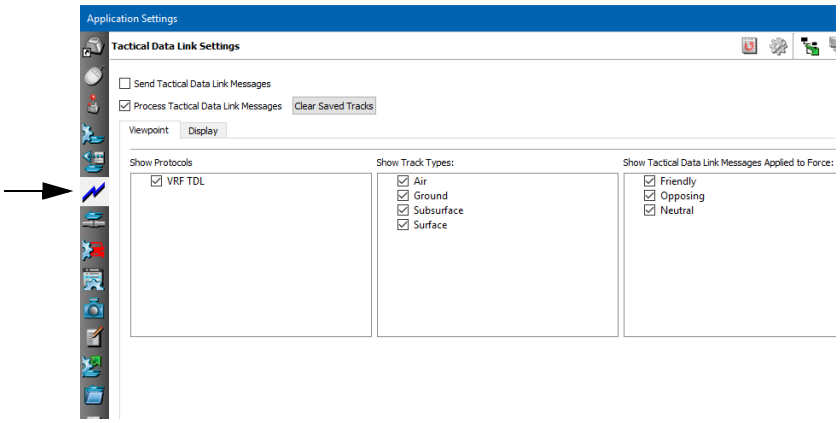


Figure 9-4. Tactical Data Link Settings page, Viewpoint tab

3. To send TDL messages, select the Send Tactical Data Link Messages check box.
4. To display TDL messages, select the Process Tactical Data Link Messages.

9.2.1. Configuring the Display of TDL Messages

You can configure how VR-Forces displays TDL messages.

To configure TDL messaging:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Tactical Data Link Settings page.
3. Select the Viewpoint tab (Figure 9-4).
4. In the Show Protocols list, select the type of protocol for which you want to show messages. (If the ITAR plug-in is installed, the VR-Forces TDL messages are not sent.)
5. In the Show Track Types list, select the platforms for which you want to show messages.
6. In the force list, select the forces from which you want to display messages.
7. Select the Display tab (Figure 9-5).

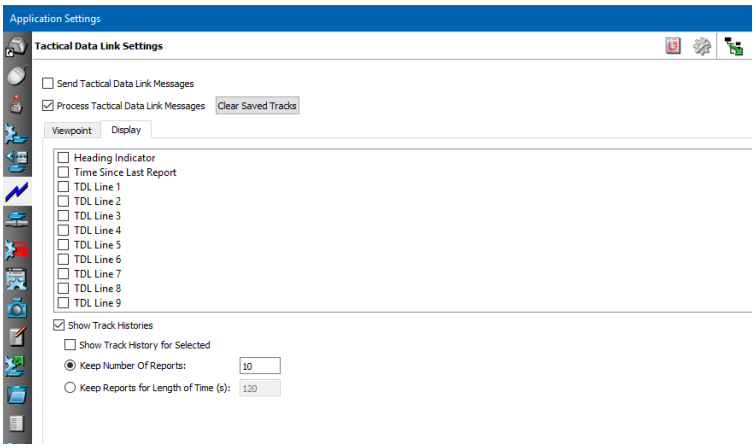


Figure 9-5. Tactical Data Link Settings, Display page

8. Select check boxes for the data you want displayed next to the TDL message icon.
9. Optionally, select the Show Track Histories check box.
10. If you enable track histories, specify whether or not to show track histories only for the selected entity, the number of reports to track, and how long to display them. For information about how to display track histories, please see "Displaying Track Histories for TDL Messages," on page 9-12.

9.2.2. Displaying Track Histories for TDL Messages

You can display track histories for tactical data link messages. When you enable display of track histories for TDL messages, it does not actually cause them to be displayed. You must also enable them for the selected message and select a message.

> The Track Histories setting on the Observer menu and the Observer toolbar do not affect the display of track histories for TDL messages.

To display track histories for a TDL message:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Tactical Data Link Settings page.
3. Select the Display tab (Figure 9-5).
4. Select the Show Track Histories check box.
5. Select the Show Track History for Selected check box.
6. Select the TDL message whose track history you want to show.

Pinning a TDL Message Track History

If you want to see more than one TDL message track history at a time, you can pin track histories. If a track history is pinned, you do not have to select the message to see its track history.

To pin the track history for a TDL message:

1. Select the message.
2. Choose **Objects** → **Decorations** → **Pin Track History**.

9.3. Managing Force Hostility Relationships

VR-Forces categorizes simulation objects as members of a force, such as friendly, opposing, or neutral. By default, friendly forces and opposing forces are hostile to each other. Neither is hostile towards neutral forces. You can change force hostility relationships dynamically. This may be particularly useful if you are simulating an environment in which civilian populations become hostile to a particular force, or in which armed militias, for example, switch allegiances to other forces.

You can add new forces in the Simulation Object Editor. For details, please see "Managing the Forces List," on page 67-34. You can change a force's hostility in a global or individual plan. For details, please see "Changing a Force's Hostility in a Plan," on page 9-15.

You can change the default hostility relationships by editing the *forceHostilty.mtl* file for the simulation model set you are using (for example, *./data/simulationModelSets/<model_set>/forceHostilty.mtl*).

When you change force hostility at runtime, the changes are saved as part of the scenario. The hostility relationships loaded with a scenario override the hostility relationships loaded by the simulation model set's hostility file.



Hostility relationships are not reciprocal. For example if you set neutral force simulation objects to be hostile to friendly force simulation objects, that does not mean that friendly force simulation objects are hostile to neutral force simulation objects.

To change the force hostility matrix:

1. Choose **Simulation** → **Hostility Matrix**. The Hostility Matrix dialog box opens (Figure 9-6). Each row in the matrix lists the hostility relationships between that row's force and all of the other forces (represented by color-coded columns).

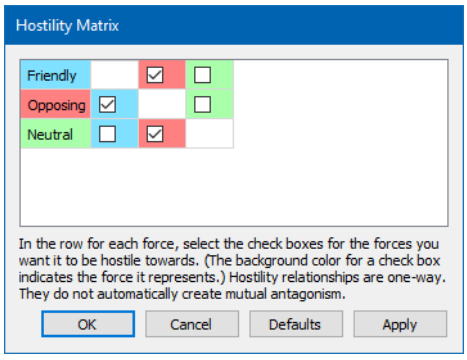


Figure 9-6. Hostility Matrix

2. For each row in the matrix, select the check boxes in the columns for the forces that you want this force to be hostile towards. Clear the check boxes for forces that you do not want it to be hostile towards.
3. Click OK.

9.3.1. Changing a Force's Hostility in a Plan

To change a force's hostility in a global or local plan:

1. Open a global plan or an individual plan.
2. Choose **Commands** → **Change Hostility**. The Change Hostility dialog box opens (Figure 9-7).

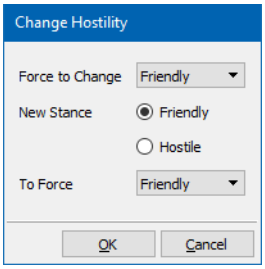


Figure 9-7. Change Hostility dialog box

3. In the Force to Change list, select the force whose hostility you want to change.
4. In the To Force list, select the force towards which you want to change hostility.
5. Select the new force relationship (New Stance), either Friendly or Hostile.
6. Click OK. The command is added to the plan.

9.4. Detecting Targets

Simulation objects can have one or more sensors, for example, visual, sonar, or radar, for detecting and identifying other simulation objects. If a simulation object has a Remote Sensor Receiver system, it can use another entity's sensors to discover sensor contacts. (For more information about remote sensors, please see "Remote Sensor," on page 34-40.)

When a simulation object detects another simulation object, it assigns it a combat identification (CID) level. If it continues to maintain contact with the simulation object, it upgrades the CID level as follows:



- CID Level 0 – not detected.
- CID Level 1 – Detected: the simulation object is detected, but only its platform is known, for example, ground.
- CID Level 2 – Classified: the simulation object's category is known, for example, tank.
- CID Level 3 – Identified: the simulation object's category and subcategory are known, for example M1A2.
- CID Level 4 – Full Knowledge: all visible information about the simulation object is known.

CID levels are assigned based on the distance to the detected simulation object and the length of time that the simulation object has been in contact. You can configure these values in the detection tables (*./data/simulationModelSets/<model_set>/vrfSim/detection/*.csv*). For details about these configuration files, please see "Detection Tables," on page 75-16.

By default, simulation objects do not fire at a detected hostile simulation object until it achieves the Identified level. You can configure this behavior by changing the detection-level-to-set-hostility parameter in the simulation object's sensor system.



Figure 9-8 shows screen captures of a simulation object as its CID level is advanced from Classified through Identified. In this scenario, spot reports are enabled. Therefore, when a simulation object is detected, it is displayed with a yellow icon, rather than the normal color for the force. It is not given the correct force color until the detected simulation object is CID level Identified.

You can find links to videos that demonstrate simulation object classification and spot reports at https://www.mak.com/support/tutorials, or click the camera icon next to the procedure whose video you want to view.

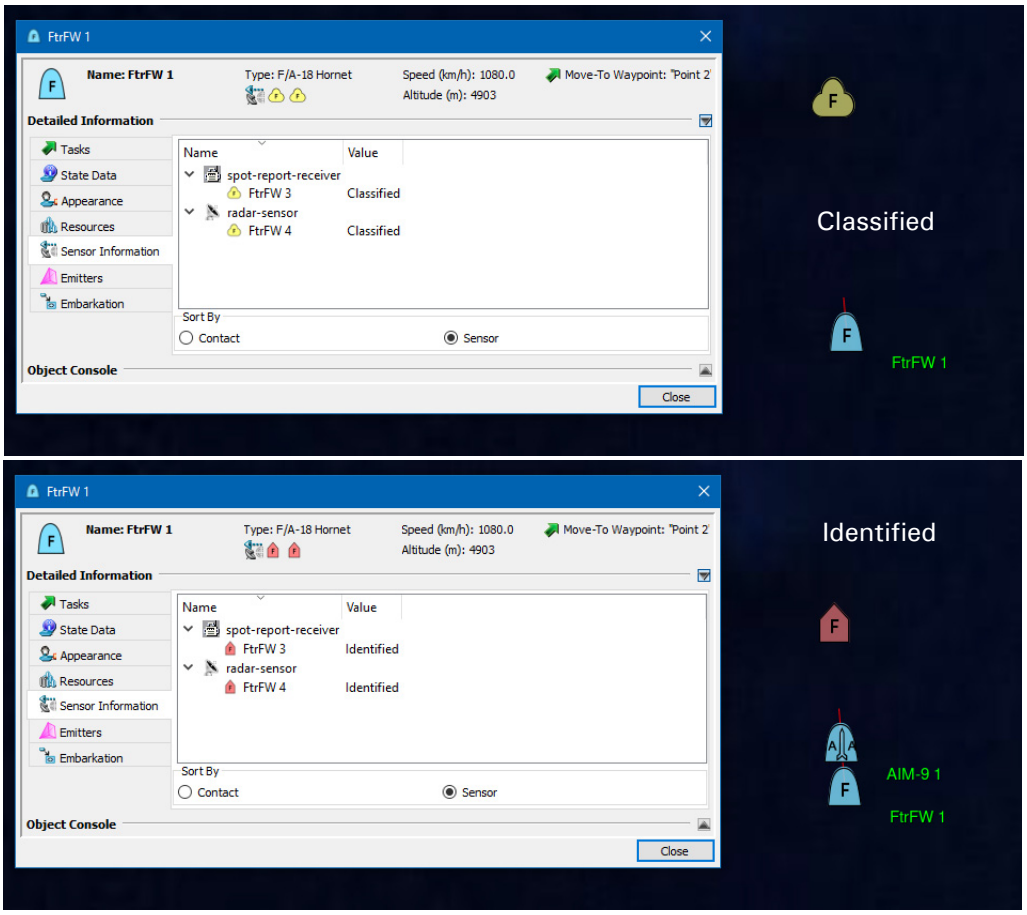


Figure 9-8. Simulation object classification and display when spot reports are enabled

9.4.1. Target Detection and Spot Reports

Simulation object detection and classification can be enhanced by enabling spot reports. Detected simulation objects are assigned CID levels based on length of contact and distance from the detecting simulation object. In some cases, such as a fixed-wing aircraft flying quickly over a battlefield, or a simulation object that is at the farthest point of its range from a target, contact might not be sufficient to advance to the Identified CID level. In such cases, even though the simulation object is technically capable of firing on the target, it might not do so because the CID level is not high enough.

If spot reports are enabled, the classification of simulation objects in the spot reports that a simulation object receives is combined with the directly sensed information. Therefore, if a simulation object classifies a target at a low CID level, but it receives a spot report that classifies the target at a high CID level, it can fire on the target based on the CID level in the spot report. For example, as noted in the previous paragraph, a fixed-wing aircraft is unlikely to maintain contact with a ground-based simulation object long enough to elevate its CID level to Identified. However, if a member of its force on the ground identifies the target and sends out a spot report indicating CID level 3 or 4, when the fixed-wing aircraft receives the spot report, it can add the target to its target list.

9.4.2. Lasing Targets

Some entities, such as DI Lasing, have laser-targeting capability. Others, such as attack helicopters, have laser targeting capability and carry laser-guided missiles. Laser-guided missiles do not use the standard VR-Forces targeting mechanisms. They only fire on targets that are identified by laser beams that have the same laser code as the laser-guided missile launcher. The laser beam could be generated by a laser designator on the same entity (autonomous lasing), or could be generated by another entity.

Laser codes are numbers in the range of 1111-1788 using the digits 1 through 8. Three-digit codes in the range 111-788 are also supported with an implied initial 1 in the code. By default, VR-Forces calculates a unique laser code for each entity when it is created. When you assign a laser code to an entity, any laser beams sent by that entity use the assigned laser code and the missiles carried by that entity (if applicable) use the same code. Therefore, for example, the default behavior for attack helicopters would be to fire their laser-targeted missiles only at targets that they have lased.



A laser-guided missile launcher must be able to see a laser spot and an appropriate target before it fires. If the missile loses the laser spot before it hits the target, it tracks to the last known laser spot.

Autonomous Lasing

By default, entities that have laser-guided missiles are configured for autonomous lasing. This means that they can lase targets and fire at them without outside intervention (for example, using the Lase Target task). Entities with autonomous lasing enabled identify and fire at targets independently, just like entities that use conventional munitions.

Synchronizing Laser Codes

There are cases in which you will want more than one entity to use the same laser code. For example, you might want one entity to lase targets (the designator), and another one to fire on them (the shooter). In this case you would assign both entities the same laser code. The designator would be given the hold-fire rule of engagement or not have any missile resources. The shooter would be configured to not lase targets (Lase Autonomous would be set to Off). It would just fire on the targets identified by the designator's laser.

You can synchronize the laser codes of the lasing entity and the firing entity by setting them for each entity with the Laser Code set data request, or you can let VR-Forces synchronize the codes with the Synchronize Laser Code set data request. For details, please see "Laser Code," on page 34-29 and "Synchronize Laser Code," on page 34-53.

9.5. Engagement Zones

Engagement zones are areas associated with ground-based simulation objects and air units that affect their rules of engagement. VR-Forces supports two types of engagement zones – missile engagement zones (MEZ) and joint engagement zones (JEZ). Engagement zones can have infinite height, like most VR-Forces areas, or can have a floor and ceiling, like extruded areas.

MEZs usually refer to zones associated with a ground-based air defense system. The system is permitted to engage targets within this defined area. Multiple engagement zones can be associated with a single air defense system and multiple systems can be associated with an engagement zone. To associate simulation objects with an engagement zone, you use the Engagement Zone set data request. For details, please see "Engagement Zones," on page 34-20.

JEZs are zones in which multiple systems can engage targets, usually both air units and ground based units. Successful management of fire in a JEZ may require a mechanism to manage the various systems and to help prevent friendly fire.

When a simulation object that is associated with an engagement zone detects a target, VR-Forces checks its rules of engagement. If the ROE is set to Fire When in Engagement Zone and if the target is in the engagement zone, it can fire on the target. If the target is not in the engagement zone, it cannot fire at it.

Ordinarily, simulation objects do not fire on other simulation objects unless they have a combat identification level of Identified or Full Knowledge. An engagement zone can be set to assume that unknown objects are hostile. In this case, if a simulation object has the Fire When in Engagement Zone rule of engagement, it can fire on unknown objects. For more information, please see "Assume Unknown is Hostile," on page 34-9.

Radars detect the position and type of potential targets, but not the force. Therefore, when a simulation object is set to fire on unknown objects, it will fire on objects in its own force if they do not identify themselves as being of the same force. The force of a target is known only if it has an active transponder (IFF is enabled) and it is of the same force as the radar.

Like other areas, engagement zones can be activated and deactivated. If an engagement zone is inactive, simulation objects whose ROE is set to Fire When in Engagement Zone cannot fire at targets when they are in the zone.

You can create engagement zones just like you would other areal objects. For details, please see Chapter 14, *Creating and Placing Objects* and "Creating Volumetric (Extruded) Tactical Graphics," on page 40-5.

9.6. Electronic Warfare

In entity-level scenarios, VR-Forces supports radar jamming by entities that have a tactical radar jammer system. The jamming entity must have line-of-sight to the jammed entity and the jammed entity must be within the cone of the electromagnetic emissions from the jamming entity. (For information about electromagnetic emissions, please see "Displaying Electromagnetic Emissions," on page 43-28.)

Radar jamming is enabled and set up using the following set data requests:

- "Add Jamming Target," on page 34-4.
- "Jammer State," on page 34-27.
- "Jamming Mode," on page 34-27.
- "Remove Jamming Target," on page 34-41.

9.7. Radar Coverage

VR-Forces can display the area in which a simulation object's radar can detect objects. The radar coverage area is color coded based on the altitude at which it is testing intersections. Each color shown indicates full visibility at that altitude and above. A sensor might be able to detect simulation objects at a lower altitude, but this is not certain. You can specify the altitudes at which you want to test intersections and can specify the colors used to display the radar coverage.

In 2D, the coverage area is depicted as bands of color (Figure 9-9). The color indicates the minimum altitude layer at which the radar can see in that direction.

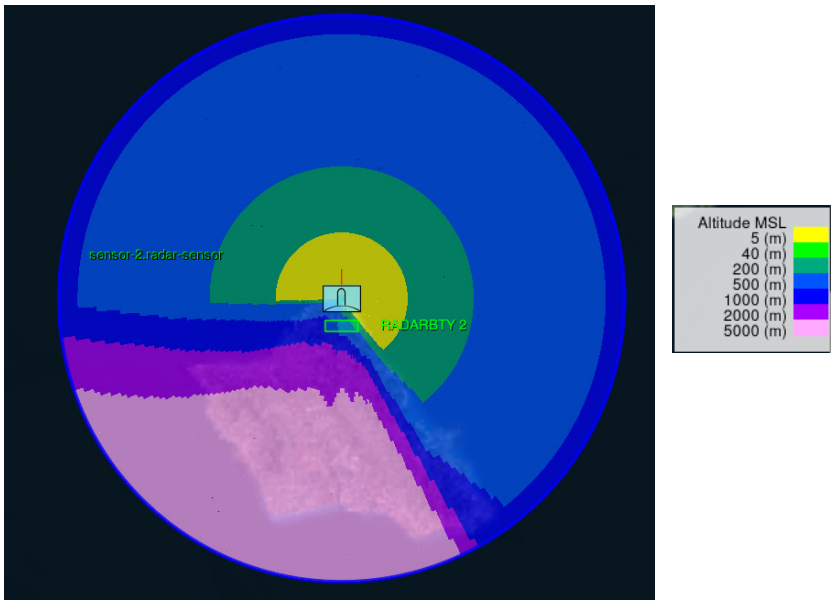


Figure 9-9. Radar coverage, 2D and color key

Figure 9-10 illustrates how the radar coverage display is calculated. Imagine a radar facility at sea level looking out over the ocean (similar to the illustration in (Figure 9-9)). The dotted lines represent the height above sea level of the radar beam as the earth curves (the illustration is not to scale). In the yellow band, the radar can detect all objects that are 5 meters above sea level or higher. In the light green band, the radar can detect all objects that are 40 m above sea level or higher. It might also detect objects between 5 m and 40 m. For each specified intersection height that you want to test, VR-Forces adds another band of color.

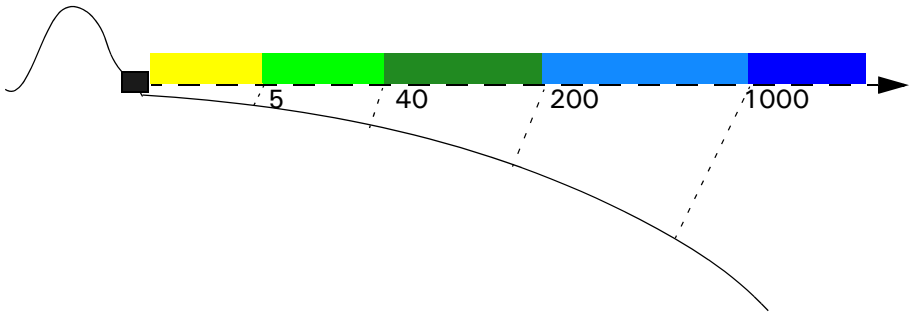


Figure 9-10. Radar coverage

In 3D, the coverage area is depicted as a shaded section of a sphere surrounding the radar. Areas covered by the radar are shaded. Areas that are obscured have no shading. A color gradient is used to show varying altitudes.

You can configure altitude layers and colors on the Display Settings dialog box, Radar Coverage Settings page.

- Radar coverage displays can take some time to calculate, and fill in gradually as the calculation proceeds.
- Because the coverage area is computationally expensive to display, this feature is only usable on units and entities that are stationary.

9.7.1. Enabling the Display of Radar Coverage

Radar coverage is enabled or disabled globally per observer. It is displayed per simulation object. For details about displaying radar coverage, please see "Displaying Radar Coverage," on page 9-22.

To enable the display of radar coverage, use one of the following methods:

- Choose **Observer** → **Radar Coverage**.
- On the Observer Toolbar, click the Radar Coverage button ().
- On the Display Settings dialog box, Observer Settings page, select the Radar Coverage check box.

9.7.2. Displaying Radar Coverage

To display radar coverage, you must enable it for the observer and then display it for a specific simulation object. For details about enabling radar coverage, please see "Enabling the Display of Radar Coverage," on page 9-22.

To display radar coverage:

1. Select a simulation object that has a ground-based radar, such as the Patriot Radar AN/MPQ-65 (entity-level) or Air Defense Radar (aggregate-level).
2. Choose **Objects** → **Decorations** → **Pin Radar Coverage**. A radar coverage display is added, centered on the selected simulation object.

9.7.3. Configuring Radar Coverage Displays

You can configure the colors used for radar coverage displays, the altitude at which colors change, and other aspects of the display.

To configure radar coverage:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Radar Coverage Settings page (Figure 9-11).

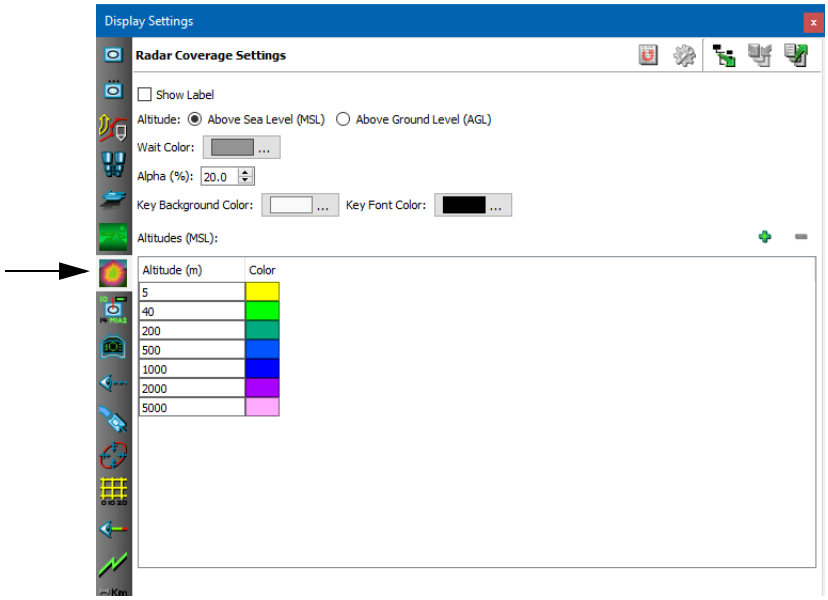


Figure 9-11. Radar Coverage Settings page

3. Change the settings as desired. Table 9-1 describes the settings.

Table 9-1: Radar coverage settings

Option	Description
Show Label	Display the name of the radar coverage area.
Altitude	Calculate altitude above sea level or above ground level.
Wait Color	The color to use for the radar coverage fan while the front-end waits for results from the back-end.
Alpha	The amount of transparency for the coverage areas.
Key Background Color	The background color to use for the altitude key.
Key Font Color	The color to use for the text on the altitude key.
Altitudes	The altitudes to test for intersections and their associated colors.

9.8. Using Sonar

VR-Forces simulates active and passive sonar. Entities can be configured with three types of sonar sensor systems: Active Sonar, Passive Sonar, and Sonar. The Sonar system includes both active and passive sonar. Both types of sonar use the VR-Forces sensor signature mechanism. The signatures are different, reflecting the different chances of detecting entities using the two types of sonar.

Passive sonar has the following characteristics:

- Sonar can be dipped at a depth other than that of the entity.
- If an entity is moving faster than a specified speed, passive sonar does not work.
- Sonar takes into account thermocline data. Thermoclines are layers of water that have different temperatures and, therefore, different density. Sound transmission varies as it passes through the thermocline. In VR-Forces sonar effectiveness is reduced as it passes through thermocline boundaries. For more information, please see "Setting the Thermocline," on page 11-37.
- Passive sonar ignores line-of-sight. It does not take into account underwater terrain or the presence of islands.
- If an entity is publishing propulsion noise, it can affect its detectability by passive sonar.

Active sonar has the following characteristics:

- Active sonar should acquire targets more quickly than passive sonar.
- If an entity has an active sonar system enabled, it is more easily detected by passive sonar systems.
- Entities with an active sonar that is enabled publish Underwater Acoustics PDUs.
- Entities can have multiple active sonar modes, similar to radar modes, using emitter beams. For details about configuring emitters, please see "Configuring Emitters," on page 79-2.
- Active sonar detects targets within the range of its emitter beams, but without regard to the direction of the beams.
- Active sonar ignores propulsion noise and the presence of active sonar on the target.

RPR FOM 1.0 does not support active sonar. If you try to create an entity that has an active sonar system, the entity will not be created and a warning will be sent to the console. If you try to load a scenario that has such an entity, it will not be created.

Sonar only works in water. Therefore rotary-wing entities with sonar systems must set their depth in order to sense anything.

9.8.1. Propulsion Noise and Sonar

If an entity is generating propulsion noise, it is easier to detect using passive sonar. VR-Forces publishes propulsion noise for surface and subsurface entities. If the power plant for an entity is turned off, it does not publish propulsion noise. To turn propulsion noise on or off, change the power plant appearance (for details, please see "Appearance," on page 34-8) or set the **power-plant-active** parameter in the propulsion-noise-generator component of the entity's movement system.

VR-Forces publishes three propulsion noise values:

- Current Shaft RPM, calculated as `current speed * speed-to-rpm-factor`.
- Ordered Shaft RPM, calculated as `ordered speed * speed-to-rpm-factor`.
- Shaft RPM Rate of Change, calculated as `current acceleration * acceleration-to-rpm-factor`.

The values for current speed, ordered speed, and current acceleration are the current values for the entity. The values for speed-to-rpm-factor and acceleration-to-rpm-factor are taken from the **speed-to-rpm-factor** and **acceleration-to-rpm-factor** parameters in the propulsion-noise-generator component of the entity's movement system.

When a target is being detected by passive sonar, if it is publishing propulsion noise, its calculated RPM is compared to a nominal RPM (default 100) to generate a sensor signature modifier. For example, if an entity is publishing an RPM of 80, its signature is scaled by 0.8. If its RPM is 150, the signature is scaled by 1.5.

RPR FOM 1.0 does not support propulsion noise. If you try to create an entity that has propulsion noise enabled, the entity will not be created and a warning will be sent to the console. If you try to load a scenario that has such an entity, it will not be created.

9.9. Launching Counter Measures (Chaff and Flare)

Fixed-wing and rotary-wing entities can launch counter measures (chaff and flare) when they are targeted by air-to-air and ground-to-air missiles. By default, entities launch counter measures automatically based on settings in the *countermeasures.sysdef* file. When an aircraft detects a hostile missile within a specified range, it determines the type of missile and looks up the type of counter measures to use in the *USCounterMeasuresTable.asl* or *CISCounter-MeasuresTable.asl* table.

Figure 9-12 shows a friendly force fighter launching counter measures against an aphid missile.



Figure 9-12. Counter measures (flare)

You can disable automatic counter-measures with the Counter Measures Auto Launch set data request. You can also launch counter-measures with the Launch Counter Measures task. The Launch Counter Measures task is not an exclusive task, so it does not interfere with the entity's current task. For details about the task and set data request, please see "Launch Counter Measures," on page 27-17 and "Counter Measures Auto Launch," on page 34-13.

You can configure the behavior of automatic counter measures in the *counter-measures-launcher.sysdef* and *cis-counter-measures-launcher.sysdef* files (in *./data/simulationModelSets/EntityLevel/vrfSim/systems/other*). Table 9-2 describes the parameters that you might want to change.

Table 9-2: Counter measure launcher parameters

Parameter	Description
auto-launch-enabled True	Enables or disables automatic launching of counter measures.
auto-launch-range	The range within which an entity will detect missiles against which to launch counter measures.
auto-launch-reset-time	The amount of time, in seconds, an entity will wait before launching additional counter measures.
auto-launch-default-resource	The default counter measure to use.
auto-launch-number-of-resources	The number of counter measures to launch for each launch sequence.
counter-measures-select-table-file	The counter measures selection table to use to determine which counter measures to fire for a particular missile.

9.10. Modeling Artillery Munitions

VR-Forces models artillery in two ways – direct fire and indirect fire. Artillery entities, such as howitzers use a direct fire model. The artillery entity is modeled and the munitions it fires are modeled. Indirect fire simply models artillery barrages within an area. For details about indirect fire, please see Chapter 10, *Indirect Fire, Ballistic Missiles, and Munition Detonations*.

By default, when an artillery entity fires, VR-Forces models the munition. Munitions are not affected by wind speed or friction. VR-Forces can also model artillery without modeling the munition. In this case, it just publishes a fire event and detonation event. Modeling munitions requires more computer resources than not doing so. Therefore, in large simulations you may want to disable modeling of munitions.

To disable modeling munitions, in the indirect-fire-actuator, set the simulate-munition parameter to false.



10. Indirect Fire, Ballistic Missiles, and Munition Detonations

This chapter explains how to create and edit indirect fire events and ballistic missiles. These features are only supported in entity-level scenarios.

Introduction to Indirect Fire 10-2
- Creating an Indirect Fire Event 10-3
- Editing Indirect Fire Events 10-6
- Deleting Indirect Fire Events 10-6
- Configuring Indirect Fire Event Default Values 10-7

Ballistic Missiles 10-8
- Firing Ballistic Missiles 10-9
- Editing Missile Target Events 10-11
- Deleting Missile Target Events 10-11

Creating Munition Detonations 10-12

10.1. Introduction to Indirect Fire

The indirect fire feature allows you to create, configure, and schedule indirect fire events. Indirect fire events are not associated with a particular simulation object. VR-Forces does not model the launcher or trajectory of the munitions. Indirect fire results in one or more salvos of one or more rounds in or near a designated area of the terrain. Figure 10-1 illustrates indirect fire within an indirect fire area.

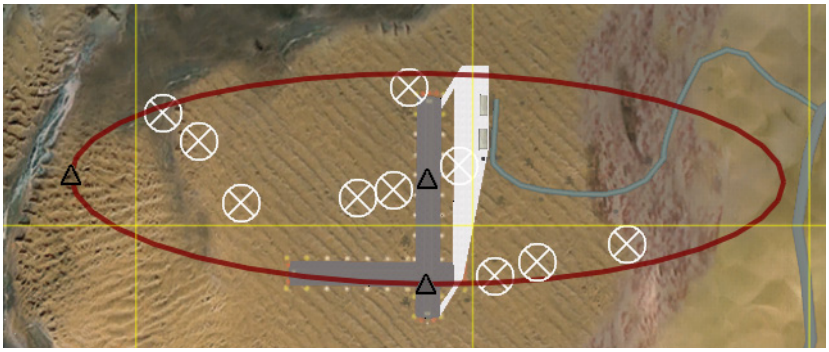


Figure 10-1. Indirect fire event

Indirect fire types are configured in *./data/simulationModelSets/EntityLevel/indirectArtilleryTypes.mtl*.

Indirect fire is not supported in aggregate-level scenarios.

10.1.1. Creating an Indirect Fire Event

When you create an indirect fire event, you specify an area in which the event is to take place, the parameters of the event, and a starting time. The area in which an indirect fire event takes place is a tactical graphic and is listed on the Objects List View of the Objects List Panel.

By default, indirect fire areas are placed on the Indirect Fire layer (which is created for the first indirect fire object). (For information about overlays, please see Chapter 39, *Overlays*.)

To create an indirect fire event:

1. Choose **Simulation** → **Munition Targets**. The Munition Target Settings dialog box opens (Figure 10-2).

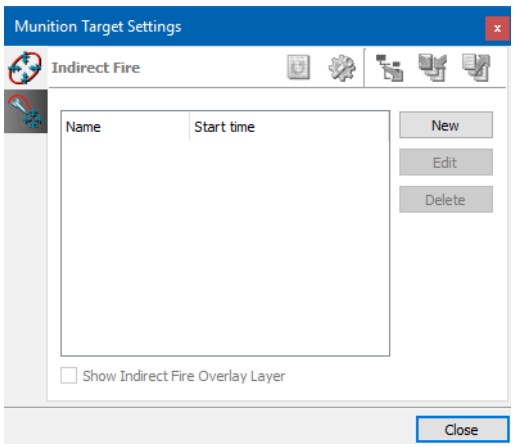


Figure 10-2. Munition Target Settings dialog box, Indirect Fire page

2. Select the Indirect Fire page.
3. Click New. The cursor changes to create mode and a tab for the Create Indirect Fire dialog box is added to the window.
4. Draw an ellipse on the map to specify the area of the fire event:
   a. Click to set the center of the ellipse. A center point is set and a small circle is displayed with a vertex indicator at the bottom of the circle.
   b. Drag the mouse up or down and click to set the height of the ellipse. A vertex indicator is added to the left side of the circle.
   c. Drag the mouse left and right and click to set the width of the ellipse (Figure 10-3).

   Optionally, you can specify the properties of the ellipse in the Create Indirect Fire dialog box.

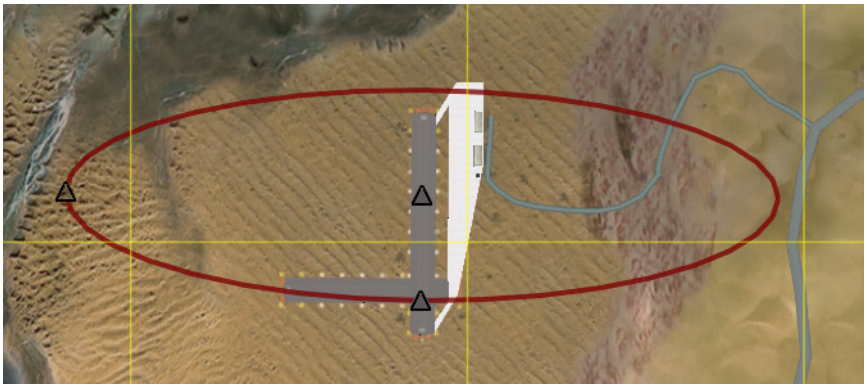


Figure 10-3. Indirect fire event area

5. Click the tab to display the Create Indirect Fire dialog box (Figure 10-4).

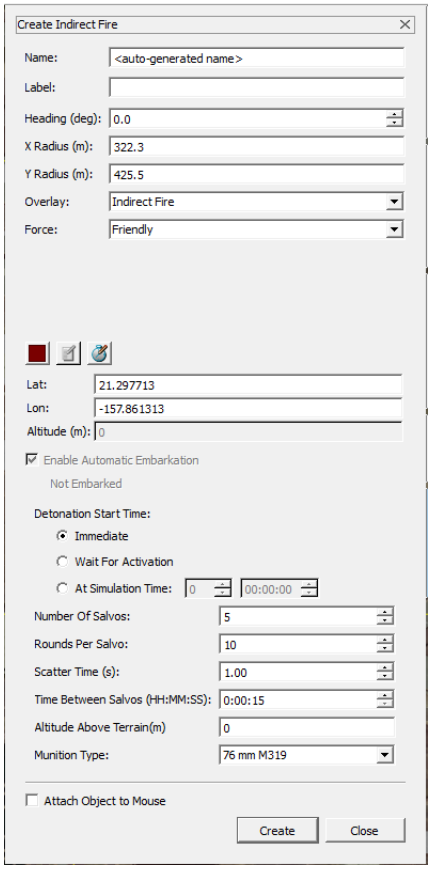


Figure 10-4. Create Indirect Fire dialog box

6. Type a name for the event in the Name text box, or accept the default name (Indirect Fire *n*).

7. Optionally, specify a layer for the event. (If there are no layers in the scenario, one iscreated when you click Create to create this indirect fire event.)
8. Configure the parameters of the fire event, as follows:
   - Select the detonation start time. The options are:
     - Immediate. Indirect fire begins immediately.
     - Wait for Activation. The indirect fire area is created in the inactive state. Indirect fire begins when an Activation set data request activates the area. This lets you activate indirect fire manually or from a plan.
     - At Simulation Time. The simulation time at which you want the munition to start firing.
   - In the Number of Salvos box, enter the number of salvos you want to fire.
   - In the Rounds Per Salvo box, enter the number of rounds you want to fire for each salvo.
   - In the Scatter Time box, enter the length of time for each salvo. The rounds in each salvo will be detonated at random times within this time period.
   - If you specified more than one salvo, in the Time Between Salvos box enter the amount of time you want to pass between each salvo.
   - Optionally, specify an altitude above the terrain that the rounds should detonate at.
   - Select the munition type in the Munition Type list.
9. Click Create. The Indirect Fire event is added to the list (Figure 10-5). The area is added as an object on the Indirect Fire layer on the Overlays View of the Objects List Panel (unless you chose a different layer).

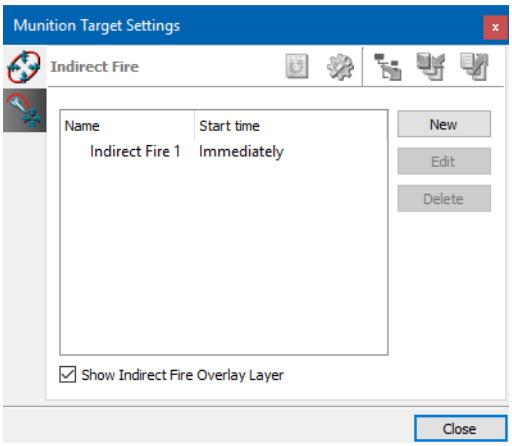


Figure 10-5. Indirect Fire page with indirect fire event

10.1.2. Editing Indirect Fire Events

You can edit the properties of an indirect fire event. You can also directly manipulate the area that specifies the event.

To edit an indirect fire event:

1. Choose **Simulation** → **Munition Targets**. The Munition Target Settings dialog box opens.
2. Select the Indirect Fire page (Figure 10-5).
3. Select the indirect fire event that you want to edit.
4. Click Edit. The Edit Indirect Fire dialog box opens.
5. Change the properties that you want to change.
6. Click OK.

10.1.3. Deleting Indirect Fire Events

To delete an indirect fire event:

1. Choose **Simulation** → **Munition Targets**. The Munition Target Settings dialog box opens (Figure 10-2).
2. Select the Indirect Fire page (Figure 10-5).
3. Select the indirect fire event that you want to delete.
4. Click Delete.
5. Click Close.

10.1.4. Configuring Indirect Fire Event Default Values

You can configure a set of default values that get applied to the Create Indirect Fire dialog box.

To configure default values for indirect fire events:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Indirect Fire Settings page (Figure 10-6).

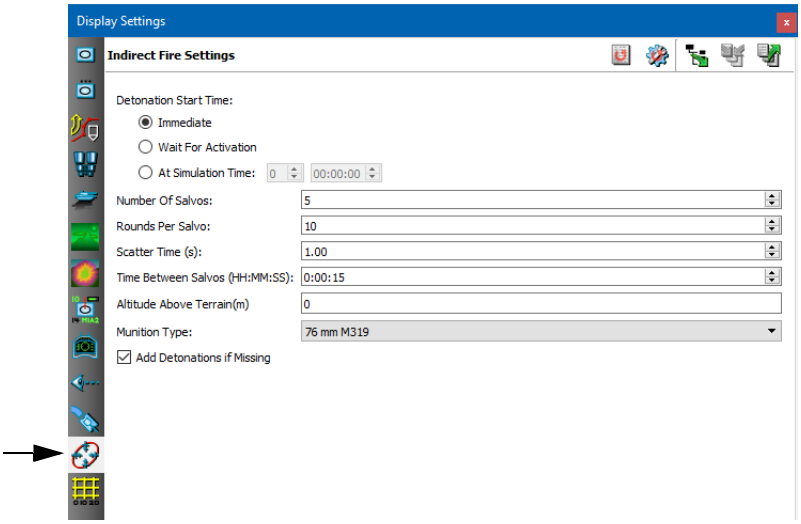


Figure 10-6. Indirect Fire Settings page

3. Set the parameters to the desired default values. Most are self explanatory. The Add Detonations if Missing check box causes VR-Forces to map a detonation to a detonation graphic if there are no mappings for that detonation event in the Detonation Mapping Settings page of the Simulation Object Type Mappings dialog box. If you do not want a detonation displayed for detonation events that are not mapped, clear this check box.
4. Click Close.

10.2. Ballistic Missiles

VR-Forces supports two types of ballistic missiles, as follows:

- Strategic missile weapon systems. Entities with strategic missile weapon systems can fire at targets using Launch Ballistic Missile tasks.
- Fixed trajectory ballistic missiles. The trajectory of a missile is determined by a series of coordinates in a comma-separated values (CSV) file. VR-Forces includes some sample CSV files. VR-Forces users are responsible for creating their own files if they want missiles to fly with different characteristics.

Strategic missile systems calculate a trajectory based on a maximum altitude and the distance to the target. They attempt to accurately represent the speed of the missile and they can take into account the curvature of the earth. (For information about strategic missile launch tasks, please see "Launch Ballistic Missile at Location," on page 27-14 and "Launch Ballistic Missile at Target Object," on page 27-14.)



The Ballistic Missile tasks do not use physics-based modeling. They calculate a simple parabola for the trajectory and move the missile along the trajectory ignoring atmospherics, thrust stages, cargo weight, gravity, and the rotation of the earth.

You fire a fixed trajectory ballistic missile by specifying a target and choosing a missile type. The characteristics of the missile type are specified in the Simulation Object Editor. A missile launches at the same altitude as the target point at a distance determined by the CSV file. Therefore, if a target point is placed at or near the ground, it is possible that due to variations in the terrain, the launch location could be underground. Once the missile launches, it follows the trajectory contained in the CSV file that it is configured with. It detonates when it reaches the target point. For details about configuring missiles, please see "Configuring Ballistic Missile Movement," on page 69-25.

This remainder of this section describes how to fire fixed trajectory ballistic missiles.



Ballistic missles are not supported in aggregate-level scenarios.

10.2.1. Firing Ballistic Missiles

To fire a ballistic missile:

1. Choose **Simulation** → **Munition Targets**. The Munition Target Settings dialog box opens.
2. Select the Missile Target page (Figure 10-7).

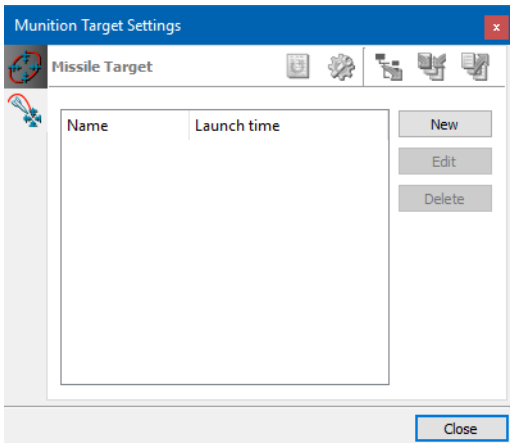


Figure 10-7. Munition Target Settings dialog box, Missile Target page

3. Click New. The cursor changes to a cross-hair.
4. Click the location where you want the missile to hit. The Create Missile Target dialog box opens (Figure 10-8).

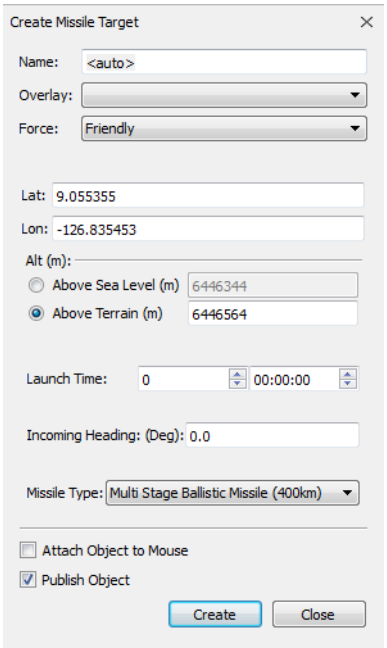


Figure 10-8. Create Missile Target dialog box

5. Optionally, specify the parameters for the target. In particular, set the following parameters:
   - Name. The name of the target point.
   - Launch Time. The simulation time at which the missile will be launched.
   - Incoming Heading. The heading, relative to the target, from which the missile will be launched.
   - Missile Type. An option from the list.
6. Click Create. The missile event is listed in the Missile Target window.

10.2.2. Editing Missile Target Events

You can edit the properties of a missile target event. You can also directly manipulate the target point.

To edit a missile target event:

1. Choose **Simulation** → **Munition Targets**. The Munition Target Settings dialog box opens (Figure 10-2).
2. Select the Missile Target page (Figure 10-7).
3. Select the missile fire event that you want to edit.
4. Click Edit. The Edit Missile Target dialog box opens.
5. Change the properties that you want to change.
6. Click Update.

10.2.3. Deleting Missile Target Events

To delete a missile target event:

1. Choose **Simulation** → **Munition Targets**. The Munition Target Settings dialog box opens (Figure 10-2).
2. Select the Missile Target page (Figure 10-7).
3. Select the missile fire event that you want to delete.
4. Click Delete.
5. Click Close.



You can also delete a target event by deleting the target point from the terrain.

10.3. Creating Munition Detonations

When weapons fire munitions, they detonate and cause damage to simulation objects, terrain, or both. You can also manually create munition detonations. They have the same effect as the simulated munitions they represent.



Munition detonations are not supported in aggregate-level scenarios.

To create a munition detonation:

1. Choose **Create** → **Munition Detonation**. The Create Detonation dialog box opens, as if you were creating a simulation object or tactical graphic. The cursor changes to a red cross-hair.
2. If you want to use the currently configured detonation type and result, click on the terrain where you want the detonation to occur. You can continue to click to create more detonations while in this mode.
3. If you want to change the munition type or result:
   a. In the Munition Type list, select the type of munition you want to detonate.
   b. In the Result list, select the type of detonation that you want to occur.
   c. Optionally, specify the location for the detonation.
   d. Click on the terrain where you want the detonation to occur. Click as many times as you want.
4. To exit detonation mode, close the Create Detonation dialog box or press **ESC**.



11. Setting Environment Conditions

This chapter explains how to set the time of day and weather conditions.

Introduction to Environment Conditions 11-3
Setting the Date and Time of Day 11-3
- Setting the Date 11-4
- Setting the Time of Day 11-5
- Choosing the Illumination Model 11-5

Specifying the Environment Conditions 11-6
- Adding an Environment Condition 11-7
- Editing an Environment Condition 11-8
- Deleting an Environment Condition 11-8

Setting Weather Conditions 11-9
- Setting the Wind Speed and Direction 11-9
- Setting the Ambient Air Temperature 11-10
- Setting Visibility (Fog) 11-11
- Specifying Precipitation Type and Intensity 11-12
- Specifying Cloud Cover 11-12

Creating Local Weather Zones 11-13
- Editing a Weather Zone's Environment 11-15

Creating and Editing Cloud Covers 11-17
- Editing Cloud Covers 11-20
- Enabling and Disabling Cloud Layers 11-20
- Removing a Cloud Cover 11-21
- Cloud Layer Parameters 11-21

Configuring Marine Conditions 11-22
- Enabling Marine Effects 11-23
- Configuring Marine Conditions 11-24
- Enabling Tidal Stream Wakes 11-26

Setting Ocean Quality 11-27
Configuring Breaking Waves 11-27
Setting the Thermocline 11-29

11.1. Introduction to Environment Conditions

You can set the date, time of day, weather, and marine conditions from the front-end and in plans. The date, time of day, and weather conditions are back-end settings, not merely front-end display effects. They affect simulation object sensors and line-of-sight. For example, if visibility is set to 4 kilometers, a simulation object will not see a simulation object that is 5 km away. If the time of day changes from daytime to nighttime, lighting conditions in the 3D view change accordingly.

Date and Time and Weather are options on the Commands menu in plans.

11.2. Setting the Date and Time of Day

You can set the date, time zone, and time of day.

- Date: When you create a scenario, you specify a date. However, you can change the date at any time while you are creating or running a scenario. The date at a location is advanced when the time at that location advances through midnight. When this happens, all locations to the east, up to the time zones at GMT + 12 hours, are in the advanced date; all locations to the west, up to the time zones at GMT -12 hours, are in the previous date. Thus there is effectively a date line at a longitude 12 hours from GMT.
- Time zone: You can change the time zone displayed in the Environmental Settings toolbar to any zone that you want. If you change the time zone in the Environment Conditions dialog box, the time displayed in the Environment Settings Toolbar changes to the current time for that time zone. However, the time at a given location on the earth does not change. The illumination at a location therefore does not change when the time zone displayed in the toolbar changes. Since the time zone at a location may not be the same as the time zone displayed in the toolbar, the illumination at that location may not match the time displayed in the toolbar.
- Time of day: The time of day advances as simulation time advances. However, they are not directly tied to each other. For example, if at the start of a simulation (simulation time 0:00:00:00), you set the time of day to 10:00 AM, after eight hours of simulation time passes, the time of day will be 6:00 PM. However, you could just as easily set the initial time of day to some other time or change it at will during the course of the simulation. These changes would not affect the elapsed simulation time in any way. They would just change the starting point for the advance of the time of day.

  VR-Forces uses an ephemeris model to calculate illumination based on the date and time of day. The amount of light affects visual sensors and scene lighting in Stealth view mode. Illumination at a location is based on the time of day at that location.

11.2.1. Setting the Date

To set the date:

1. Choose **Simulation** → **Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
2. Select the Date and Time page (Figure 11-1).

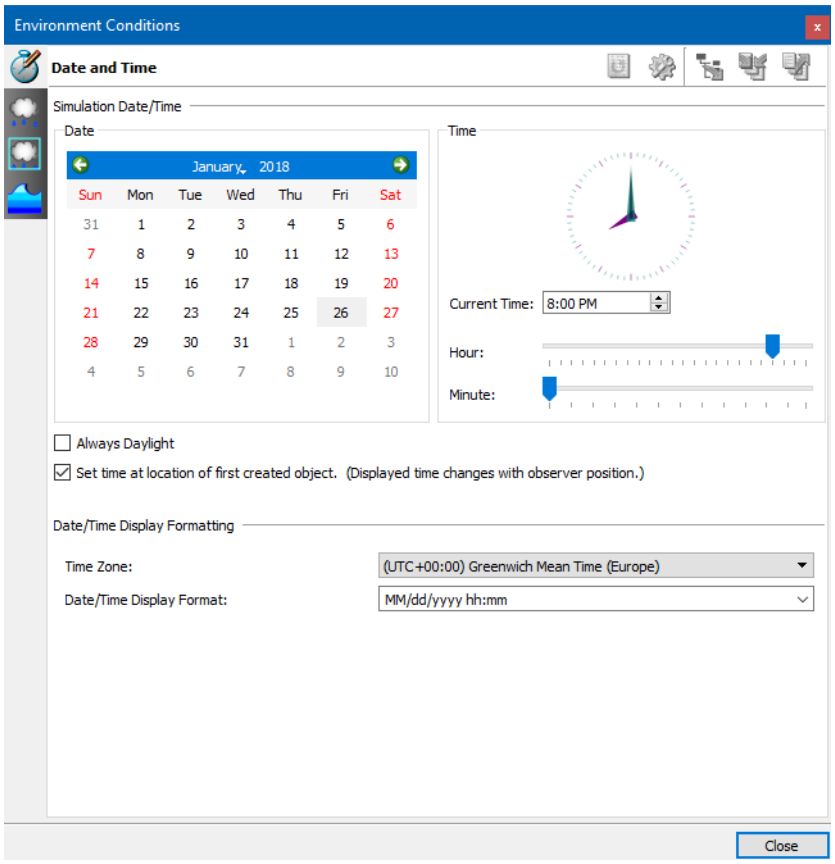


Figure 11-1. Date and Time page

3. To change the month, click the right or left arrows, or click the month and select a month from the list.
4. To change the year, click the right or left arrows, or click the year and type in a year or use the spin controls to select a value.
5. To specify the date, click on the date in the calendar.

11.2.2. Setting the Time of Day

As time advances, the time displayed on the Environment Settings Toolbar changes. The time of day setting is independent of simulation time. The default time zone is Greenwich Mean Time. You can change the time zone to the local zone or any of the zones in the Time Zone list.



If you want to set the time of day and time zone, set the time zone first.

To set the time of day:

1. Choose **Simulation** → **Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
2. Select the Date and Time page (Figure 11-1).
3. Set the time of day by changing the value in the Current Time box or by adjusting the Hour and Minute sliders.
4. Select the time zone in one of the following ways:
   - Select a time zone in the Time Zone list.
   - Select Use Local Time Zone. The time zone set for your computer is used.
5. Optionally, select a date/time display format. This changes the format used in the Environment Settings Toolbar.

11.2.3. Choosing the Illumination Model

Illumination affects visual sensors. Illumination can be on all of the time or it can be affected by the time of day. The illumination setting does not affect the front-end. It only affects back-end calculations. If the day/night illumination model is active (Always Daylight is not checked), the back-end calculates illumination based on a simulation object's location given current Greenwich Mean Time, offset by the time zone where the simulation object is located (based on longitude). This calculation does not take into account daylight savings time or localized time zone variations.

To specify the illumination model:

1. Choose **Simulation** → **Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
2. Select the Date and Time page (Figure 11-1).
3. Select or clear the Always Daylight check box.

11.3. Specifying the Environment Conditions

VR-Forces has a set of preconfigured environment conditions. Each environment condition has settings for:

- Weather:
  - Ambient air temperature. (VR-Forces does not use this information. It just publishes it for use by other applications.)
  - Wind speed and direction.
  - Cloud cover.
  - Visibility.
  - Precipitation type and intensity.
  - Fog height and color.
- Marine:
  - Sea swell.
  - Sea state.
  - Surface transparency.
  - Underwater visibility.
  - Choppiness.
  - Surge depth.

Conditions range from clear to stormy and let you quickly set the weather without needing to adjust all the various weather parameters. However, if you want to, you can change any of the individual weather parameters. Changing a weather or marine setting does not affect the saved settings unless you specifically overwrite them.

You can save and import environment settings. Environment settings support the standard VR-Forces settings behaviors, as described in "Managing VR-Forces Settings," on page 3-15.

You can add your own environment conditions.



The settings on the Weather page always reflect the current settings for environment conditions. They do not necessarily match any of the preconfigured environment conditions as listed in the Apply Environment Conditions list except immediately after a preconfigured environment condition is applied. In other words, the Apply Environment Conditions list lets you apply environment conditions, but is not intended to summarize the status of the current environment conditions.

To specify an environment condition:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-2).

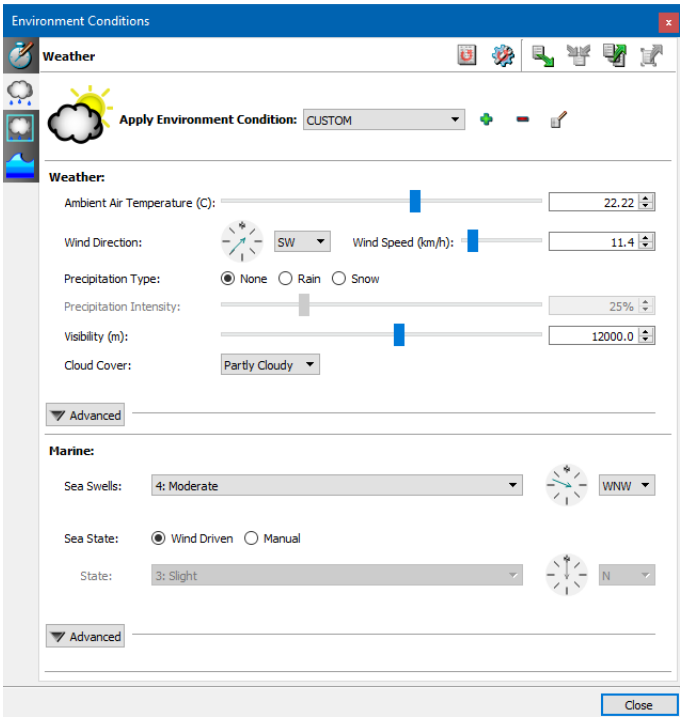


Figure 11-2. Weather page

3. In the Apply Environment Condition list, select the condition you want.

> After you select a preconfigured environment condition, if you change any of the settings, the Apply Environment Condition list says CUSTOM instead of the condition that you chose.

11.3.1. Adding an Environment Condition

If you add an environment condition, it gets added to the Apply Environment Condition list.

To add an environment condition:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-2).
3. Change the weather settings to the ones you want for the new condition.
4. Click the Add button (✚). The Name an Environment Condition dialog box opens.
5. Type a name for the new condition.
6. Click OK.

11.3.2. Editing an Environment Condition

You can change the settings for an environment condition. Editing a condition essentially means overwriting a condition with the current set of weather and marine conditions.

To edit an environment condition:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-2).
3. Optionally, in the Apply Environment Condition list, select a condition you want to use as your starting point.
4. Change whatever conditions you want.
5. Click the Edit button ( ). The Select an Environment Condition dialog box opens.
6. In the Select Name list, select the condition you want to overwrite.
7. Click OK.

11.3.3. Deleting an Environment Condition

If you delete one of the built-in environment conditions, you can restore it from the factory settings.

To delete an environment condition:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-2).
3. Click the Delete button ( ). The Select an Environment Condition dialog box opens.
4. In the Select Name list, select the condition you want to delete.
5. Click OK.

11.4. Setting Weather Conditions

You can set the following weather conditions:

- Cloud cover.
- Visibility.
- Precipitation.
- Wind speed and direction.
- Ambient air temperature.

When you specify weather, the back-end sends out state updates with these conditions. Simulation objects respond to them only if their model has the ability to do so.

11.4.1. Setting the Wind Speed and Direction

You can set the wind speed and direction. If you do not specify the sea state manually, it is set based on the wind conditions. (The maximum wind speed that you can set is 360 km/h.) The wind direction affects the angle at which precipitation falls. Wind speed and direction affect particle systems (dust clouds, smoke, and so on).

To set wind speed and direction:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-3).

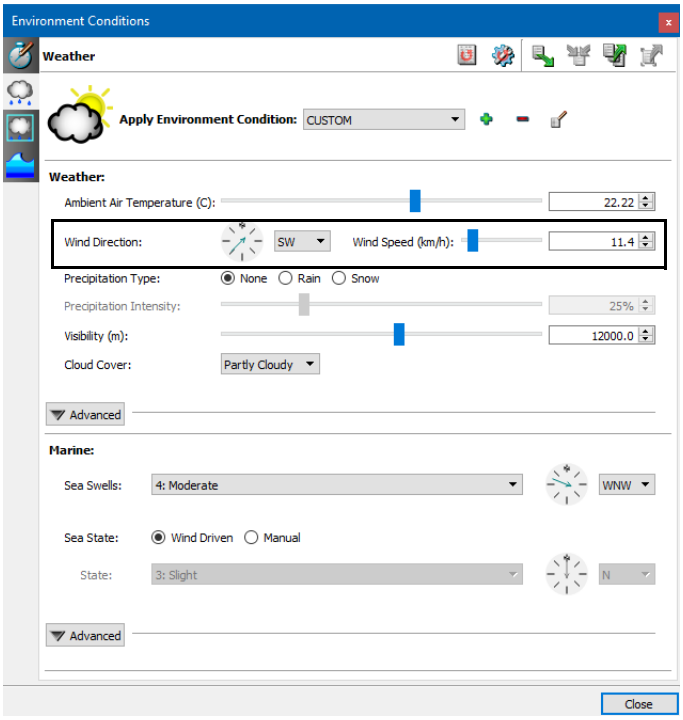


Figure 11-3. Weather page

3. Drag the Wind Direction indicator to the desired compass point or select an option from the list.
4. Drag the Wind Speed slider or enter a value in the box.

11.4.2. Setting the Ambient Air Temperature

You can specify the ambient air temperature. VR-Forces does not do anything with this information. It is sent over the network for the use of applications that need to model air temperature.

To set the ambient air temperature:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-3).
3. Drag the Ambient Air Temperature slider or enter a value in the box.

11.4.3. Setting Visibility (Fog)

The visibility setting determines the denseness of the fog effect, which determines how far you can see.

To set visibility:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-3).
3. Adjust the Visibility slider or enter a value in the box.

Configuring Fog

You can set the fog height and color. The fog height affects how much sky the observer can see when looking up from ground level. For example, if you set the fog height to 50m and look up, you will see some amount of sky above you, and to the sides it will be very gray. If you then set it to 10m and look up, you will see mostly blue sky above you, but still quite a bit of gray to the sides.



Fog settings are not modeled in the back-end. Therefore, they are not available when you set weather conditions in a plan.

To configure fog:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-3).
3. In the Weather section of the page, click Advanced. The dialog box expands to show options for fog.
4. To set the fog height, adjust the slider or enter a value in the box.
5. To set the fog color:
   a. Click the color swatch. A color picker dialog box opens.
   b. Select the color you want to apply to the fog. It is applied immediately, so you can easily see the effect of the new color and change it if you want to.
   c. Click OK.

11.4.4. Specifying Precipitation Type and Intensity

VR-Forces supports display of rain and snow. You can configure the intensity of precipitation.

> Precipitation has no effect on the simulation. It is only a visual effect. Snow does not accumulate.

To specify the precipitation type and intensity:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-3).
3. Select a Precipitation Type. If you select rain or snow, the Precipitation Intensity setting becomes editable.
4. Adjust the Precipitation Intensity slider or enter a value in the box. The range is 0% through 100%.

11.4.5. Specifying Cloud Cover

VR-Forces simulates varying amounts and types of cloud cover, from clear, to cloudy, to thunderstorm. When you choose an environment condition, it includes an appropriate degree of cloud cover. You can also change the cloud cover at any time.

To specify the cloud cover:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-3).
3. Select an option from the Cloud Cover list.

11.5. Creating Local Weather Zones

You can create local weather zones. Each zone can have different weather conditions and simulation objects can respond to those conditions as they move from zone to zone.

Weather zones can be boxes, free form areas, circles, and ellipses. You can create them from the Tactical Graphics Palette just like you would create the comparable tactical graphics and you can create box weather zones from the Environment Conditions dialog box, Local Weather Zones page.

Weather zones have a priority. If two zones overlap and one has a higher priority than the other, a simulation object in the overlapping area responds to the weather conditions in the higher priority zone. If overlapping zones have the same priority, the one that gets discovered first on the network is used.

By default, weather zones are placed on the Weather overlay. Weather zones display a graphic that represents the weather conditions. Figure 11-4 illustrates a weather zone set to the Thunderstorm environmental condition. You can move and resize them just as you can move and resize other tactical graphics.

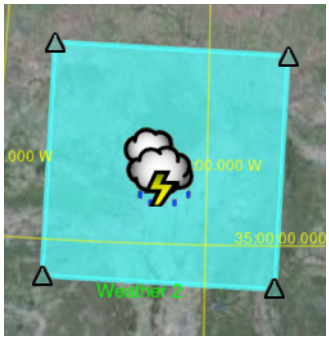


Figure 11-4. Weather zone

To create a local weather zone from the Local Weather Zones page:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Local Weather Zones page (Figure 11-5).

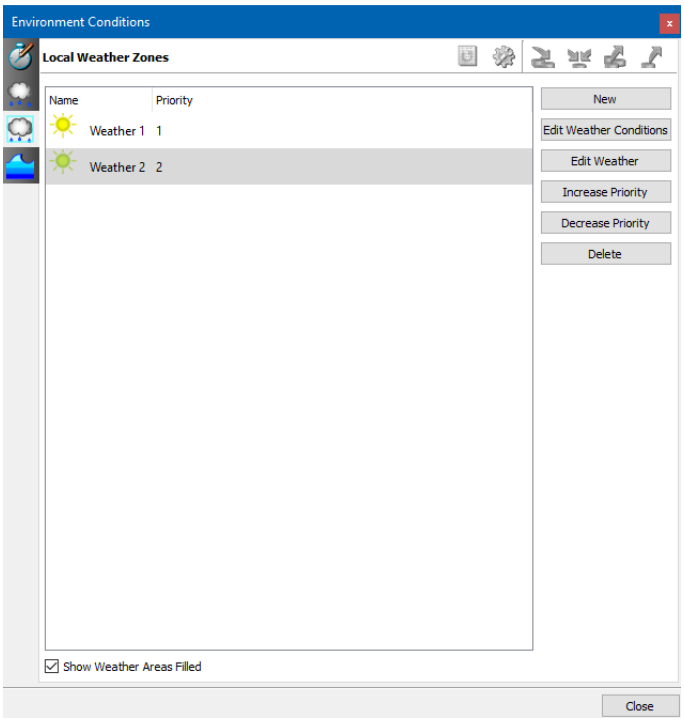


Figure 11-5. Local Weather Zones page

3. Click New. A weather zone is added to the scenario and the Create Weather dialog box opens.

The default size for a weather zone is 100,000 meters per side, so on small terrains, the zone might cover the terrain. If this is a problem, edit the Length and Width in the Create Weather dialog box to reduce it to a manageable size.

---

4. Optionally, edit the various parameters for the weather zone. In particular, select an environmental condition or edit the environment to specify the conditions you want to apply to this weather zone. For details about editing the environment, please see "Editing a Weather Zone's Environment," on page 11-15.
5. Click Create.
6. On the Local Weather Zones page, select the Show Weather Areas Filled check box to fill all weather zones. Clear the check box to show weather zones as outlines.

11.5.1. Creating a Weather Zone as a Tactical Graphic

You can create all types of weather zones from the Tactical Graphics Palette. They are listed as:

- Weather (Area).
- Weather (Box). This is the same weather zone as gets created from the Local Weather Zones page.
- Weather (Circle).
- Weather (Ellipse).

The procedures for drawing them are the same as for other similarly shaped graphics, as described in "Placing Objects," on page 14-15, "Drawing Circles and Ellipses," on page 40-4, and "Drawing Boxes," on page 40-4. The main difference is that when you create a weather zone, the Edit dialog box always opens so that you can specify the weather-specific parameters.

11.5.2. Editing a Weather Zone's Environment

You can edit a weather zone's environment when you create it or after you create it. You can assign it one of the preconfigured environmental conditions or change individual environment settings. You can edit all types of weather zones from the Local Weather Zones page. It does not matter how you created them. For details about preconfigured environmental conditions, please see "Specifying the Environment Conditions," on page 11-6.

To edit a weather zone's environment:

1. Do one of the following:
   - In the Create Weather dialog box, click Edit Environment. The Weather dialog box opens (Figure 11-6).
   - On the Local Weather Zones page of the Environment Conditions dialog box, select the weather zone you want to edit and click Edit Weather Conditions. The Weather dialog box opens.

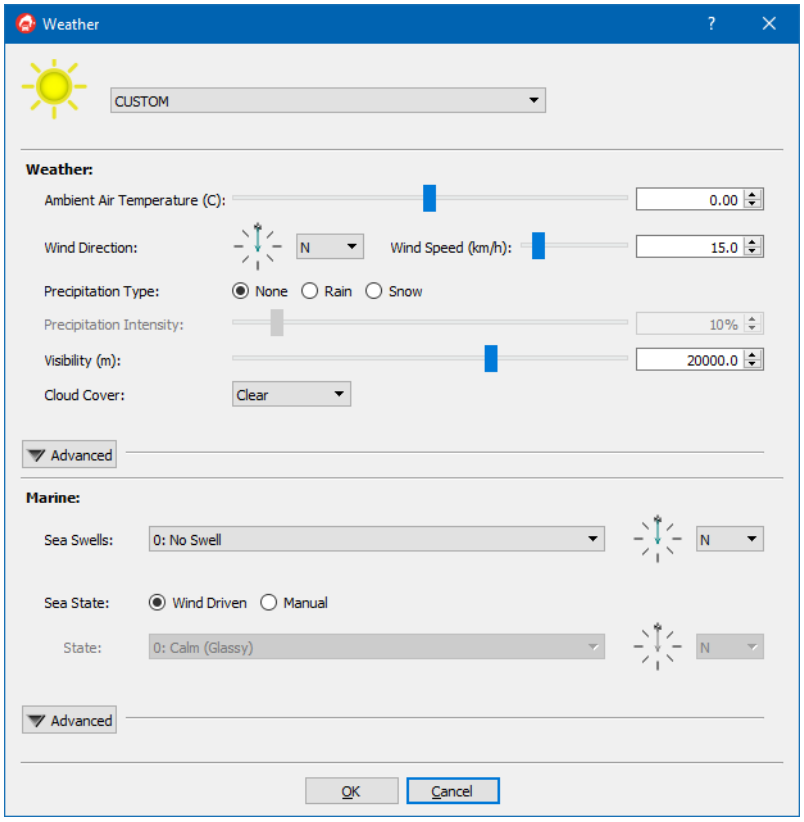


Figure 11-6. Set Weather dialog box

2. Change weather conditions as desired. They are the same as those on the Weather page of the Environment Conditions dialog box. For details, please see "Setting Weather Conditions," on page 11-9.
3. Click OK.

11.6. Creating and Editing Cloud Covers

As part of specifying environment conditions, you can specify a cloud cover. Cloud covers consist of zero or more cloud layers. For each cloud cover you can specify the cloud layers and their parameters. You can create new cloud covers and edit the existing cloud covers.

Cloud layers are created (and recreated) at the location of the current camera view when VR-Forces is started and whenever a terrain is closed or opened.

Changes to cloud covers are saved automatically. Like other settings, you can restore them to factory settings and revert to startup settings.



The cloud layers described in this section are not the same thing as cloud layer tactical graphics, although they have a similar effect on the display of clouds in the 3D view. For information about cloud layer tactical graphics, please see "Creating Cloud Layers," on page 40-16.

To create a cloud cover:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Cloud Cover Editor page (Figure 11-7).

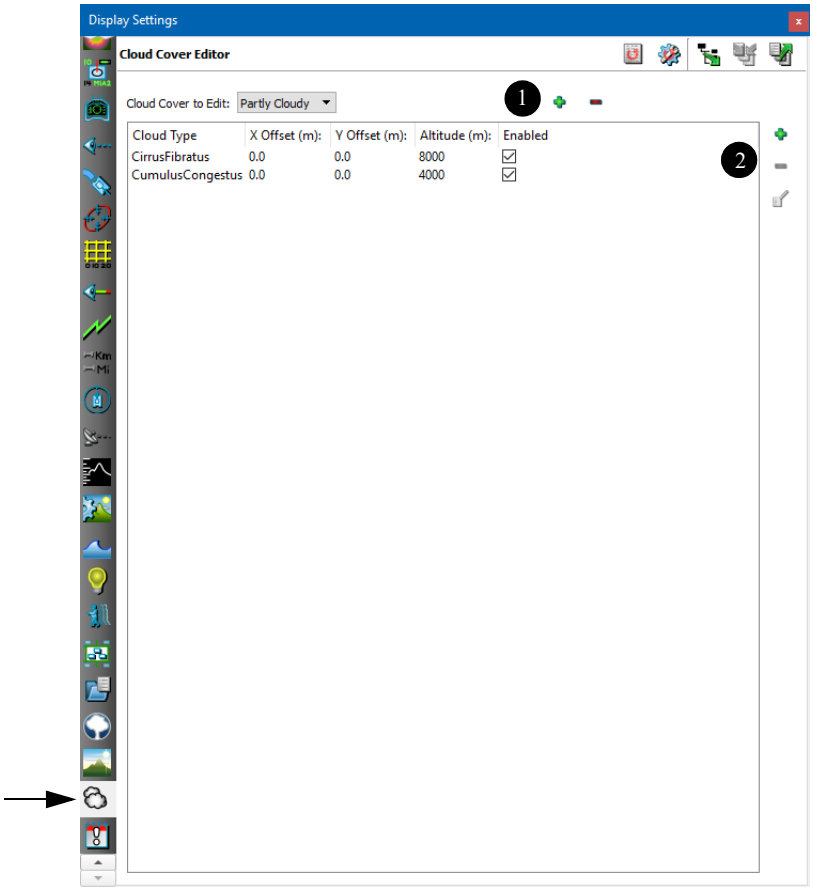


Figure 11-7. Cloud Cover Editor page

3. Click the Add button (+) to the right of the Cloud Cover to Edit list (callout 1 in Figure 11-7). The Create Cloud Cover dialog box opens.
4. Type a name for the new cloud cover.
5. Click OK. The cloud cover is added to the list. The cloud cover selected on the Environment Conditions page changes to the new cloud cover and the VR-Forces window displays the new cloud cover. (A new cloud cover does not have any cloud layers yet, so the sky is clear.)
6. Click the Add button (+) to the right of the Cloud Layers window (callout 2 in Figure 11-7). The Add Cloud Layer dialog box opens (Figure 11-8).

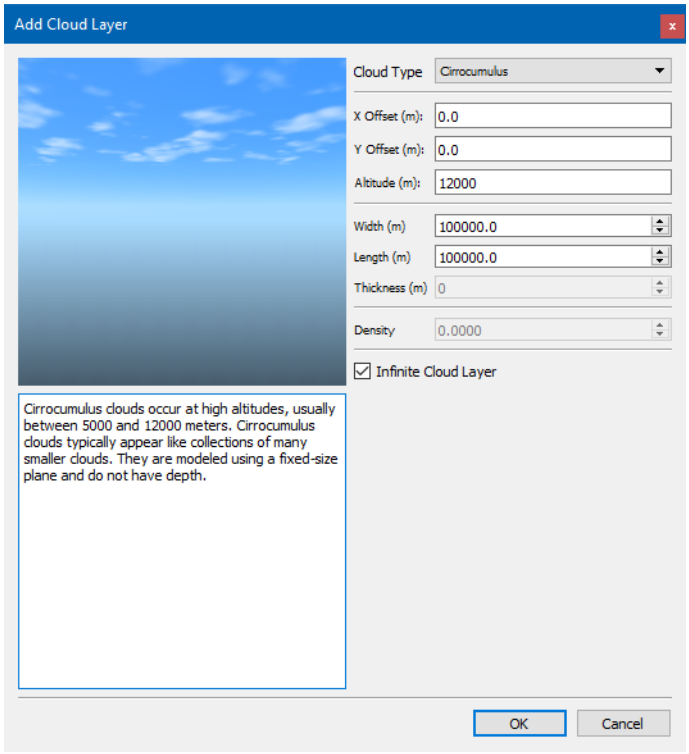


Figure 11-8. Add Cloud Layer dialog box

7. In the Cloud Type list, select a type of cloud to add as a cloud layer.
8. Edit the parameters of the cloud layer as desired.
9. Click OK.
10. Add additional cloud layers as desired.

11.6.1. Editing Cloud Covers

You can add and remove cloud layers from a cloud cover.

To edit a cloud cover:

1. In the Cloud Cover to Edit list, select the cloud cover you want to edit. The Cloud Layer window lists the layers in this cloud cover.
2. To add a cloud layer to this cloud cover:
   a. Click the Add button (+) to the right of the Cloud Layers window (callout 2 in Figure 11-7). The Add Cloud Layer dialog box opens.
   b. In the Cloud Type list, select a type of cloud to add as a cloud layer.
   c. Edit the parameters of the cloud layer as desired.
   d. Click OK.
3. To remove a cloud layer from this cloud cover:
   a. Select the cloud layer you want to delete.
   b. Click the Delete button (−) to the right of the Cloud Layers window. The cloud layer is removed from the cloud cover and the cloud cover in the VR-Forces window is updated.
4. To edit a cloud layer:
   a. Select the cloud layer you want to edit.
   b. Click the Edit button (✎) to the right of the Cloud Layers window. The Edit Cloud Layer dialog box opens (similar to Figure 11-8).
   c. Edit the parameters of the cloud layer as desired.
   d. Click OK.

11.6.2. Enabling and Disabling Cloud Layers

You can disable cloud layers without removing them from a cloud cover.

To enable or disable cloud layers:

1. In the Cloud Cover to Edit list, select the cloud cover you want to edit. The Cloud Layer window lists the layers in this cloud cover.
2. Select the cloud layer you want to enable or disable.
3. Select or clear the check box in the Enabled column. The cloud display in the VR-Forces updates.

11.6.3. Removing a Cloud Cover

To remove a cloud cover:

1. In the Cloud Cover to Edit list, select the cloud cover you want to remove.
2. Click the Delete button (−) to the right of the Cloud Cover to Edit list (callout 1 in Figure 11-7).

11.6.4. Cloud Layer Parameters

The Edit Cloud Layer dialog box describes each of the available cloud layers. For each cloud layer, you can specify the following parameters:

- X Offset. An offset from the camera location in X used when the layer is created.
- Y Offset. An offset from the camera location in Y used when the layer is created.
- Altitude (Z). The Z coordinate is the center for layers that have thickness. It is the Z value for layers that are planes.
- Length. The length of the cloud layer
- Width. The width of the cloud layer
- Altitude. The height above terrain at the camera location at which the base of the cloud layer is placed. (The height of the base is automatically updated if the camera moves a large distance to recalibrate to new terrain heights).
- Thickness: The height of the cloud layer above the base height.
- Density. Density controls how much of the area of the cloud layer is filled with clouds. Specified as a number between 0.0000 and 5.0000.
- Whether the cloud layer is infinite or not.

11.7. Configuring Marine Conditions

VR-Forces supports a variety of dynamic ocean effects (Figure 11-9), including:

- Douglas Sea State. This includes wind waves (wind sea), swell character, and the directions of each. VR-Forces allows each to be separately configured.
- Wave chop.
- Surface transparency. The ability to see through the water from above sea level.
- Underwater visibility. The ability to see underwater.
- Swell.
- Surge depth. When bathymetry data is present, this is the depth of the water at which full storm effects take place. As the water gets shallower, it gets calmer.
- Tidal current speed and direction. If tidal stream wakes are enabled, they are automatically applied to streamed buoys and beacons.



Figure 11-9. Ocean effects

Enabling dynamic ocean effects can affect performance.

11.7.1. Enabling Marine Effects

Marine effects can reduce performance.

To enable marine effects (dynamic ocean):

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 11-10).

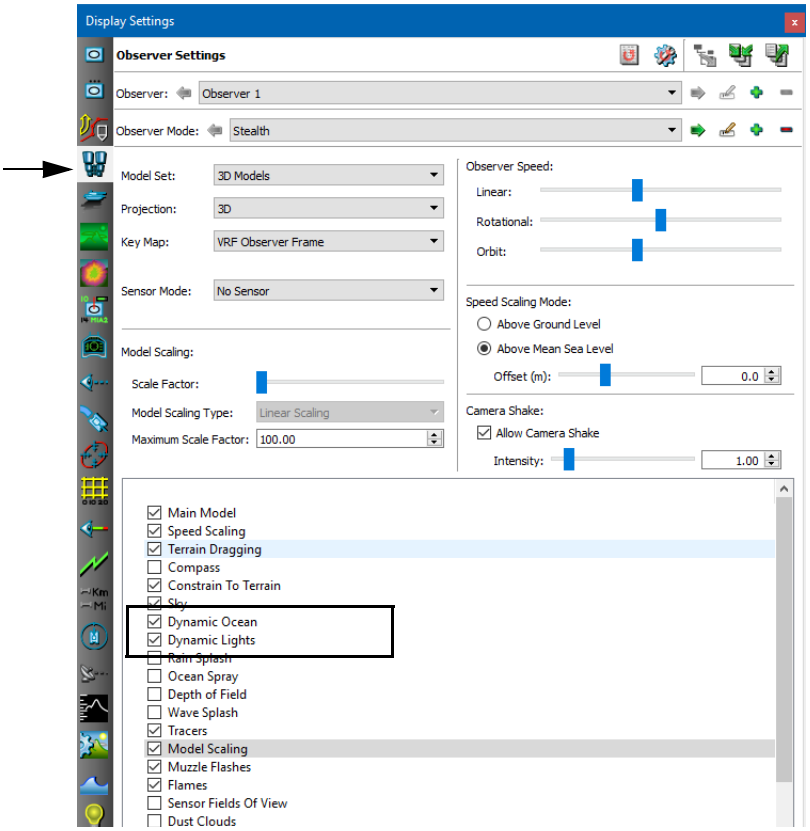


Figure 11-10. Observer Settings page

3. Select or clear the Dynamic Ocean check box.

> You can also enable or disable dynamic ocean by choosing **Observer** → **Dynamic Ocean**.

11.7.2. Configuring Marine Conditions

The preconfigured environment conditions include marine conditions. You can also customize the marine settings at any time.

To configure marine effects:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 11-11).

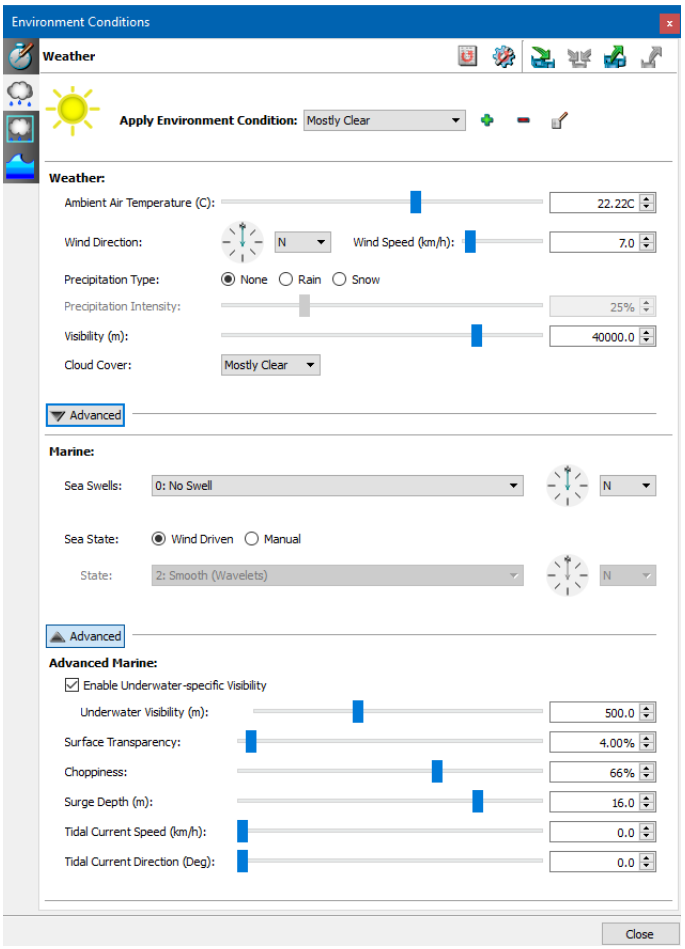


Figure 11-11. Weather page

3. If the Marine section is not expanded, click Advanced. The dialog box expands to show additional configuration options.

4. Change settings as desired. Table 11-1 describes the marine setings.

Table 11-1: Advanced marine settings

Parameter	Description
Sea Swells	The amount of rise and fall of waves independent of wind. Choose from 10 fixed swell states. You can also set the compass direction from which swells originate.
Sea State	Determined by the wind direction and speed or set manually. If manual, one of the swell states for the Douglas sea state scale. You can also set the compass direction from which sea state originates.
Advanced Parameters	
Surface Transparency	The depth at which VR-Forces begins to apply transparency to the water surface if the terrain has bathymetry or underwater polygons. This setting helps reduce Z-fighting along the shoreline. It does not mean that you can see some distance into the water when the observer is over deep water. The maximum value represents transparency to 500 meters.
Underwater visibility	The distance you can see underwater if the observer is underwater.
Choppiness	The percentage of turbulence on the water.
Surge depth	When bathymetry data is present, this is the depth of the water at which full storm effects take place. As the water gets shallower, it gets calmer.
Tidal Current Speed	The speed of the tidal current. This affects the display of tidal wakes on buoys and beacons. Default: 0.
Tidal Current Direction	The direction of the tidal current flow. This affects the display of tidal wakes on buoys and beacons.

11.7.3. Enabling Tidal Stream Wakes

Tidal stream wakes create wake effects around buoys and beacons based on the movement of the tides. Tidal stream wakes are supported as stand-alone models. They are automatically attached to streamed buoys and beacons. VR-Forces includes wake models for each of the supported buoy and beacon models. They are named for the buoy or beacon type they support, for example, BuoysPillarWake or BeaconsLatticeWake. You may want to edit the **wakeLength** parameter to provide the best visuals for your terrain.

Tidal stream wakes are disabled by default.



Even if tidal stream wakes are enabled, you will not see any wakes unless you also specify a tidal current, which is 0 by default. For details about setting tidal current, please see "Configuring Marine Conditions," on page 11-24.

To enable or disable tidal stream wakes:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 11-12).

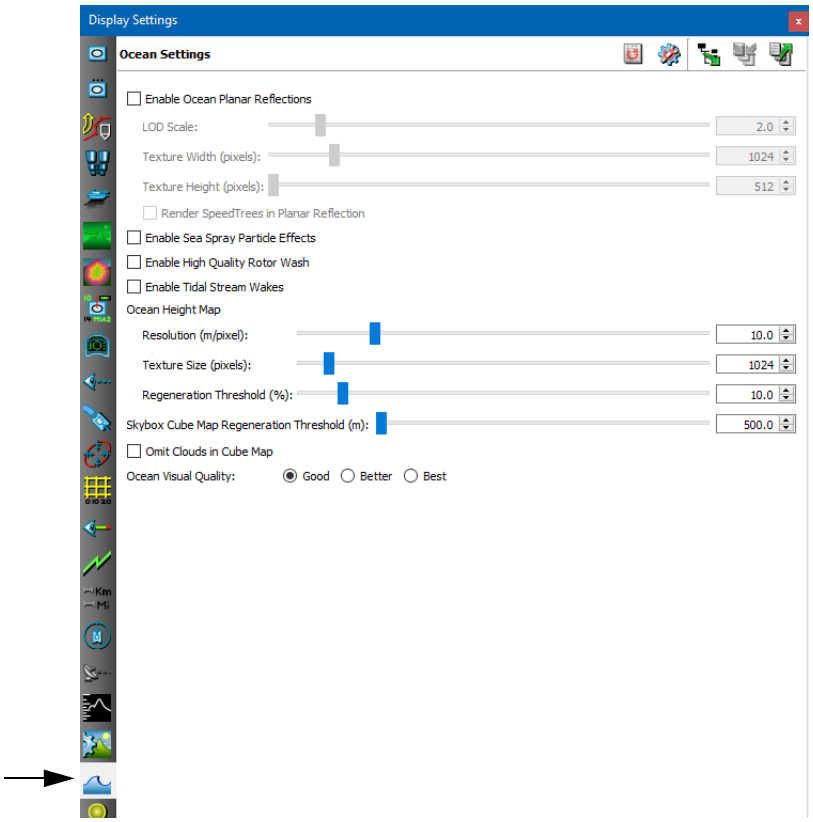


Figure 11-12. Ocean Settings

3. Select or clear Enable Tidal Stream Wakes.

11.7.4. Setting Ocean Quality

You can adjust the overall quality of visualization of the ocean. Increasing ocean quality decreases performance. Best quality may affect overall performance of your computer. Therefore this setting is primarily for use in demonstrations rather than large scenarios.

To set ocean quality:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 11-12).
3. Select one of the Ocean Visual Quality options.

11.7.5. Configuring Breaking Waves

VR-Forces can simulate the effect of waves breaking along the shore line. This feature uses shapefile data that defines the direction of the breaking waves. The *MAK Earth (online).mtf* terrain includes sample data for breaking waves. To see it, view the island of Oahu, Hawaii.

You can set the amplitude of breaking waves. By default, the amplitude is 0.



This is considered an experimental feature. It may be useful, but may affect performance.

To configure breaking waves:

1. Choose **Settings** → **Experimental Features**. The Experimental Features dialog box opens (Figure 11-13).

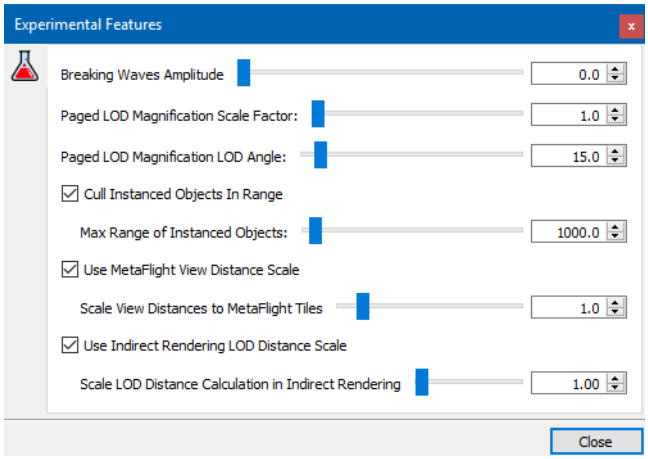


Figure 11-13. Experimental Features dialog box

2. Adjust the Breaking Waves Amplitude slider or type a value in the box. The maximum value is 10.0.

11.8. Setting the Thermocline

Thermoclines are layers of water that have different temperatures and, therefore, different density. Sound transmission varies as it passes through the thermocline. In VR-Forces, sonar effectiveness is reduced as it passes through thermocline boundaries. By default, VR-Forces does not set any thermoclines. You can specify the depth of up to five thermoclines.

To specify a thermocline:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Thermoclines page (Figure 11-14).

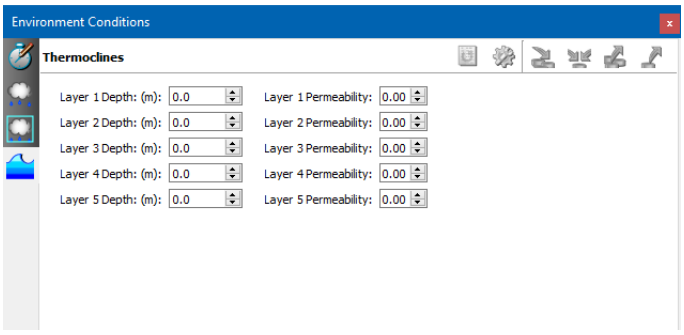


Figure 11-14. Thermoclines page

3. For each thermocline that you want to specify, type or select a value in the depth box and the permeability box.



12. Scenario Files

This chapter describes the files that get created when you create a scenario.

Introduction 12-2
The Scenario File 12-3
Scenario Parameters 12-4
Specifying Pathnames for Scenario Files 12-7
Changing the Terrain Database for a Scenario 12-8
The Plan File 12-9
The Object Map File 12-9
Editing the Scenario Objects File 12-10
Saving Default Parameters to the Scenario Objects File 12-10
The Scenario Extras File 12-10
Temporary Scenario Directories 12-11

12.1. Introduction

When you create and save a scenario, VR-Forces stores information about the scenario in the following files:

- The scenario file, named *scenario_name.scn*, lists the locations of the other scenario files, the terrain database, and the simulation model set file, and sets several parameters for the scenario.
- The scenario objects file, named *scenario_name.oob*, stores identification and state information for simulation objects, control objects, and published tactical graphics.
- The object map file (*scenario_name.omp*) maps objects to the back-ends on which they were created.
- The plan file (*scenario_name.pln*) stores individual plans and global plans.
- The scenario extras file (*scenario_name.xtr*) lists the forces that each force is hostile to and it stores spawn pattern templates.
- The scripts file (*scenario_name.spt*) stores scenario-specific scripts.
- The overlay file (*scenario_name.ovl*) lists tactical graphics that are not published. Published tactical graphics are in the scenario objects file.
- The selection groups file (*scenario_name.sgr*) stores the selection groups in the scenario and their members.
- The saved views file (*scenario_name.osrx*) stores observer views, if any. It can also specify a startup view.
- The scenario-specific GUI settings file (*scenario_name.gui_settings*) contains GUI settings that only apply to this scenario.

By default, scenarios are saved in a compressed zip archive with the extension *.scnx*. If you examine a compressed scenario file in an unzipping application, you will see that it contains the individual files described in the previous paragraphs.

For the most part, you will not want to edit scenario files by hand. In some cases, we recommend that you not edit them. However, this chapter explains how to edit them if you need to.

12.2. The Scenario File

When you create a scenario, you can set its parameters in the New Scenario dialog box (Figure 12-1).

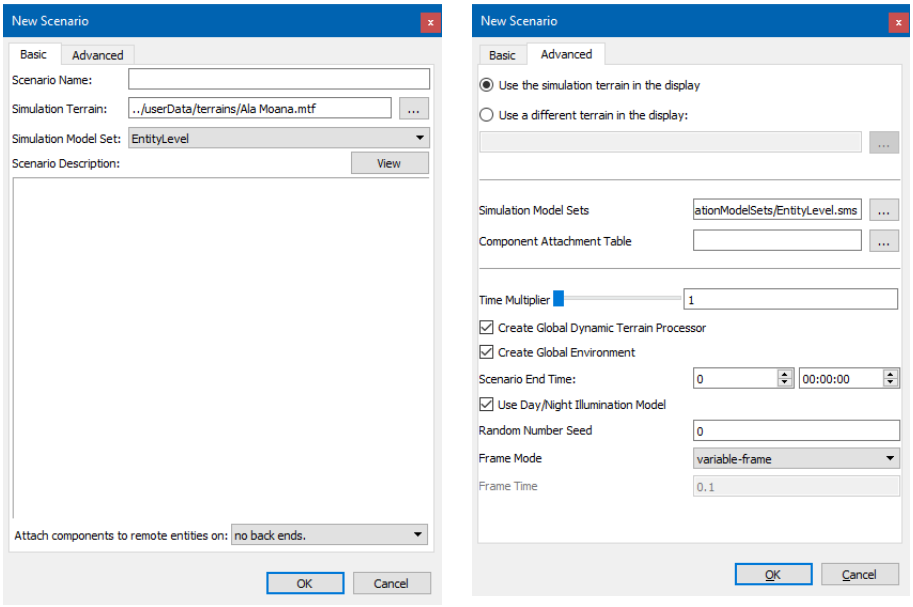


Figure 12-1. New Scenario dialog box

The default values for scenario settings in the New Scenario dialog box are drawn from the file *./appData/settings/vrfGui/default-scenario.tmpl*. If you want to change the default settings, you can edit that file.

12.2.1. Scenario Parameters

A typical scenario file looks like the following (assume a scenario file called *myscenario.scn*):

```
(Scenario
  (Component-Attachment  0)
  (Component-Attachment-Table  "")
  (Terrain-Database  "..\userData\terrains\Ground-db.mtf")
  (Gui-Terrain-Database  "..\userData\terrains\Ground-db.mtf")
  (Order-Of-Battle  "myscenario.oob")
  (Scenario-Scripts  "myscenario.spt")
  (Orbat "myscenario.orb"
  (Plan  "myscenario.pln")
  (Overlay  "myscenario.ovl")
  (SelectionGroups  "myscenario.sgr")
  (Object-Map  "myscenario.omp")
  (Scenario-extras  "myscenario.xtr")
  (time-multiplier  1.000000)
  (auto-reorganize False)
  (random-number-seed  0)
  (frame-mode  "variable-frame")
  (frame-time  0.100000)
  (run-duration-time 0)
  (sim-time  0.000000) //no loner used
  (exercise-start-time  0.000000) // no longer used
  (additional-opds )
  (Simulation-Model-Set-Files "$(DATA_DIR)\simulationModelSets\
    EntityLevel.sms")
  (scenario-name  "myscenario"
  (scenario-data
    (ChangeOnSave_ComponentAttachmentTable  "")
    (Create_Global_Dynamic_Terrain  "1")
    (Create_Global_Environment  "1")
    (DefaultObserverView  "1 - Absolute")
    (GuiObserverViews  "myscenario.osrx")
    (GuiScenarioSettings  "myscenario.gui_settings")
    (HiddenTerrainViews  "AlaMoana PVD:2D;AlaMoana Port:3D")
    (Language  "en")
    (ScenarioDescription  "Some text that describes the scenario.")
    (ScenarioExtentInformation  "-2.81258e+006,-4.33259e+006,
        3.72903e+006,938.957")
    (Set_Scenario_Start_Time_At_Local_On_First_Object_Creation  "36000")
    (UseDayNightModel  "1")
  )
  (version  3.200000)
)
```



Paths in the scenario files are relative to the application directory, for example, *./bin64*.

The Starting Time of Day is stored in the scenario objects file, not the scenario file.

Table 12-1 describes the parameters in a scenario file.

Table 12-1: Scenario file parameters (Sheet 1 of 3)

Parameter	Description
Component-Attachment	Specifies whether or not attachment of components to remote simulation objects is allowed in the scenario. The options are: ◆ 0: no component attachment. ◆ 1: the first back-end manages all remote components. ◆ 2: remote component attachment is load-balanced on all back-ends. Although you can set this value in the New Scenario and Load Scenario dialog boxes, it is not saved in the scenario file. To force a specific value, you must edit the scenario file.
Component-Attachment-Table	Specifies the component attachment table to use if attachment of components to remote simulation objects is allowed.
Terrain-Database	Specifies the terrain database to be loaded by the back-end.
Gui-Terrain-Database	Specifies the terrain database loaded in the front-end.
Order-Of-Battle	Specifies the scenario objects file.
Scenario-Scripts	Specifies the file that has scenario-specific Lua scripts.
Orbat	Specifies the order of battle file.
Plan	Specifies the plan file.
Overlay	Specifies the overlay file.
SelectionGroups	Specifies the selection groups file.
Object-Map	Specifies the object mapping file.
Scenario-Extras	Specifies the extras file, which records force hostility relationships, and stores spawn pattern templates. It may be used for other purposes in the future.
time-multiplier	Specifies whether a scenario runs in real time or faster than real time. A value greater than 1 runs the simulation faster than real time. You can set this value dynamically using the Time Multiplier toolbar. If you are running a scenario using HLA Time Management, it is strongly recommended that you set time-multiplier to 1.
auto-reorganize	Lets you specify the default behavior for unit reorganization. Set it to True or False. For information about reorganization, please see "How Units Are Organized," on page 21-3.

Table 12-1: Scenario file parameters (Sheet 2 of 3)

Parameter	Description
random-number-seed	Sets the default random number seed. It provides a starting point for the generation of random numbers needed by VR-Forces. Enter an integer between 0 and the largest integer permitted on your system.
frame-mode	Sets the frame rate mode to one of the following: ◆ Variable-Frame Run-To-Complete (`variable-frame`) ◆ Fixed-Frame Best-Effort (`fixed-frame`) ◆ Fixed-Frame Run-To-Complete (`fixed-frame-run-to-complete`). These frame rate modes are described in "Exercise Clock Modes," on page 3-12.
frame-time	Specifies the length of a frame, in seconds. If frame-mode is set to `fixed-frame` or `fixed-frame-run-to-complete`, you must set the frame time to a non-zero value. A value of zero for frame time will prevent simulation time from advancing in either of these modes.
run-duration-time	Specifies how long the scenario should run before automatically stopping.
sim-time	This parameter is not supported in scenarios created after VR-Forces 4.0.3 and is here for backwards compatibility
exercise-start--time	This parameter is not supported in scenarios created after VR-Forces 4.0.3 and is here for backwards compatibility
additional-opds	Specifies custom OPD files to load (for backwards compatibility).
Simulation-Model-Set-Files	Specifies a list of SMS files to load. The path is relative to the default data directory (*./data*) or a user-specified data directory.
scenario-name	An optional parameter that assigns a name to the scenario. It is independent of the file name and is only used in some message dialog boxes.

Table 12-1: Scenario file parameters (Sheet 3 of 3)

Parameter	Description
scenario-data	A block of data with the following additional parameters: ◆ ◆ Create_Global_Dynamic_Terrain specifies whether or not the global dynamic terrain processor should be enabled. If this is disabled, dynamic terrain does not work unless you create a dynamic terrain area. ◆ Create_Global_Environment specifies whether or not the scenario supports environmental conditions. ◆ DefaultObserverView specifies the observer view to use at startup. ◆ GuiObserverViews specifies the saved views file for the scenario. ◆ GuiScenarioSettings specifies the *.gui_settings* file. It records scenario-specific GUI settings. ◆ HiddenTerrainViews lists views associated with the scenario terrain that are not displayed in the Observer Views Panel because they were cleared from the Observer Views Panel. ◆ ScenarioDescription is the text of the scenario description. ◆ ScenarioExtentInformation lists the extents of the scenario for the zoom to extents option. ◆ Set_Scenario_Start_Time_At_Local_On_First_Object_Creation specifies the start time for the scenario, in seconds. ◆ UseDayNightModel specifies whether or not to use the day/night illumination model.
version	Specifies versioning information for the scenario file. Do not edit this parameter.

12.2.2. Specifying Pathnames for Scenario Files

Pathnames in a scenario file can be absolute or relative. When VR-Forces saves a new scenario, it creates pathnames that are relative to the application directory. If you think you might need to move a scenario to another computer or a new directory structure, change the paths to be relative to the *./bin64* directory. To change pathnames, edit the scenario file in a text editor. VR-Forces preserves the new paths when it saves the scenario.

If you plan to run multiple front-ends and back-ends on multiple machines, particularly if they use different operating systems, you must change the pathnames for the terrain database and object parameter database to relative paths or simple pathnames.

Configuring How Paths are Interpreted

The following parameters in *vrfSim.mtl* determine how pathnames for the object parameter database and terrain database are resolved:

- defaultParameterDatabasePath specifies the path to search for OPD files when an explicit path is not specified in a scenario file.
- defaultTerrainDatabasePath specifies the path to search for terrain database files when an explicit path is not specified in a scenario file.

When the back-end loads a scenario file, it looks for the object parameter database using the following search order:

1. If the path is absolute, load the file in the specified path.
2. If the path is relative, try to load the file relative to the application path (*./bin64/vrfSimprotocol*).
3. Look in the directory specified by defaultParameterDatabasePath.

When the back-end loads a scenario file, it looks for the terrain database using the following search order:

1. If the path is absolute, load the file in the specified path.
2. If the path is relative, try to load the file relative to the application path.
3. Look in the directory specified by defaultTerrainDatabasePath.
4. Use the path in the user-defined environmental variable VRF_TERRAIN_-PATH, if it is specified.

12.2.3. Changing the Terrain Database for a Scenario

When you create a new scenario, you must choose terrain databases for the front-end and back-end. By default, the back-end uses the same database as the front end.

You can edit the pathname of a terrain database in the scenario file or in the Scenario Properties dialog box. However, unless the new terrain database covers the same geographic area as the old one and is the same type of terrain (for example, UTM or geocentric), the scenario objects and overlay information associated with the scenario may be confusing or useless. (For details about editing Scenario Properties, please see "Editing Scenario Properties," on page 7-26.)

12.3. The Plan File

VR-Forces saves all individual and global plans to a plan file (*.pln*).

Plan files are created interactively in the VR-Forces Plan window. The Plan window ensures that the plan is written using the correct syntax. We recommend that you not try to create plan files or edit plan files by hand.

12.4. The Object Map File

When you start VR-Forces with multiple back-ends, you can specify which back-ends simulate the various objects you create. The object map file records the back-ends on which objects are simulated. The next time you open the scenario, VR-Forces looks for the back-ends specified in the object map file. If any of the specified back-ends is not available, the front-end asks you if you want to remap objects to the back-ends that are present. You can run the simulation without remapping. However, the behavior of a simulation that is missing simulation objects is unpredictable.

Mappings between objects and back-ends are saved to the object map file. If you have a scenario that uses multiple back-ends and you are uncertain about the site:application values you need to use to run the scenario, you can view the object map file to refresh your memory. The following example shows two entities and a route mapped to two different back-ends:

```
(map-entry
  (address  1 3201)
  (name  "M1A2 1")
)
(map-entry
  (address  2 3400)
  (name  "T80 2")
)
(map-entry
  (address  1 3201)
  (name  "route_1")
)
```

The address parameter lists the site and application number used for the back-end on which the object was simulated.

If you want to run an existing scenario using back-ends with different site and application values, you could edit the object map file and change the address field to match the new site and application values. However, if you make any mistakes in this process, your scenario might not run correctly. As an alternative to editing the object map file by hand, you could use the load balancing option to remap objects to back ends. For details about load balancing, please see "Load Balancing Scenarios," on page 7-30.

12.5. Editing the Scenario Objects File

When you create a scenario, you populate it with simulation objects and tactical graphics. When you save or checkpoint a scenario, VR-Forces saves a record of each simulation object and tactical graphic, including its state information and geographic location, to the scenario objects file. The next time you load this scenario, VR-Forces creates the simulation objects using the settings in the object parameter database, but then overrides those settings with the information in the scenario objects file. Therefore, changes you make to the object parameter database might not get picked up by existing scenarios. If you want to change a basic value, you might need to edit the scenario objects file.

12.5.1. Saving Default Parameters to the Scenario Objects File

When a simulation object in a scenario is saved to the scenario objects file, parameters that are unchanged from the initial values in the object parameter database entry are written out in the form:

```
(parameter-name USE-DEFAULT)
```

The USE-DEFAULT keyword indicates that the next time this file is loaded (as part of a scenario) it should be initialized from data in the simulation object's OPD entry. The practical effect of the USE-DEFAULT keyword in scenario objects files is that if you modify the object parameter database, the next time you load a scenario those changes can affect the simulation objects in the scenario.

This lets you experiment with changes to parameters and view the effects of the changes to existing scenarios, without modifying the scenario objects file.

12.6. The Scenario Extras File

VR-Forces lets you configure hostility relationships. That is, for any force you can specify which forces it is hostile to. By default Friendly and Opposing forces are hostile to each other. If you change the hostility relationships, they are saved in the scenario extras file (*.xtr*).

This file also stores spawn patterns.

12.7. Temporary Scenario Directories

When you load a scenario, VR-Forces creates temporary directories in which it stores files that it needs to maintain and update the scenario. To prevent conflicts in the temporary directories when running multiple copies of VR-Forces from the same directory, VR-Forces creates unique temporary directories. The directories are created in the path specified by the TMP environment variable. If this variable is not specified on Linux, the */tmp* directory is used. The TMP variable must be specified on Windows. If VR-Forces cannot create a temporary directory, an error message is printed and the application exits. In the front-end, the temporary directory is created by *DtVrfRemoteController* and starts with the prefix *vrffe*. In the back-end, the temporary directory is created by *DtCgfDispatcher* and has the prefix *vrfbe*. In both cases, the temporary directory is removed when VR-Forces shuts down.



III. Simulation Objects

The chapters in this section explain how to add simulation objects to a scenario, how to select them, and how to display information about them.



13. Introduction to Simulation Objects

Simulation objects are the entities and units that VR-Forces simulates as part of a scenario.

Simulation Objects 13-2
How Simulation Objects are Identified 13-2
UUIDs 13-3
Simulation Object Names 13-3
Echelon IDs 13-4
Object IDs 13-4
Labels 13-4
How Simulation Objects are Organized 13-5
How Echelon IDs are Assigned 13-6
How Simulation Objects Communicate 13-6
Custom Communication Models 13-6
VR-Forces Uses a Component Architecture 13-7
Sensors 13-7
Controllers 13-8
Actuators 13-8
Simulation Object Behaviors and Tasks 13-8
Entity-Level Modeling and Aggregate-Level Modeling 13-9
Collision, Obstacle, and Feature Avoidance 13-11
Entity Movement and Soil Type 13-12

13.1. Simulation Objects

The players in a simulation are called simulation objects. There are two broad classes of simulation objects:

- Entities represent a discrete individual vehicle or person, such as a tank or a submarine.
- Units represent a hierarchical group of entities, units, or both, such as a platoon or a company.

The concepts described in this section are generally applicable to entity-level modeling and aggregate-level modeling. For details about the differences between aggregate-level modeling and entity-level modeling, please see "Entity-Level Modeling and Aggregate-Level Modeling," on page 13-9.

Simulation objects can act in and respond to their environment. They execute tasks and they have behaviors. Each simulation object is part of a force and may be a member of an organizational unit.

The VR-Forces application comes with a set of simulation object models that you can instantiate in a simulation. Application developers can create additional simulation objects with the VR-Forces toolkit. You can also create new simulation object types in the Simulation Object Editor. For information about creating new simulation objects, please see Section XII, *Creating and Editing Simulation Models*.

13.2. How Simulation Objects are Identified

VR-Forces identifies simulation objects by UUID (universal unique ID), name, echelon ID, object ID, and label. Table 13-1 summarizes the characteristics of each type of identifier.

Table 13-1: Simulation object identifiers

Identifier	Characteristics
UUID	◆ Unique. Assigned by VR-Forces. ◆ Persists from exercise to exercise.
Name	◆ Persists from exercise to exercise. ◆ Set by user. ◆ Limits on the length of the character string. ◆ Does not have to be unique.
Echelon ID	◆ Must be unique. ◆ Persists from exercise to exercise. ◆ Assigned by VR-Forces. Changes if an entity is organized into a unit.

Table 13-1: Simulation object identifiers

Identifier	Characteristics
Object ID	◆ Unique. ◆ Does not persist from exercise to exercise. ◆ No user control.
Label	◆ Does not have to be unique. ◆ Persists from exercise to exercise. ◆ Controlled by user. ◆ No limit on character length.

13.2.1. UUIDs

UUIDs are unique identifiers generated by VR-Forces to ensure that every simulation object can be uniquely identified and that the identification persists each time you run the scenario. In particular, the use of UUIDs supports importation of scenarios because while simulation objects in different scenarios might have the same names, they will not have the same UUIDs, so it is possible to import them and not change their names if you do not want to. (The default is to rename duplicate simulation objects.)

13.2.2. Simulation Object Names

A name is a text string that, by default, consists of the simulation object's short name, as defined in the object parameter database, and an integer. For example, if you create a new scenario and add two M1A2 tanks and two Apache helicopters, the default names are `M1A2 1`, `M1A2 2`, `AH-64A 1`, and `AH-64A 2`. When VR-Forces generates a name, it picks an integer that makes the name unique. However, if you import a scenario and do not specify that duplicate names be renamed, it is possible that more than one simulation object will have the same name.

You can edit a simulation object name to make it meaningful to you.

Simulation object names have the following restrictions on their length:

- Entities – 11 characters.
- Units – 31 characters.



If you are using a character set that requires multiple ASCII characters to represent a visible character, entity and unit names will need to be correspondingly shorter.

13.2.3. Echelon IDs

An echelon ID is a text string that identifies the simulation object's place in the force hierarchy. For example, if you create an `M1A2 1` in Blue force (or Force 1), the echelon ID would be `1 M1A2, 1 Force`. The syntax of echelon IDs and the details of simulation object organization are described in "How Simulation Objects are Organized," on page 13-5. You cannot edit a simulation object's echelon ID. The echelon ID changes if you change the force hierarchy by adding, removing, or editing simulation objects.

13.2.4. Object IDs

Each simulation object has a unique object ID (global object designator). The global object designator is equivalent to the HLA object name in HLA, and the DIS entity identifier in DIS. The object ID is used to identify the simulation object internally and for network communications.

13.2.5. Labels

A label is a text string that you can use to identify simulation objects without the restrictions of uniqueness or character length that affect the other identifiers.

You can also identify simulation objects using extended labels – additional text strings that you can give descriptive names to. For details, please see "Using Extended Labels," on page 16-17.



In the VR-Forces GUI, entity labels refer to informational graphics displayed next to simulation objects. The label described in this section is one of the symbol decorations displayed in Plan View observer mode as part of the entity label. For more information about the different types of labels and information displays, please see "Displaying Entity Labels," on page 16-9.

13.3. How Simulation Objects are Organized

All simulation objects simulated by VR-Forces exist in the context of a hierarchy. At the top-most level, simulation objects are grouped according to force ID (friendly, opposing, neutral, and so on). At lower levels, simulation objects are grouped according to military organization, in which a simulation object has a designation in the hierarchy, such as 2nd Platoon, B Company. At each echelon level in the organization, a simulation object has a single superior simulation object. Disaggregated units contain one or more subordinate entities, units, or both. The hierarchy's lowest level contains an individual entity.



In entity-level scenarios, units can be in an aggregated state or a disaggregated state. For details, please see "Units in Entity-Level Scenarios," on page 21-2.

Each simulation object has an echelon ID in the following format:

```
designator [category] [echelon], designator [category] [echelon], ...,
  force_number Force
```

where:

designator is a number or letter that differentiates the simulation object from other members of the echelon, such as 1 (M1A1), 2 (M1A2), and so on.

category is an optional descriptor for the simulation object, such as M1A2.

echelon is an optional descriptor for the organizational unit, such as "Platoon".

force_number is the force the simulation object belongs to, based on the force type enumerations. The force is the highest level in the hierarchy.

When you create a new simulation object, it exists only within the default force. Therefore its echelon ID would be similar to:

```
1 M1A2, 1 Force
```

If a simulation object becomes part of a unit, the unit information becomes part of the echelon ID, and the designator may change, for example:

```
2 M1A2, 2 Plt, 1 Force
```

13.3.1. How Echelon IDs are Assigned

The short name and echelon labels that make up an echelon ID, such as "M1A2", "Plt", and so on, are specified in the VR-Forces object parameter database and are associated with object-types. When you create a simulation object of a certain type, the short name and echelon name associated with that type are assigned to the simulation object. You cannot change the labels interactively. You can change the labels by editing the object parameter database.

Designators are assigned by VR-Forces. You have no direct control over them. When you create new simulation objects, they are assigned designators sequentially, starting with one (1). If you aggregate a group of entities, the unit leader is assigned the designator 1, and the other entities in the unit are assigned successive designator numbers.

13.4. How Simulation Objects Communicate

VR-Forces includes a default communication model called simple-radio-comm-model. Using this simple communication model, radio messages always reach their intended destination (as long as the destination is on a reachable network), and messages are all passed instantly. The simple radio communication model has one parameter, which determines how the radio networks are set up. There are three connection-modes available:

- `aggregate` mode specifies that each unit has its own radio network. Messages sent from a simulation object can reach any simulation object that is in the same organization, at the same level (the simulation object's peers in the organization), and can also reach the simulation object's direct superior. All other simulation objects are outside the network, and are unable to receive messages sent by the simulation object. This is the default mode.
- `force` mode specifies that there is a radio network for each force in VR-Forces. Messages sent from a simulation object can reach any other simulation object of the same force.
- `all` mode specifies that all simulation objects in VR-Forces are part of one large radio network. In this case, any simulation object can receive messages sent by any other simulation object.

The communication model is configured in *./data/simulationModelSets/<sms>/CommModelParams.mtl*. For details about configuring radios, please see "Configuring Radios," on page 79-4.

13.4.1. Custom Communication Models

You can create new communication models and install them in VR-Forces. This is useful if you want to define your own connectivity model, or write your own algorithms for determining if radio messages get delivered. For details, please see API documentation.

13.5. VR-Forces Uses a Component Architecture

VR-Forces simulation objects use a component architecture similar to that used in many robotics applications. There are three basic types of components supported: sensors, controllers, and actuators. Components communicate with each other through data ports and port groups, which may provide data as simple as a single number (such as a throttle value) or more complex data such as a list of simulation objects that have been detected. The port groups make it possible to multiplex and prioritize multiple data sources to a single data recipient. For example, the automotive actuator of ground vehicles can receive inputs from a collision-avoidance controller, move-to-point controller, and follow-route controller, but it does not want inputs from all of those controllers at once.

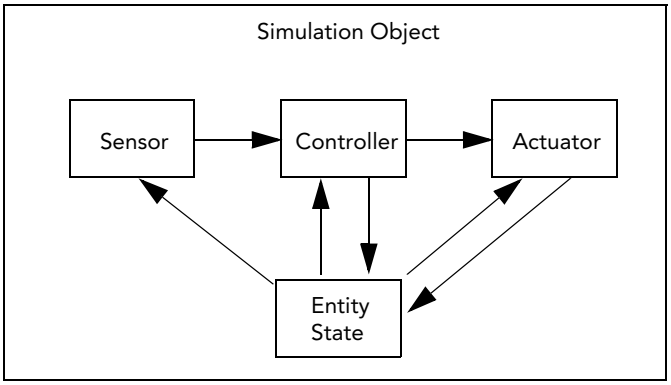


Figure 13-1. Simulation object component architecture

The component architecture supports any number of sensors, controllers, and actuators in a simulation object. Components are specified in the object parameter database by using component descriptors. Component descriptors indicate the type of component that should be created. They can also provide parameter information to a component.

For more information, please see Appendix C, *Object Parameters*.

13.5.1. Sensors

Sensor components provide models of the simulated environment, which are used by controller components to make decisions and perform tasks. The simplest sensor might provide (simulated) ground truth, while more sophisticated ones could use complex models for sensing electromagnetic emissions, RADAR, or a cognitive model to simulate a soldier or crew member's perception of the simulated world. Sensor components can get information from:

- The virtual network.
- From a terrain database.
- By monitoring the state of the simulation object model.
- Many other potential sources.

The use of sensors is optional.

13.5.2. Controllers

Controller components typically provide control inputs to actuator components (although this is not required). Controllers can use information provided by sensors to perform their functions, but this is not a requirement either. For example, the controller that implements the various forms of the Wait task neither requires sensor information, nor provides control inputs to any actuators.

13.5.3. Actuators

Actuator components:

- Provide the physical model of the simulation object being simulated.
- Change the simulation object's state.
- Interact with other simulation objects (such as by sending messages).

Actuators can use control inputs provided by controller components as parameters to its model each simulation frame. Actuators may:

- Send messages on the radio system.
- Generate events on the virtual network, such as detonations.
- Modify the state of the simulation object: position, velocity, damage, and so on.

Actuators can also use data directly from a sensor, when modeled or filtered data is preferred over simulated ground truth for a model that does not require a controller.

13.6. Simulation Object Behaviors and Tasks

A behavior is something that a simulation object knows how to do on its own. Behaviors describe how a simulation object reacts to a specific condition or situation or how it performs a task. A task is something that you tell a simulation object to do. A simulation object does not innately know that it should move to a point. However, if you task it to move to a point, it knows how to locate the point and how to move to it.

Behaviors can override task assignments. For example, if an entity is moving along a route and detects another entity in its path, the collision-avoidance behavior overrides the movement task until the entity successfully avoids the obstacle and can return to moving along the route.

VR-Forces has behavioral models for the included simulation object types. These models simulate the behaviors you would expect from live participants and objects functioning in the real world. You can change the parameters for the behavior models. With the VR-Forces toolkit, you can change the models themselves and add additional models. For more information, please see *VR-Forces Developers Guide*.



- No semi-automated forces (SAF) system, such as VR-Forces, can completely emulate the behavior of a complex real world entity. The behaviors provided by VR-Forces handle the lower level activities necessary to complete a task, and allow you to focus on the higher level activities involved in assigning tasks and preparing plans.
- If you do not want a simulation object to engage in any independent intelligent behaviors, you can disable them with the AI Enabled set data request. For details, please see "AI Enabled," on page 34-5.

13.7. Entity-Level Modeling and Aggregate-Level Modeling

VR-Forces supports two different approaches to modeling simulation objects – entity-level simulation and aggregate-level simulation.

In entity-level simulations, VR-Forces models the interaction among individual entities, such as ships, tanks, automobiles, airplanes, and people. When you create a simulation object, you are usually creating one tank, one ship, or one human, and so on. Combat takes place between entities based on their weapon systems and their ability to withstand munitions.

In an entity-level simulation, you can combine entities into higher echelon units, such as platoons and companies, but when they engage in combat, they do so by modeling the interactions of their members at the individual entity level.

Aggregate-level simulations focus on the high level movement and interaction of large numbers of personnel and equipment in hierarchical units. For the most part, they do not model individual simulation object interactions and combat. When you create a simulation object, you are usually creating a company, or a brigade, and so on. Aggregate-level combat is modeled based on the relative strengths and weaknesses of simulation objects in personnel, equipment, and location, not on the engagement of individual entities.

As in entity-level simulations, in an aggregate-level simulation, you can combine entities and units into higher echelon units.

The type of modeling used in a scenario depends on the systems that simulation objects use, which are configured in the simulation model set it uses. If you use *EntityLevel.sms* or an SMS similar to it, it uses entity-level modeling. If you use *AggregateLevel.sms*, or an SMS similar to it, it uses aggregate-level modeling. Therefore, when you create a scenario, you must consider which type of modeling you want to use and you must remember which type of modeling you are using when you populate the scenario.

Aggregate-level modeling only works with HLA Evolved and MAK FOM extensions. (If you try to create a scenario with *AggregateLevel.sms* and you are using an unsupported protocol, VR-Forces displays an error message.) For a detailed discussion of the aggregate warfare model, please see Chapter 22, *How Aggregate-Level Modeling Works*.



AggregateLevel.sms has some simulation objects that represent individual entities. However, they use data-driven models that are different from those used by similar entities in *EntityLevel.sms*.

Although it is possible to create a scenario that uses both types of simulation model sets, the resulting behavior will be unpredictable, because simulation objects that use aggregate-level modeling cannot interact with simulation objects that use entity-level modeling. Combining these SMSs is not recommended and not supported.

13.8. Collision, Obstacle, and Feature Avoidance

By default, in entity-level scenarios, simulation objects avoid other simulation objects, features, and obstacle control objects. If, while carrying out a task, a ground, lifeform, or rotary-wing entity detects that it is on course to collide with another entity, a feature, or an obstacle, it alters its route to avoid the collision. Then it returns to its task.



- In entity-level scenarios, aggregated units do not avoid obstacles and collisions. When disaggregated units move, the individual members of the units use collision avoidance.
- In aggregate-level scenarios, there is no collision avoidance between simulation objects. If a unit encounters an obstacle, it stops, sends a message to the console, and awaits further instructions.

When an entity avoids an obstacle, it does not necessarily follow the outline of the obstacle, it avoids the obstacle's bounding volume, which is conceptually the smallest rectangle that bounds all of the points in the obstacle.

Ground entities avoid features, such as buildings and plants. By default, entities avoid other entities (some cultural features are created as entities), buildings, water, and vegetation. (LCACs do not avoid vegetation by default.)

Collision avoidance has the following qualifications:

- To specify that ground entities should not move through the terrain (for example, through a wall or through a hill), set the check-terrain-collisions parameter in the automotive actuator component (powertrain) to True. For performance reasons, it defaults to False, so that entities that do not care about terrain collisions do not perform the check.
- If an entity is tasked to move to a point that is inside an obstruction, or very close to one, it might never reach the point, but it will continue to try to reach the point as long as the task is in effect. This behavior also applies to entities following routes that have vertices inside an obstruction.
- If a terrain has geometric features as well as point vector features, the feature avoidance may not work because the points are placed on the terrain on “top” of the existing geometry, which means they are on top of the geometric feature.
- Do not place entities inside obstacles. Their behavior will be unpredictable.
- Although you can assign a force to an obstacle when you create it, it has no effect on entity behavior. Entities avoid all obstacles.

You can configure the objects that an entity will avoid by editing its entry in the object parameter database. For details about configuring obstacle avoidance, please see “Configuring Obstruction Avoidance,” on page 78-4.

13.9. Entity Movement and Soil Type

In VR-Forces, the movement of simulation objects is affected by the surfaces on which they are moving. The speed of movement over a given surface (as a percentage of ordered speed) is configured in the soil-list block in a simulation object's object parameter database entry.

Table 13-2 lists the soil types supported for terrain databases and how they map to the roughness types that can configured in the soil-list block for VR-Forces simulation objects.

Table 13-2: Soil type/roughness type mappings

Soil type	Roughness type
ocean	deep-water
deepLake	deep-water
shallowLake	shallow-water
deepRiver	deep-water
shallowRiver	shallow-water
shallowPond	shallow-water
shallowStream	shallow-water
dryGround	hard-packed
pavedRoad	paved-road
gravelRoad	gravel
dirtRoad	gravel
asphalt	hard-packed
USRailroad	hard-packed
softSoil	sand
swamp	muck
mud	muck
forest	hard-packed
grass	hard-packed
cultivatedFields	hard-packed
orchards	hard-packed
rock	rocks
boulder	rocks
sand	sand
building	hard-packed
tree	hard-packed



14. Creating and Placing Objects

This chapter describes generic procedures for creating and placing simulation objects and tactical graphics. The later chapters assume that you are familiar with these procedures.

Creating Objects 14-3
Selecting the Object to Create 14-4
Selecting the Object to Create on an Object Palette 14-4
Selecting the Object to Create on the Create Menu 14-7
Filtering Object Creation Palettes 14-8
Draw Mode 14-11
Placing Objects 14-13
Placing an Object Using Default Values (Click to Create) 14-13
Specifying an Object's Properties Before You Create It (Click to Locate) 14-15
Default Placement of New Entities 14-17
Placing Entities Inside Buildings 14-17
Placing Simulation Objects on Other Simulation Objects 14-17
Specifying an Object's Altitude 14-18
Setting Altitude Dynamically 14-18
Setting Altitude in the Create Object or Edit Object Dialog Box. 14-18
Specifying the Altitude for All of the Vertices in a Route 14-19
Specifying an Object's Heading Dynamically 14-20
Locking the Mouse to the Object Being Created 14-21
Simulation Object Resources 14-22
Moving Objects 14-23
Dragging an Object to a New Location 14-23
Editing Simulation Objects 14-25
Editing an Object's Name in an Object List 14-26

Copying and Pasting Objects 14-28
Copying Objects 14-28
Pasting Objects 14-29
Pasting Specific Entity Characteristics 14-29
Deleting Simulation Objects 14-32
Simulation Object Groups 14-32
Creating Random Versions of Similar Entity Types 14-33
Creating Cultural Features 14-33
Embedding Cultural Features in the Terrain 14-34
Creating Props 14-35
Adding an Object to the Favorites List 14-36
Removing an Object from the Favorites List 14-36

14.1. Creating Objects

VR-Forces lets you create simulation objects, cultural features, tactical graphics, and props. You can create simulation objects and cultural features from the Create menu or the Simulation Objects Palette. You create tactical graphics from the Create menu or the Tactical Graphics palette. You create props from the Props Palette. The procedure for creating an object is essentially the same regardless of the type of object:

1. Select the object that you want to create.
2. Specify the location and properties of the object.

However, there are several ways to select the object you want to create and several ways to specify an object's location and properties. This chapter describes the different generic ways to create objects. Additional procedures specific to different object types are described in later chapters.

When you create an object, VR-Forces adds a tab to the right side of the window, below the object creation palettes. The tab is labeled Create *object*, where *object* is whatever object you selected to create. If you click that tab, it opens a dialog box that lets you specify the properties of the object you are creating. Table 14-1 lists the properties that you can set for each object type. You can also edit these same properties after you create an object.

Table 14-1: Editable object properties

	Entity	Cultural Feature	Tactical Graphic	Prop
Force	X	X	X	
Heading	X	X		
Label (and extended labels)	X	X	X	
Location	X	X	X	X
Name	X	X	X	
Overlay	X (Edit only)	X (Edit only)	X	
Style			X	
Publish Object			X	

14.2. Selecting the Object to Create

You can select the object to create on the Create menu or on an object palette (Simulation Objects Palette, Tactical Graphics Palette, or Props Palette). Once you have selected the object, placing the object is the same regardless of how you started the process. For details, please see "Placing Objects," on page 14-13.

The Create menu does not contain the full list of objects configured in VR-Forces. It only lists the simulation objects and tactical graphics that you have marked as "Favorites". It gives you quick access to your most frequently used objects. For details, please see "Adding an Object to the Favorites List," on page 14-36.

14.2.1. Selecting the Object to Create on an Object Palette

VR-Forces has the following object palettes that let you quickly choose the object that you want to create:

- Simulation Objects Palette. Create simulation objects, cultural features, and buildings.
- Tactical Graphics Palette. Create tactical graphics, such as control objects, lines, circles, points, and text. You can also create weather objects, volumetric lines and areas, contamination areas, and various types of hazards, obstacles, and pedestrian-oriented graphics. (Some of these graphics are aggregate-level only.)
- Props Palette. Create props that can be saved as part of the terrain. (The default window layout does not display the Props Palette. You can open it from the View menu.)

Figure 14-1 shows the location of the palettes (with the Props Palette enabled).

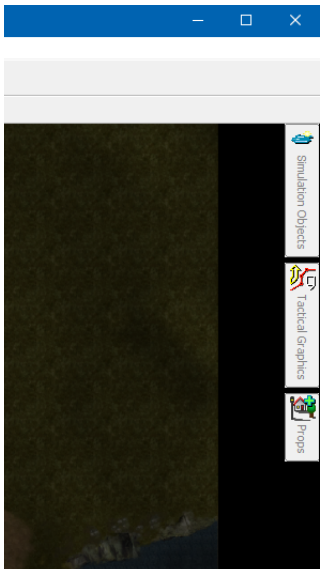


Figure 14-1. Object creation palette tabs

Figure 14-2 shows the Simulation Objects Palette and Tactical Graphics Palette when they are open.

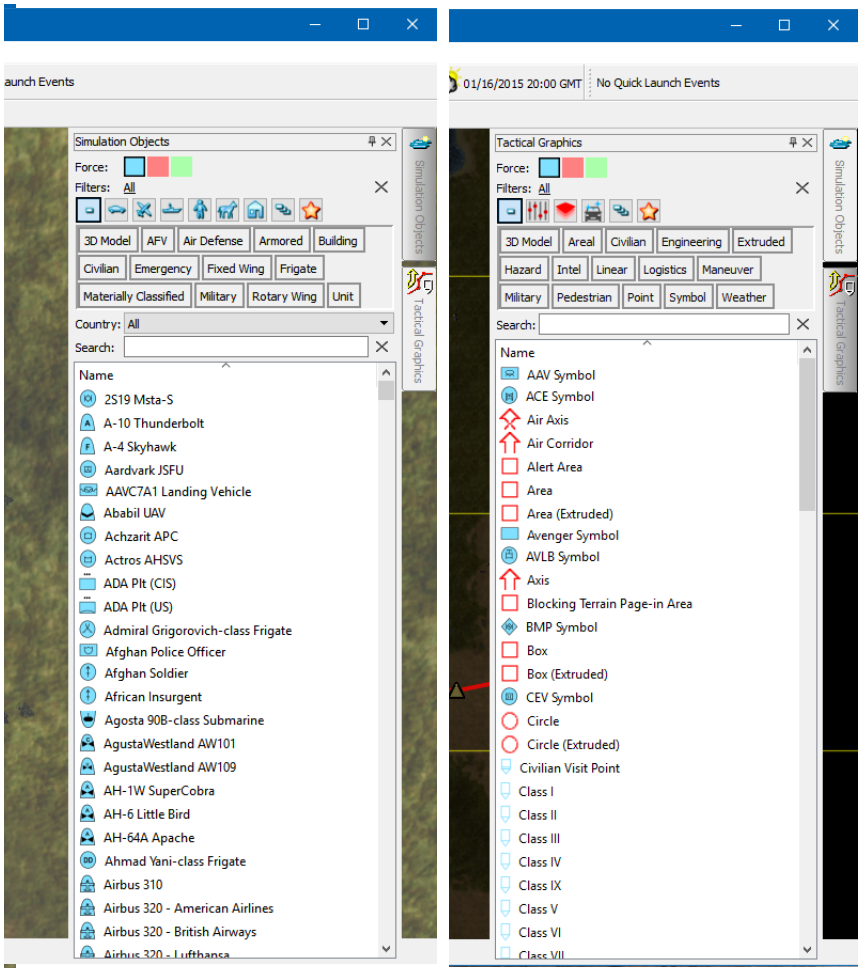


Figure 14-2. Simulation Objects and Tactical Graphics palettes

The objects in the Simulation Objects Palette and Tactical Graphics Palette are identified by their 2D icon and their name. You can use images other than the default icons to identify objects. For details, please see "Using an Alternative Icon List," on page 68-12.

To select the object to create on an object palette:

1. If you are using multiple simulation engines, in the Selected Simulation Engine Toolbar select the simulation engine on which you want to simulate the object.
2. Click the tab for the palette you want to open. The palette is displayed (Figure 14-2).

   The palettes list all objects that can be created. There is no preconceived force for any of them. You can filter the Simulation Objects Palette list by tag, country of use, and text string. You can filter the Tactical Graphics Palette by tag and text string. The Props Palette organizes props in a tree list.
3. On the Force line, click the button that represents the force for which you want to create this simulation object.
4. Optionally, filter the list of objects. (For information about how to filter object creation palettes, please see "Filtering Object Creation Palettes," on page 14-8.)
5. Optionally, filter the Simulation Objects Palette by selecting a country in the Country list.



The Country filter lists simulation objects of the selected tag that are used by the specified country. VR-Forces does not provide an exhaustive list of countries or the platforms that they use. You can add additional countries and categorize simulation objects in the Simulation Object Editor. For details, please see "Editing a Simulation Object's Country List," on page 68-9.

6. Optionally, type a search string to further filter the list.
7. In the list of objects, click the object that you want to create. The cursor changes to draw mode and a Create Object tab is added to the window. For details about draw mode, please see "Draw Mode," on page 14-11.
8. Create the object, as described in "Placing Objects," on page 14-13.

Pinning an Object Palette to the Window

Ordinarily, after you select an object to create, the palette closes. However, you can pin the palette to keep it open so that you can select additional objects to create without needing to click the palette to reopen it. If you pin a palette, it becomes dockable and undockable and you can drag it off the window like the other panels. (For details about undocking panels, please see "Dockable Panels and Toolbars," on page 83-2.)

➤ To pin a palette so that it stays open after you select an object, click the pin button (⊣) in the palette's title bar (Figure 14-3).

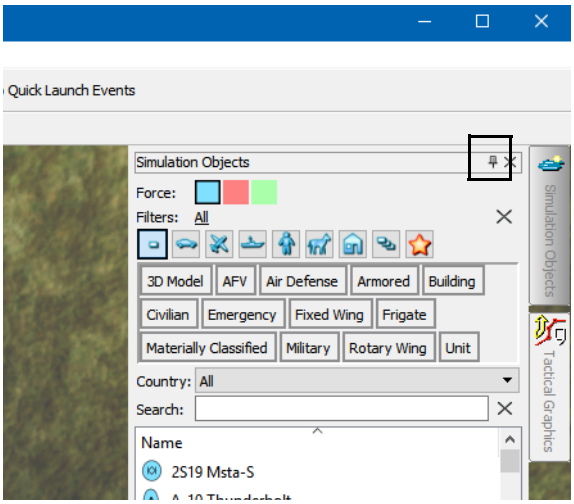


Figure 14-3. Palette pin

14.2.2. Selecting the Object to Create on the Create Menu

Waypoints, routes, areas, and objects marked as Favorites are available on the Create menu.

To select the object to create on the Create menu:

1. If you are using multiple simulation engines, in the Selected Simulation Engine Toolbar select the simulation engine on which you want to simulate the object.
2. To create an object or graphic, choose **Create** → *tag* → *object_type*, where:
   - *tag* is a category of entity, such as Ground or Human, a category of cultural feature, or a category of tactical graphic.
   - *object_type* is a specific type within the category, such as M1A2 or T80.

   The cursor changes to draw mode and a Create Object tab is added to the window. For details about draw mode, please see "Draw Mode," on page 14-11.
3. Create the object, as described in "Placing Objects," on page 14-13.

14.2.3. Filtering Object Creation Palettes

You can filter object creation palettes to make it easier to find the object that you want to create. You can filter simulation objects by tags, country of use, and text strings. You can filter tactical graphics by tags and text strings.

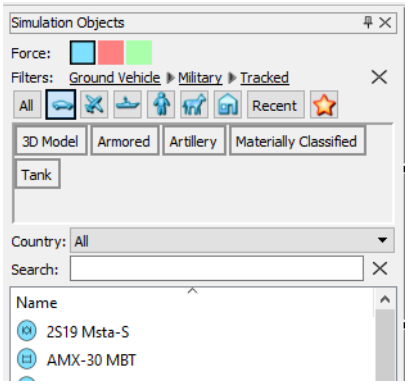


Figure 14-4. Object palette filtering

Tags are categories, such as land, air, military, civilian, and so on.

Many object types are used by multiple countries. If you want to see the objects used by a particular country, you can select the country of interest. (This filter does not apply to tactical graphics.)

The Search field lets you enter a text string to search for in object names. As you type, simulation objects that contain that string are listed. The Tag window is updated to show only the tags that apply to the listed objects.

Simulation objects get assigned to tags and countries in the Simulation Object Editor. For details, please see Chapter 67, *The Simulation Object Editor and SMSs* and Chapter 68, *Editing Common Parameters*.

To filter the objects listed in an object creation palette:

1. To filter by tags, select the tags you want to filter by. For details, please see "Filtering Object Creation Palettes Using Tags," on page 14-9.
2. To filter by country, select a country in the Country list.
3. To filter by text, type a text string in the Search box.

Filtering Object Creation Palettes Using Tags

All simulation objects are categorized using tags. An object can have multiple tags. For example, a ground vehicle can have the Ground Vehicle tag, but it can also have other tags such as Military, Tracked, Materially Classified, and so on. You can filter for any of these tags.

Tags are listed in the Filters section of an object creation palette. The top row of tags (represented by buttons) is static during any VR-Forces session (Figure 14-5). It contains the highest ranking tags. As you select tags, the Filters: line lists the tags you have selected from most inclusive to least inclusive. You can clear any level of filtering by clicking the label to its left.

The Tag window lists the tags that apply to simulation objects in the objects list that have not yet been filtered out. This list redisplays as you select additional tags. (If the number of tags that could be listed in the Tag window exceeds the space available, VR-Forces displays tags based on frequency of use and ranking. For details about ranking tags, please see "Adding a New Tag," on page 67-27.)

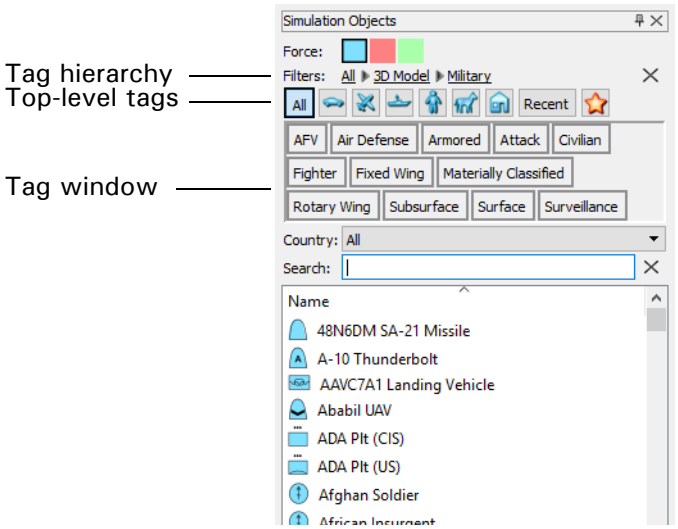


Figure 14-5. Simulation Objects Palette filters

VR-Forces remembers the tags selected for each top level tag. So if you set up some tags for Ground Vehicle and then filter using the Watercraft tag, if you go back to the Ground Vehicle tag, the tags you selected previously are still selected.

You can make tags "sticky". A sticky tag applies to all top level tags. For example, suppose you only want to see simulation objects that have 3D models, regardless of the platform. You could make 3D Models a sticky tag. Then no matter which top level tag you selected, it would be filtered for the 3D Models tag. Sticky tags are displayed in red text on the Filters line (Figure 14-6).

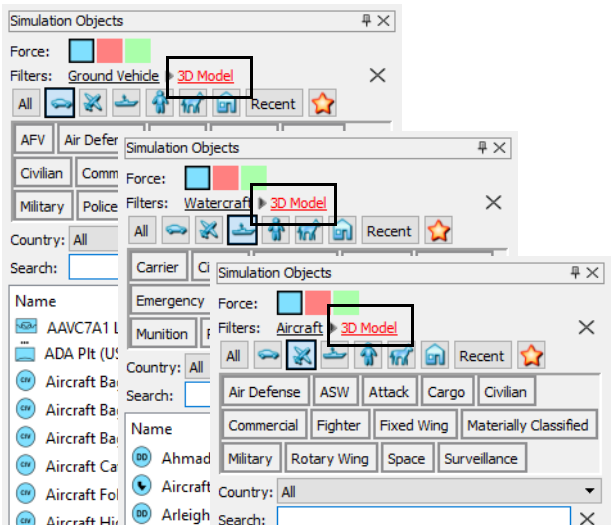


Figure 14-6. Sticky tags

To filter a list using tags:

1. Click a top level tag, or All.
2. Select additional tags in the Tag window to narrow the filtering. As you select tags, the object list redisplays and the tag list redisplays to show the tags that apply to the remaining members of the object list. As you select additional tags, the hierarchy of tags on the Filters line is updated.
3. If you always want to use a certain tag, such as 3D Model, or Materially Classified, right click the tag button. The tag is displayed in red text on the Filter: line. It applies to all top level tags.
4. To clear tags, in the breadcrumb path on the Filters line, click the tag to the left of the tag you want to clear. To clear all tags, click the X button.

14.2.4. Draw Mode

When you create an object, the cursor changes to draw mode. If you are creating a simulation object or a cultural feature in the 2D view, the cursor changes to a 2D icon that represents the chosen simulation object(Figure 14-7).

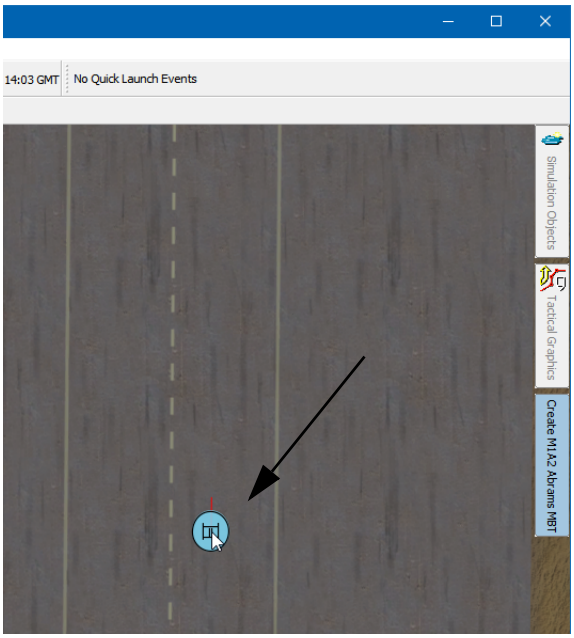


Figure 14-7. 2D entity creation and Create *object* tab

In the 3D view, the cursor changes to the 3D model for the chosen entity (Figure 14-8). If you are zoomed out, 3D models may be hard to see. If you are creating multiple human characters of a type that randomizes the appearance of the character that gets created, such as Civilian Male, after each character you create, the model attached to the cursor changes to show you the appearance of the next character that will be created.



Top-down view Ground-level view

Figure 14-8. 3D entity creation

If you are creating a tactical graphic, the cursor displays the waypoint symbol when you create a point and a vertex symbol when you create any type of line or area. Figure 14-9 illustrates tactical graphics draw mode cursors.

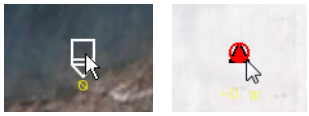


Figure 14-9. Tactical graphics cursors in draw mode

If you are creating a prop, the cursor changes to show the model for the prop. In the 2D view, the model is a top down view and is proportional to the terrain, so it may be difficult to see if you are zoomed out.

14.3. Placing Objects

After you select the object that you want to create, the Create Object tab is added to the right side of the window below the palettes.

There are two basic methods for placing simulation objects, as follows:

- Click to Create. Each mouse click creates an instance of the selected object. For multi-vertex tactical graphics, each click specifies a vertex. Click to Create is best for rapidly creating objects using the default values for the object's properties.

  The create object graphic is attached to the mouse cursor. Although you can use the Create Object dialog box to specify an object's name or other properties, as you move the cursor into the dialog box, the creation location (as represented by the object graphic) moves with it, which can be disconcerting. You can set heading and altitude dynamically, so there is no need to use the Create Object dialog box for these properties.

- Click to Locate. Mouse clicks specify the coordinates, but do not cause the simulation object to be created. The create object graphic is placed at the specified coordinates, but is not attached to the mouse cursor.

  If you want to explicitly set the object's location, and particularly if you want to set the altitude relative to sea level, you must use this method. This method may also be more comfortable for persons who want to take a deliberative approach to specifying an object's properties before creating it.

14.3.1. Placing an Object Using Default Values (Click to Create)

Although you can specify an object's properties using the Click to Create method, it is best suited for quickly creating objects using default properties. Objects are placed on the default overlay for the object type.



You can use this method to automatically embark simulation objects and attach tactical graphics. For details, please see "Embarking and Disembarking Simulation Objects Instantly," on page 17-4 and "Attaching a Tactical Graphic Automatically," on page 40-16.

To place an object with each mouse click:

1. Select the object that you want to create as described in "Selecting the Object to Create," on page 14-4. A Create object tab is added to the side of the window below the palettes.
2. For simulation objects and waypoints, click on the terrain for each object you want to place. You can create multiple single point objects without exiting create mode. For details about disabling this create mode, please see "Locking the Mouse to the Object Being Created," on page 14-21.

   For multi-vertex tactical graphics, click on the terrain to place each vertex. Right click to place the last vertex in a route or area. (Do not try to close areas by clicking on the starting point. Doing so only creates another vertex near the first one.)

   To draw a circles, ellipses, boxes, or text, please see the sections in Chapter 40, *Creating and Editing Tactical Graphics*.

- If new objects or vertices do not get displayed as you click, it is possible that Attach Object to Mouse is disabled. To restore Click to Create mode:
  1. Click the Create Object tab. The Create Object dialog box opens.
  2. Select the Attach Object to Mouse check box.
- If tactical graphics are not displayed after you create them, check to see if the display of tactical graphics has been disabled. For details, please see "Displaying Tactical Graphics," on page 38-6.

3. Optionally, you can set the altitude of an object as you create it. For details, please see "Setting Altitude Dynamically," on page 14-18.
4. Optionally, you can set the heading of an object as you create it. For details, please see "Specifying an Object's Heading Dynamically," on page 14-20.



If you want to specify any of the object's properties in the Create Object dialog box while using Click to Create, you can do so. However, be aware that as you move the cursor into the dialog box, the object graphic will move also. Once you specify the desired properties, click on the terrain to create an object with those properties. The properties will be applied to any additional objects you create until you change them or close the dialog box.

5. Right-click or press **ESC** to exit create mode.

14.3.2. Specifying an Object's Properties Before You Create It (Click to Locate)

In the Click to Locate method, clicking on the terrain sets the coordinates of the object, but does not create it. You can specify all of the object's properties in the Create Object dialog box before creating it. Table 14-1 lists the properties that you can set for the different types of objects.



VR-Forces remembers how you use the Create Object dialog box on a per-object basis. For example, if you create a waypoint and open the Create Waypoint dialog box to specify its properties, the next time you create a waypoint, the Create Waypoint dialog box opens automatically. If the dialog box opens and you do not want it open for this object creation or for future creations of the current object type, click the Create Object tab. The dialog box will close and VR-Forces will still be in object creation mode. The next time you create this object type, the dialog box will not open automatically.

To specify the properties of an object when you create it:

1. Select the object that you want to create as described in "Selecting the Object to Create," on page 14-4. A Create Object tab is added to the window below the palettes (Figure 14-7).
2. Click the Create Object tab. A Create Object dialog box that is appropriate for the object opens (Figure 14-12).

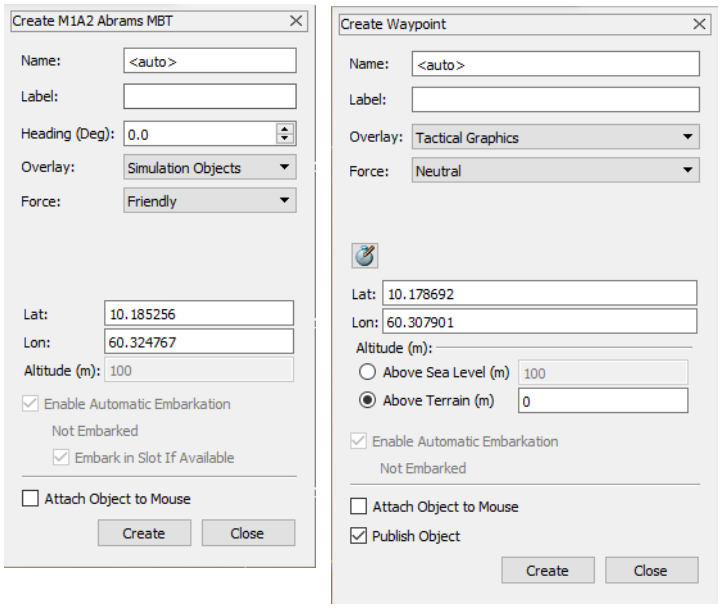


Figure 14-10. Create Object dialog boxes

3. Clear the Attach Object to Mouse check box. For details about this option, please see "Locking the Mouse to the Object Being Created," on page 14-21.
4. To specify the object's name, type a name in the Name box. For details about how simulation objects are named, please see "How Simulation Objects are Identified," on page 13-2.
5. Optionally, specify a label.
6. Optionally, if extended labels are available, specify them. For information about extended labels, please see "Using Extended Labels," on page 16-17.
7. To specify the object's heading, type or select a value, in degrees, in the Heading box. (Simulation objects only.)
8. To specify the overlay on which to place an object, select an overlay from the Overlay list.
9. Specify a force.
10. To specify a simulation object or point object's coordinates, click on the terrain or type coordinate values in the dialog box. To specify the altitude, follow the procedure in "Specifying an Object's Altitude," on page 14-18.

    To specify the vertices of a route or area:

    a. Click on the terrain to place the first vertex, or type the coordinates in the dialog box.
    b. Click the Add button (+) to add a vertex.
    c. Click to locate the vertex or type the coordinates.
    d. Continue adding vertices until you are finished specifying the route or area. You can insert vertices anyplace in the object by selecting an existing vertex and clicking the Add button.

    To draw circles, ellipses, boxes, or text, please see Chapter 40, *Creating and Editing Tactical Graphics*.
11. For tactical graphics, specify whether or not to publish the object. For information about publishing tactical graphics, please see "Publishing Tactical Graphics," on page 40-8.
12. Click Create. The object is created. You can continue to create additional objects of this type.
13. To close the dialog box, click Close.

14.3.3. Default Placement of New Entities

By default, ground, lifeform, rotary-wing, and fixed-wing entities get placed on the ground. Surface and subsurface entities are created at sea level.

Ground-based entities are placed at the highest possible terrain intersection at the location. If a terrain supports buildings with multiple levels, the entity is placed at the highest level. Once the entity begins moving, if the next calculated location resides in-between two levels of terrain, and if the entity is configured to respect terrain intersections when moving, and there is no terrain blocking its movement, the entity places itself at the closest elevation to the desired location.

14.3.4. Placing Entities Inside Buildings

One of the benefits of creating entities in a 3D observer mode is the ease of placing them in the precise location and orientation that you want them to be in, particularly inside buildings. However, placing entities inside small enclosed spaces can be hindered by the location of walls relative to the observer. You can work around this problem by:

- Adjusting the near clipping plane to remove the display of walls and floors until you see the location where you want to place entities. For details about changing the clipping plane, please see "Setting the Clipping Planes," on page 82-9.
- Adjusting the transparency of a building so that you can see inside of it. For details about changing transparency, please see "Setting the Opacity of Props," on page 57-25.



If you enable view constraints, it may restrict your ability to move the observer into buildings. You can disable view constraints by pressing **x**.

14.3.5. Placing Simulation Objects on Other Simulation Objects

If you want to place a simulation object on or in another simulation object, such as an aircraft on an aircraft carrier or a dismounted infantry entity in a truck, then you must embark the simulation object. Creating a simulation object on top of another simulation object will not provide the results you want. For example, if you create a fixed-wing entity and manipulate its location and altitude so that it looks like it is on the deck of an aircraft carrier, as soon as you start the simulation, the jet will start flying at that altitude, because it is not attached to the carrier. It is just created at a particular location. For complete details about embarking simulation objects, please see "Embarking and Disembarking Simulation Objects," on page 17-2.

14.4. Specifying an Object's Altitude

You can specify an altitude for air platforms, sub-surface entities, the vertices of waypoints and routes, and props when you create them or edit their properties. (You can also set altitude for simulation objects using the Location set data request, Fly Altitude tasks, and the Move to Location task. You can set altitude dynamically or in the various dialog boxes for setting object properties or actions.)

14.4.1. Setting Altitude Dynamically

You can set altitude dynamically for simulation objects that support an altitude, for waypoints, for individual vertices of a route, and for props that have not been saved into a terrain. When you set altitude dynamically, it is set relative to the terrain.

To set altitude dynamically:

- If you are creating an object, press and hold **ALT** and scroll the mouse wheel, or press and hold **ALT**+left mouse button and drag the mouse.
- If you are editing a simulation object or vertex:
  a. Double-click the object to make it editable.
  b. Press and hold **ALT** and scroll the mouse wheel, or press and hold **ALT**+left mouse button and drag the mouse.
  c. Click to set the new altitude.

14.4.2. Setting Altitude in the Create Object or Edit Object Dialog Box

When you set altitude in a dialog box, you have the following options:

- Above Sea Level. The distance above sea level. If the terrain is higher than the altitude entered, the entity will be below the surface of the ground.
- Above Terrain. The distance above the terrain at this location.

If you are creating or editing a route, you can adjust the altitude of all of the vertices at the same time and in the same way. For details, please see "Specifying the Altitude for All of the Vertices in a Route," on page 14-19.

To set altitude in a dialog box:

1. If you are creating or editing an object:
   a. Click the Create Object (or Edit Object) tab. A dialog box opens.
   b. Clear the Attach Object to Mouse check box. The Location group box becomes enabled. (Although the entity will change its location as you move the cursor into the dialog box, it snaps back to its original location as soon as you clear the check box.)
2. In the Altitude group box, select one of the options.
3. Type an altitude in the box.

14.4.3. Specifying the Altitude for All of the Vertices in a Route

When you create a route or edit its vertices, you may want to change the altitude of all of its vertices in a similar way. VR-Forces lets you:

- Adjust all vertices by the same amount.
- Set all vertices to the same height above sea level.
- Set all vertices to the same height above the terrain.

To change the altitude for all of the vertices in a route:

1. If you are creating a route, open the Create Route dialog box. If you are editing a route:
   a. Select the route.
   b. Choose **Objects** → **Edit**. The Edit Route dialog box opens.
2. Click the Adjust Points button ( ). The Adjust All Points dialog box opens (Figure 14-11).

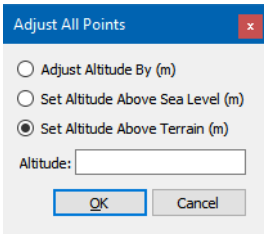


Figure 14-11. Adjust All Points dialog box

3. Select the adjustment action as follows:
   – Adjust Altitude By. Changes the altitude of each vertex by the amount specified in the Altitude box.
   – Set Altitude Above Sea Level. Sets the altitude of each vertex to the amount above sea level specified in the Altitude box. This could result in some vertices being below ground.
   – Set Altitude Above Terrain. Sets the altitude of each vertex the amount above the terrain specified in the Altitude box.
4. Type the desired altitude adjustment value in the Altitude box.
5. Click OK.

14.5. Specifying an Object's Heading Dynamically

When you create a simulation object, cultural feature, or prop, you can set its heading in the Create Object dialog box, or dynamically. Afterwards, you can edit a simulation object or cultural feature's heading directly or use the Heading set data request. (For more information, please see "Heading," on page 34-20.)



In the 2D view, as you change the heading for a simulation object you can see the heading indicator change heading. However cultural features do not have heading indicators, so if you want to set the heading dynamically, you may want to switch to 3D view so that you can see the model rotate.

➤ To set an object's heading dynamically as part of the creation process, hold down the **SHIFT** key and the left mouse button, and drag the mouse.

➤ To edit the heading of an object that has already been created, double-click it to put it into edit mode. Then hold down the **SHIFT** key and the left mouse button, and drag the mouse.

14.6. Locking the Mouse to the Object Being Created

By default, when you create simulation objects or waypoints, you can create multiple objects by continuously left-clicking without exiting create mode. This is a very convenient feature if you want to create objects using default naming conventions and do not need to specify specific location coordinates or overlay assignments during the creation process. (You can always edit an object's properties after you create it.)

However, if you want to use the Create Object dialog box to specify location or other properties for each object as you create it, this feature can be inconvenient because as you move the mouse to the Create Object dialog box, the placement location changes with the mouse movement. Disabling this feature can give you greater control over the creation process, because clicking to set the location of the object or a vertex does not create it – it sets the coordinates, but lets you make further changes in the Create Object dialog box before you create the object.

If you disable an object's attachment to the mouse when you are creating objects, you create them by clicking Create in the dialog box rather than clicking on the terrain.



If you have cleared the Attach Object to Mouse check box in the Create object dialog box, you must still specify the minimum number of points for the object before the Create button is enabled. For simulation objects this is just the location of the object, but for tactical graphics such as lines, circles, and areas, you may need to specify two, three, or more points. Since the object is not attached to the mouse, you will not see dynamic resizing of the object as you specify the points.

➤ To enable or disable the mouse's attachment to the creation location, in the Create Object or Edit Object dialog box, select or clear the Attach Object to Mouse check box (Figure 14-12). When you clear the check box, a Create button gets added to the dialog box. The setting gets carried over to future object creations.

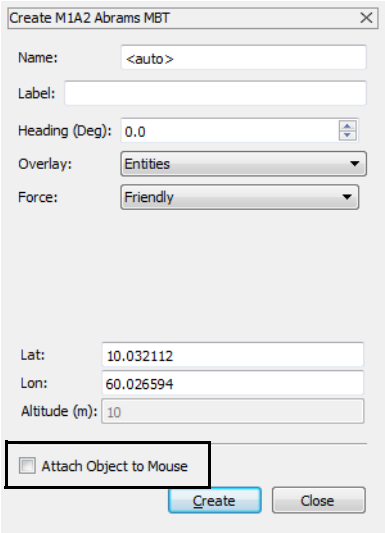


Figure 14-12. Create Object dialog box

14.7. Simulation Object Resources

When you create a simulation object, it has a full complement of resources, as defined in the object parameter database. You do not have to assign resources to a simulation object when you create it. However, if you want a simulation object to start a simulation with less than a full set of resources, you can use the Resources set data request to set its resources at some amount less than 100%. For details about the Resources set data request, please see"Resources," on page 34-34.

14.8. Moving Objects

You can move an object by dragging it, changing its location in the Edit dialog box, or for simulation objects only, sending a Location set data request.

For information about setting the location of a simulation object, please see "Location," on page 34-26.

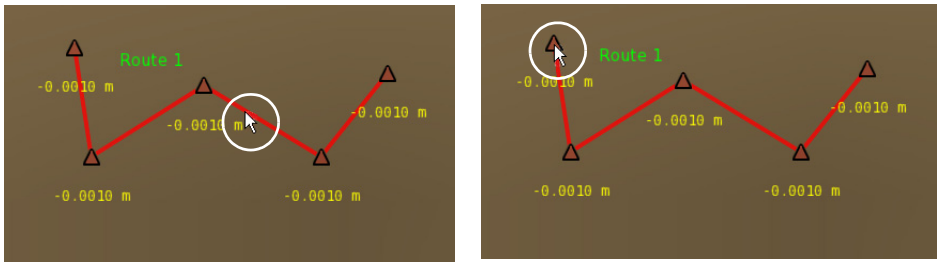


You can only move objects simulated by VR-Forces.

14.8.1. Dragging an Object to a New Location

When you move objects, they are centered on the mouse location as follows:

- If you select an object by double-clicking, it attaches to the mouse at the location where you clicked it. (For simulation objects and point objects, the objects are centered on the mouse location, but for lines or areal objects, the attachment is at the point where you clicked.)
- If you select an individual object and move it by choosing the Move command:
  - Point objects are centered on the mouse, just as when you double-click them.
  - Routes attach to the mouse cursor at the first point of the route (Figure 14-13).
  - Areas attach to the cursor at the center of mass.



Double-click drag Move command drag

Figure 14-13. Mouse attachment when moving individual lines

- If you select multiple objects and move them by choosing the Move command:
  - Point objects attach to the cursor at their center of mass.
  - Multiple objects have their centers of mass averaged and the center point attaches to the cursor (Figure 14-14).

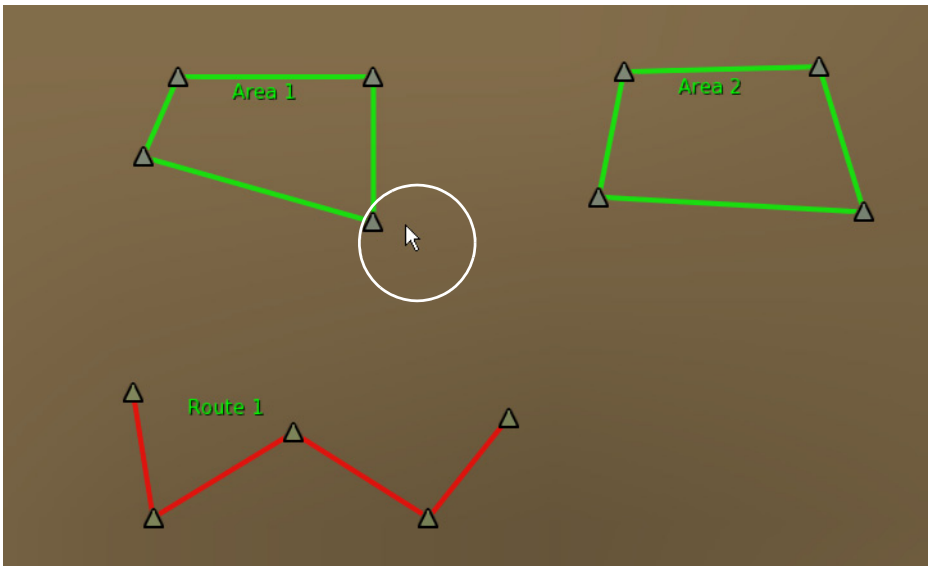


Figure 14-14. Moving multiple objects

If you click the terrain and then **SHIFT**+click or **CTRL**+click an object, the terrain is selected and the Move command will not be available.

To drag an object from one point on the terrain to another:

1. Double-click the object, or select the object and choose **Objects** → **Move**.
2. Move the mouse to change the location of the object.
3. Left-click the mouse.

- You can drag multiple objects and they can be of different types.
- To drag multiple objects, you must select the objects and choose **Objects** → **Move**.

Canceling a Drag-and-Drop Move

➤ To cancel a move made by dragging an object, press ESC before you click the left mouse button.

14.9. Editing Simulation Objects

You can edit the following simulation object properties:

- Name.
- Label (and extended labels if available).
- Force.
- Location.
- Heading.
- Altitude.
- Overlay.

To edit a simulation object:

1. Select the simulation object.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens (Figure 14-15).

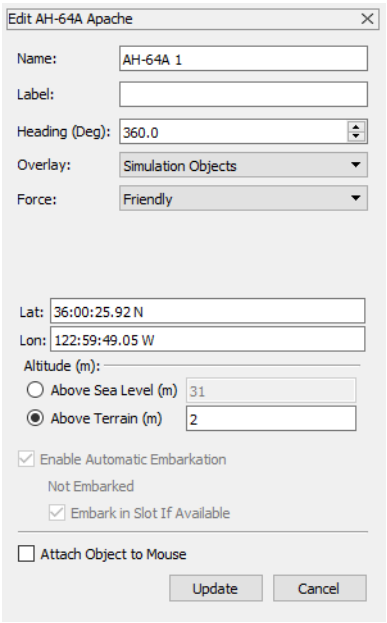


Figure 14-15. Edit Object dialog box

3. Change the properties as desired.
4. Click OK.

You can also change the name of an object by selecting it in the Objects List Panel and then clicking its name to make it editable.

You can also change the location, altitude, and heading dynamically, through a variety of movement tasks, or with set data requests. For details, please see:

- "Setting Altitude Dynamically," on page 14-18.
- "Specifying an Object's Heading Dynamically," on page 14-20.
- "Moving Objects," on page 14-23.
- Chapter 26, *Movement Tasks for Entity-Level Scenarios*.
- Chapter 30, *Tasks for Aggregate-Level Scenarios*.
- Chapter 34, *Setting the State of Entity-Level Simulation Objects*.

14.9.1. Editing an Object's Name in an Object List

You can edit an object's name in an Edit dialog box or directly in an object list. The ability to edit an object's name in an object list is disabled by default. This prevents you from inadvertently going into edit mode when you just want to select an object.

To enable and disable editing an object's name in an object list:

1. Choose Settings Application Settings. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 14-16).

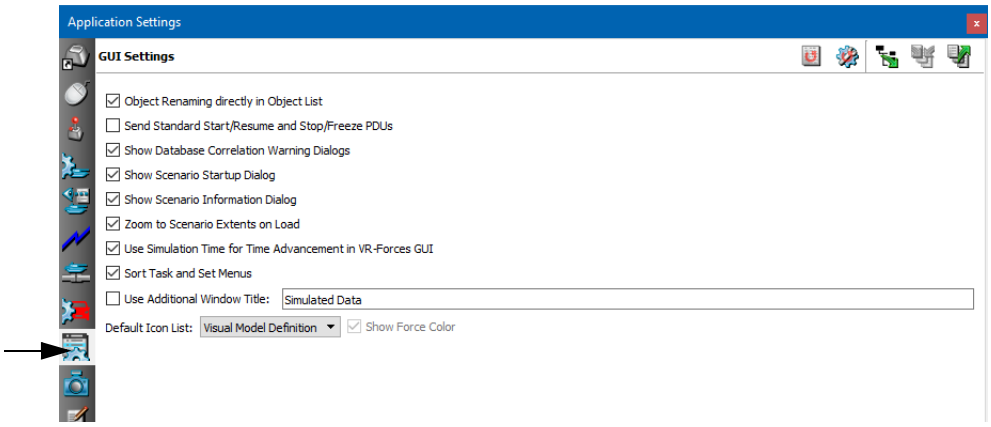


Figure 14-16. GUI Settings page

3. To allow editing of names in an object list, select the Object Renaming Directly in Objects List check box. To prevent editing of names, clear the check box.
4. Click Close

To edit an object's name in an object list:

1. Click the name of the object, then click again. This is not a double-click. The two clicks should be about 1 second apart. The name becomes editable.
2. Edit the name.
3. Click someplace else in the window.

14.10. Copying and Pasting Objects

You can copy objects and paste the copies elsewhere in a scenario or even in another scenario. The Paste Special option allows you to specify which entity characteristics you paste.

VR-Forces copies the following data about an object:

- Environmental objects:
  - Original name.
  - Label.
  - Force.
  - Color.
- Entities:
  - Original name.
  - Label.
  - Force.
  - Heading.
  - Appearance (lights, landing gear, and so on).
  - Rules of engagement.
  - Plan.
- Units copy the same information as entities plus the list of subordinates.

VR-Forces does not copy:

- The current task.
- State information, except as noted previously.
- Load balancing status. All objects get pasted into the currently selected back-end.
- Embarkation structures. If, for example, you copy an aircraft carrier that has planes embarked, only the carrier is copied, not the embarked entities.

! The copy/paste process is resource intensive. It is primarily for use during scenario development. Use during a simulation could affect performance.

14.10.1. Copying Objects

To copy objects:

1. Select the objects you want to copy.
2. Choose **Objects** → **Copy**.

14.10.2. Pasting Objects

A simple paste pastes all of the attributes of an object that get copied. It pastes objects as follows:

- Tactical graphics are placed on the currently selected overlay. If you copied objects that were on different overlays, that differentiation is not preserved.
- Copied objects maintain the state at the time that they were copied. Therefore, if you paste objects some time after they were initially copied, the copies do not reflect any changes that may have taken place to the original simulation objects. For example, if a simulation object was copied and then deleted, the copied version of the entity is still pasted.
- Embarkation structures are not copied, only the parent entity.

! Pasting plans may have unexpected consequences, because their object references do not change when you copy them. For example, suppose the plan for Entity A refers to Waypoint 1. You copy Entity A and Waypoint 1 and paste them as Entity B and Waypoint 2. If you copy the plan, Entity B's plan will still refer to Waypoint 1. If you expected it to refer to Waypoint 2, you will not see the behavior you expected. You can choose not to paste plans. For details, please see "Pasting Specific Entity Characteristics," on page 14-29.

To paste objects:

1. Choose **Objects** → **Paste**. The objects in the paste buffer are displayed on screen. They float as you move the cursor to show you where they will be placed when you paste them.
2. Click on the terrain where you want the objects to be pasted. They are pasted in the relative positions in which they were copied. If an object has an altitude, it is pasted using the same altitude above the terrain as the original object.
3. Optionally, continue pasting additional copies if desired.
4. Right-click to exit paste mode.

14.10.3. Pasting Specific Entity Characteristics

When you paste simulation objects, you can specify which of the following entity characteristics you want to paste with the entity:

- Plans.
- Appearance.
- Heading.
- Rules of engagement.
- Entity name.
- Altitude. You can specify that the previous altitude be applied as above terrain or as above sea level.

To paste particular entity characteristics:

1. Choose **Objects** → **Paste Special**. The Paste Special dialog box opens.

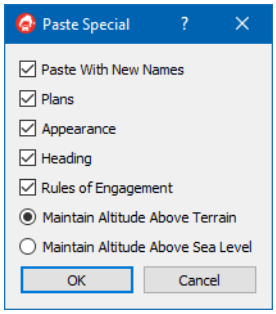


Figure 14-17. Paste Special dialog box

2. By default, all check boxes are selected. Clear the check boxes for the options you do not want to paste.
3. Choose the altitude option that you want to use.
4. Click OK. The objects in the paste buffer are displayed on screen. They float as you move the cursor to show you where they will be placed when you paste them.
5. Click on the terrain where you want the objects to be pasted. They are pasted in the relative positions in which they were copied.
6. Optionally, continue pasting additional copies if desired.
7. Right-click to exit paste mode.

Pasting Altitude

When you paste objects, if the pasted object can have an altitude value, such as a fixed-wing entity, a waypoint, or the vertices of a route, the default behavior is to replicate the original altitude above the terrain. If you use the Paste Special operation, you can choose to apply the original altitude as being above sea level. Figure 14-18 illustrates these alternatives. A waypoint is placed on the terrain. It is 0 meters above the terrain. However, the terrain itself is 200m above sea level. If the waypoint is copied and pasted at point A and the altitude above the terrain is maintained, the point is placed at 0m above the terrain. If the point is pasted at point B and altitude above sea level is maintained, it is placed 200m above sea level.

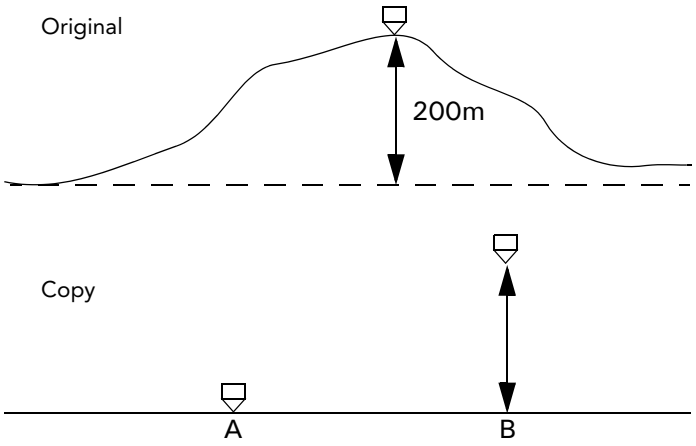


Figure 14-18. Pasting altitude

14.11. Deleting Simulation Objects

To delete a simulation object:

1. Select the object you want to delete.
2. Choose **Objects** → **Delete** or press the **`Delete`** key.



- If you delete a unit, all its subordinates also get deleted.
- If you delete a simulation object that has an intervisibility object pinned to it, the intervisibility object also gets deleted. For details, please see "Deleting Intervisibility Objects," on page 48-16.

14.12. Simulation Object Groups

Simulation object groups (SOG) are a convenient mechanism for creating reusable configurations of simulation objects and tactical graphics. You create them in the Simulation Object Editor and they are listed on the Simulation Objects Palette with other simulation objects. VR-Forces places them in the relative positions that you specified when you created the SOG. VR-Forces includes example SOGs, including a carrier group, a Patriot missile battery, and a barracks with several soldiers. These examples demonstrate SOGs with tactical graphics, cultural features, and embarkation.



The Barracks With Guards SOG has some tactical graphics, but they are hidden by default. To see them, click the overlay icon at the bottom of the window, or select the check box on the Objects List Panel, Overlays View.

The Simulation Objects Palette has a tag for SOGs (Object Group), but like other simulation objects, they can have multiple tags. SOGs can be distinguished from other simulation objects in the lists because the object name includes the number of objects in the group.

For information about creating SOGs, please see "Creating a Simulation Object Group," on page 68-28.

14.13. Creating Random Versions of Similar Entity Types

VR-Forces has many different models for generic types of entities, such as civilian vehicles or commercial aircraft. When you are creating a scenario with these sorts of entities, you can choose the specific entity types you want to use. However, if you want to quickly populate a scenario with a variety of similar objects, it is simpler to just let VR-Forces randomly create instances of the generic entity. This is done using the random simulation object.

A random simulation object is configured in the Simulation Object Editor with a list of specific simulation object types that it can draw from. When you create a scenario, you choose the random entity type on the Simulation Objects Palette and when you click on the terrain to place the object, VR-Forces picks a specific simulation object from the list.

VR-Forces includes a random simulation object for civilian vehicles.

You can add your own random simulation objects in the Simulation Object Editor.

14.14. Creating Cultural Features

Cultural features are objects that you can add to a scenario to simulate human artifacts such as buildings, bridges, and so on. They have the Cultural Feature tag on the Simulation Objects Palette. You add them to a scenario the same way you would add an entity or tactical graphic. Cultural features are listed in the Objects List View of the Objects List Panel. You can apply some set data requests to them, such as Location and Heading.

Some of the choices on the cultural feature lists are the same as those available on the Props Palette. However, cultural features are not the same things as props. Props are terrain objects and can be saved as part of a terrain. Cultural features are not part of the terrain, they are a type of simulation object and are saved in the scenario objects file.

14.14.1. Embedding Cultural Features in the Terrain

If you have a cultural feature, such as a house, that has object geometry, VR-Forces can treat that object as part of the terrain when you place it in a scenario. This means that entities can walk on it or in it (if it has interior geometry).

If you place the object outside of a navigation area, entities interact with it similarly to how they would interact with a prop. If you place the object in a navigation area, VR-Forces dynamically regenerates navigation data for the sectors in which the cultural feature is located. Entities can then use the navigation data to walk around or in the cultural feature more intelligently than if it was outside of a navigation area.

You do not need a license for generating navigation areas to dynamically regenerate navigation data. However, you must have a terrain on which navigation areas have been created, such as the terrains provided with VR-Forces. For information about navigation areas and navigation data, please see Chapter 61, *Generating Navigation Data*.

This feature has the following requirements and constraints:

- The cultural feature must be configured in the Simulation Object Editor to use object geometry in the simulation engine and to be embedded in the terrain. For details, please see "Configuring Cultural Features To Be Embedded in the Terrain," on page 69-9.
- If you place the cultural feature in a navigation area, the area must be sectorized. If it is not sectorized, the navigation data will not be generated.
- Navigation data gets generated dynamically. It is not saved with the terrain or with the scenario. When you reload the scenario, the navigation data gets generated after the scenario is loaded. If you have many such cultural features in a scenario, this can take some time. During that time, you might not be able to use VR-Forces or other applications on your computer.
- Creating a navigation area and generating navigation data for a portion of a terrain in which you have placed a cultural feature does not generate navigation data for the cultural feature. The navigation data is generated based on the content of the terrain and the cultural feature is a simulation object, not part of the terrain. If you want the geometry of a cultural feature to be part of the terrain, add it as a prop.

➤ To embed a cultural feature in the terrain, create it in a scenario.

14.15. Creating Props

Props are terrain objects that you can select independently of the terrain. "Extracting Props from a Terrain," on page 66-8 explains how to create them by extracting them from a terrain or a feature layer. However, you can also create them from the Props Palette, similarly to how you create simulation objects. This is a valuable feature if your terrains do not have any features that you can extract as props. However note the following important details:

- Props are not saved in a scenario. If you want to save a prop, you must save the terrain. You can save the terrain to a different name or overwrite the existing terrain. The next time you open the terrain, the prop will be part of it.
- When you create a prop, the back-end does not know anything about it, because it is a terrain object. To have the back-end take the prop into account, you must save the terrain (as mentioned in the previous paragraph) and restart the scenario.
- Props get created on the front-end terrain. If the front-end and back-end are using different terrains, the back-end terrain will not include the props.
- You can export props to a shapefile. This lets you add them to a terrain using an earth file. For details, please see "Exporting Props and SpeedTrees to a Shapefile," on page 57-27.
- You can also create cultural features that look exactly like the props that you can create. However cultural features are not the same things as props. Cultural features are treated like simulation objects. They are listed with simulation objects and tactical graphics and get saved as part of the scenario.



The list of props that you can create from the Props Palette is based on the model definitions defined on the Model Definitions tab of the Visual Model Editors dialog box. Therefore, you can add new props to the Props Palette by adding new model definitions. For details, please see "Adding Object Models as Props," on page 90-13.

14.16. Adding an Object to the Favorites List

As mentioned in "Selecting the Object to Create," on page 14-4, the Create menu does not include all of the objects that you can create. It just lists the objects that you have designated as "Favorites". When you specify that a simulation object or tactical graphic be a favorite, it is automatically added to the Create menu. An option is added to the menu for each tag associated with the favorite. As you add additional favorites, they are listed under all of the tags that apply to them.

There are no visible indications on the object creation palettes that objects are favorites. However, you can filter the object list using the Favorites tag ().

The favorites list is specific to a simulation model set. It is saved to *./data/simulationModelSets/<model_set>/gui/favorites.mst*. You can copy the file to other instances of VR-Forces or future releases and your favorites list will be enabled.

To add or remove an object from the Favorites list:

1. Open the palette that has the object you want to add to the Favorites list.
2. Right-click the object.
3. On the pop-up menu, select **Add to Favorites**.

14.16.1. Removing an Object from the Favorites List

When you remove an object from the Favorites list, it also gets removed from the Create menu.

To remove an object from the Favorites list:

1. Open the palette that has the object you want to remove from the Favorites list.
2. Right-click the object.
3. On the pop-up menu, select **Remove from Favorites**.



15. Selecting Objects

This chapter describes ways to select simulation objects and tactical graphics.

The Objects List Panel 15-2
Selecting Simulation Objects, Tactical Graphics, and Props 15-4
Selecting Objects in the Window 15-5
Selecting Objects in the Objects List Panel 15-6
Selecting a Simulation Object in the Object Console Summary Panel .. 15-6
Unselecting Objects 15-7
Using Selection Groups 15-8
Creating a Selection Group 15-8
Selecting a Selection Group 15-9
Adding Objects to a Selection Group 15-10
Removing Objects from Selection Groups 15-10
Renaming a Selection Group 15-10
Deleting a Selection Group 15-10

15.1. The Objects List Panel

There is a great deal of similarity in how you select, move, and view the different types of objects. You can manage them on the terrain, to some extent from the keyboard, and in the Objects List Panel, which is the central repository of object information.

The Objects List Panel has the following views:

- Object View: displays simulation objects in the simulation in a hierarchical list by force. Also lists tactical graphics (Figure 15-1).
- Embarkation View: displays the embarkation relationships of simulation objects (Figure 15-1).
- Overlays View: lists overlays and the objects that are on them (Figure 15-2).
- Selection Groups View: lists selection groups and their members (Figure 15-2).

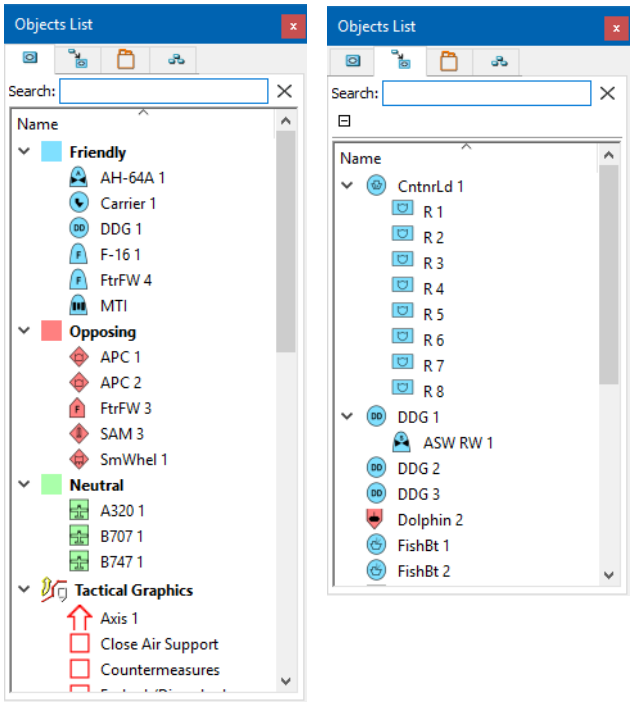


Figure 15-1. Objects List Panel – Objects List View and Embarkation View

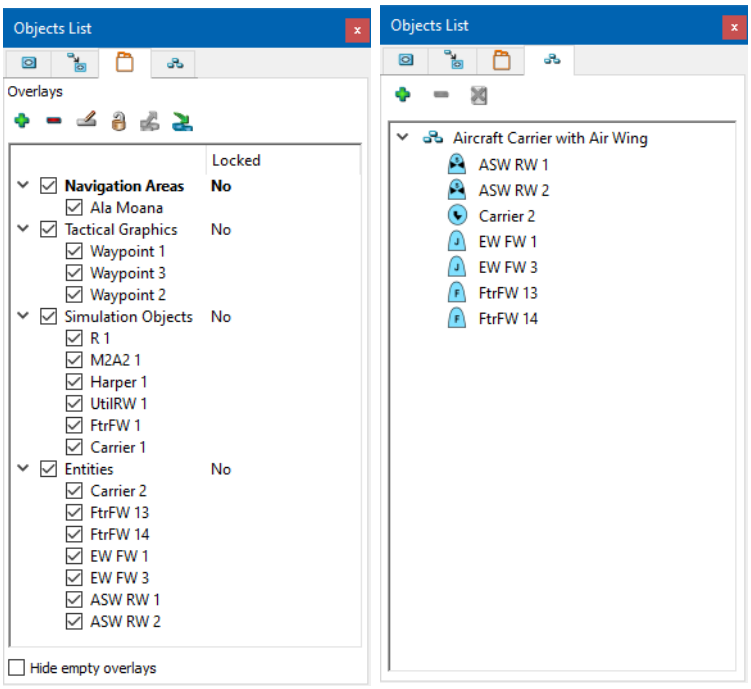


Figure 15-2. Objects List Panel – Overlays View and Selection Groups View

The objects in the objects list panel are identified by their 2D icon and their name. You can use images other than the default icons to identify objects. For details, please see "Using an Alternative Icon List," on page 68-12.

15.2. Selecting Simulation Objects, Tactical Graphics, and Props

VR-Forces displays the following kinds of objects that you might want to select:

- Simulation Objects. Any of the entity and unit types that you can create.
- Cultural features. Objects that have the Cultural Feature tag on the Simulation Objects Palette.
- Tactical graphics. Control objects and the points, lines, areas, text, and symbols on the Tactical Graphics Palette and Create menu.
- Props. Any of the props that you can create from the Props Palette or which are already part of the terrain.

You can select all types of objects in the window. You can select simulation objects, cultural features, and tactical graphics in the Objects List Panel (Figure 15-1). You can select simulation objects in the Object Console Summary Panel, and you can select props on the Props page of the Terrain Settings dialog box (Figure 15-3).

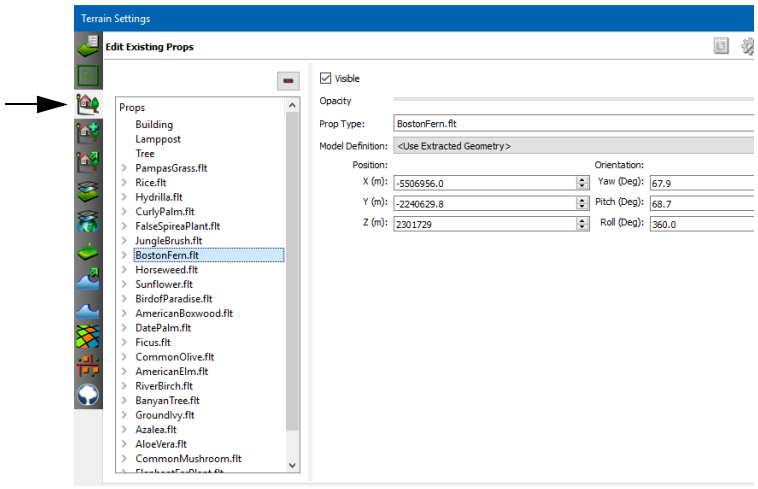
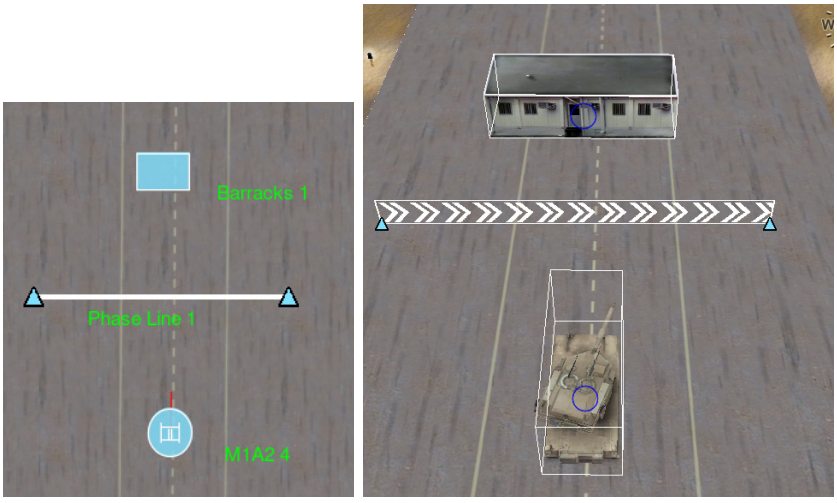


Figure 15-3. Terrain Settings dialog box, Props page

When you select a simulation object or tactical graphic in the 2D view, its border color changes to white. In the 3D views, all selected 3D objects display a bounding box (Figure 15-4). Tactical graphics change to white lines.



2D 3D

Figure 15-4. Selected objects

15.2.1. Selecting Objects in the Window

To select objects in the window, use any of the following mouse actions:

Mouse Action	Result
Click the object.	Selects an object (and unselects any other selected objects).
CTRL + click the object.	Selects an object while keeping existing selections.
Double-click the object.	Selects the object and makes it editable. This is the equivalent of selecting an object and choosing **Objects** → **Edit**.
ALT + drag a bounding box.	Selects multiple objects by dragging a selection box around them.
Right-click the object.	If no objects are selected, right-clicking selects an object and displays a menu. If one or more objects are selected, right-clicking a selected object maintains the selection and displays a menu. Right-clicking an unselected object changes the selection to the object you right-clicked.

When you select an object in the window, the views on the Objects List Panel add a gray highlight to the entry for the object to indicate that it is selected.

15.2.2. Selecting Objects in the Objects List Panel

To select an object in the Objects List Panel:

1. Select a view in the Objects List Panel (Figure 15-1).
2. Locate the object you want to select. You can filter the list to make it easier to find particular objects. (For details, please see "Filtering the Object Lists," on page 15-6.) Disaggregate and expand units if necessary. (For details, please see "Expanding and Collapsing Units," on page 24-2 and "Triggering Unit State Transitions," on page 23-11.)
3. Click the entry for the object. The selected object is highlighted in the Objects List Panel and on the terrain.
4. Optionally, select multiple objects, as follows:
   - **SHIFT**+click to select a contiguous range of simulation objects.
   - **CTRL**+click to select multiple non-contiguous simulation objects.

Filtering the Object Lists

You can filter the display in the Objects List.

➤ To filter the display of objects, type a string in the Search box at the top of the list (Figure 15-1).

15.2.3. Selecting a Simulation Object in the Object Console Summary Panel

The Object Console Summary Panel lists messages that have been sent to simulation objects from the simulation engine, from a simulation object's plan, from other simulation objects, and from scripts. (For information about the Object Console Summary Panel, please see "Configuring Object Console Messages," on page 16-23.)

To select a simulation object in the Object Console Summary Panel:

1. Right-click a message from the simulation object you want to select.
2. On the shortcut menu, choose Select Entity.

15.2.4. Unselecting Objects

To unselect objects in the window, choose one of the options in Table 15-1.

Table 15-1: Unselecting objects on the terrain

Action	Result
Click in the window away from the selected object or objects.	Unselects all objects.
Right-click.	Exits edit mode, but object remains selected.
Click a selected object.	If multiple objects were selected, unselects all except the object you clicked.
CTRL + click a selected object.	Unselects the object. If multiple objects were selected, does not unselect other objects.

To unselect objects on the Objects List Panel, choose one of the options in Table 15-2.

Table 15-2: Unselecting objects in the Objects List Panel

Action	Result
If one object is selected.	**CTRL** + click the object.
If a group of objects is selected:	
CTRL + click an object.	Unselects the object without affecting other selections.
Click an object.	Selects the clicked object and unselects all other objects.

15.3. Using Selection Groups

Selection groups are named subsets of the objects in an exercise. Their purpose is to allow you to quickly select specific objects without needing to locate them on the map or the Objects List View in the Objects List Panel and select them individually. Using selection groups can be helpful if you want to quickly select a group of objects that do not fit neatly into the default filter options on the Objects List Panel views, for example a mix of friendly and opposing forces and tactical graphics. The group definitions are saved and loaded as part of the scenario.

An object can be in multiple selection groups.

15.3.1. Creating a Selection Group

You can create selection groups in any of the following ways:

- In the Selection Groups View on the Objects List Panel.
- From Selection Groups menu options.
- Using keyboard shortcuts.



When you add a simulation object group to a scenario, VR-Forces automatically creates a selection group for all the objects in the simulation object group. For information about simulation object groups, please see "Simulation Object Groups," on page 14-32.

To create a selection group in the Selection Groups Panel:

1. On the Objects List Panel, select the Selections Groups View (Figure 15-5).

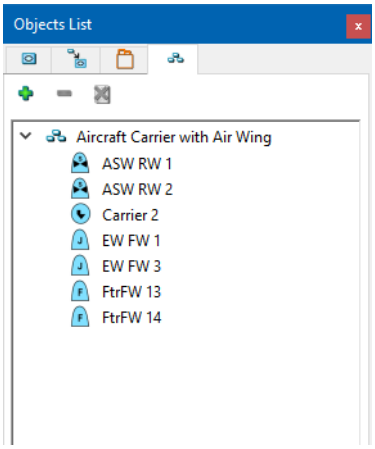


Figure 15-5. Selection Groups tab

2. Click the Add button (+). The New Selection Group dialog box opens.
3. Type a name for the selection group.

4. Click OK. The group is added to the group list.

Creating a Selection Group from a Selection Group Menu Command

When you create a selection group from a Selection Group menu command, you can combine the procedures for creating a new selection group and adding an object to it.

To create a selection group from a Selection Group menu command:

1. Select the objects you want to put in a new selection group.
2. Choose **Objects** → **Selection Groups** → **Add to New Selection Group**. The New Selection Group dialog box opens.
3. Type a name for the selection group.
4. Click OK. The selection group gets created and the selected objects get added to it.

Creating a Selection Group from a Keyboard Shortcut

You can quickly create a selection group with a default name using a keyboard shortcut. When you create a selection group from the keyboard it is created as follows:

- If no objects are selected, an empty selection group is created.
- If objects are selected, a selection group is created with these objects as members.
- If the keyboard shortcut is for a selection group that already exists, a new selection group replaces the previous one. If the previous group had members and the new one is created without any objects selected, the new group is empty.

➤ To create a selection group from the keyboard, press **CTRL** + *number*, where *number* is 1, 2, 3, 4, 5, 6, 7, 8, 9, or 0. A selection group is created named Selection Group *number* (0 creates Selection Group 10). (The keypad numbers do not support this operation.)

15.3.2. Selecting a Selection Group

When you select the objects in a selection group, they are selected in the views on the Objects List Panel and in the display window.

To select a selection group, use one of the following methods:

- In the Selection Groups view, select a selection group in the Group list.
- Choose **Objects** → **Selection Groups** → **Select Selection Group** → *group_name*, where *group_name* is one of the groups listed on the pull right menu.
- If selection groups named with the format Selection Group *number* exist, pressing a number key selects the corresponding selection group. (The keypad numbers do not support this operation.)

15.3.3. Adding Objects to a Selection Group

To add objects to a selection group:

1. Select the objects you want to add to a selection group.
2. Choose **Objects** → **Selection Groups** → **Add To Selection Group** → *group_name*, where *group_name* is one of the groups listed on the pull right menu.

15.3.4. Removing Objects from Selection Groups

There are two ways to remove an object from a selection group. If you remove all the objects from a selection group, the group continues to exist.

To remove an object from a selection group:

1. Select the objects you want to remove from a selection group.
2. Choose **Objects** → **Selection Groups** → **Remove from Selection Group** → *group_name*, where *group_name* is one of the groups listed on the pull right menu.

To remove an object from a selection group in the Selection Groups view:

1. Select the selection group from which you want to remove an object.
2. Select the object you want to remove.
3. Click the Remove Selected from Selection Group button ( ).

15.3.5. Renaming a Selection Group

To rename a selection group:

1. In the Selections Group view, double-click the name of the group you want to rename. It becomes editable
2. Type a new name for the selection group.
3. Press **ENTER**.

15.3.6. Deleting a Selection Group

Deleting a selection group does not delete the objects from the scenario. However, if you delete all the members of a selection group, the selection group also gets deleted.

To delete a selection group:

1. In the Selections Group view, select the group you want to remove.
2. Click the Delete button ( ).



16. Viewing Information about Objects

This chapter describes how to get information about simulation objects.

Displaying Information About an Object 16-3
Information Dialog Boxes for Entity-Level Scenarios 16-3
Information Dialog Boxes for Aggregate-Level Scenarios 16-5
Information Dialog Boxes for Tactical Graphics 16-7
Viewing Information for Multiple Objects 16-8
Displaying Entity Labels 16-9
Displaying Text Labels for 3D Models 16-12
Turning Entity Labels On and Off 16-13
Pinning 3D Text Labels to the Window 16-13
Displaying Labels for 2D Icons 16-14
Configuring Shadowing and Background Color for 2D Labels 16-16
Using Extended Labels 16-17
Adding an Extended Label 16-18
Changing an Extended Label's Index 16-19
Deleting an Extended Label 16-19
Changing Force Colors 16-19
Displaying HAT Lines for Entities (3D Only) 16-20
Displaying Entity Resources 16-21
Viewing Numerical Resource Data 16-21
Displaying Bounding Volumes 16-22
Enabling and Disabling Bounding Volumes for Entities 16-22
Showing Bounding Volumes Only for Selected Entities 16-23
Configuring Object Console Messages 16-23
Setting the Notification Level for Console Messages 16-24
Viewing All Console Messages 16-25
Answering Questions from Scripted Tasks 16-26

Saving Console Messages 16-26
Displaying an Object Console Warning Icon 16-27
Clearing the Object Console 16-27
Viewing Object Counts 16-28

16.1. Displaying Information About an Object

You can display information about individual objects, including entities, units, environmental objects, and Stealths. You cannot edit this information. You can also display entity labels. For details about entity labels, please see "Displaying Entity Labels," on page 16-9.

To display an object information dialog box:

1. Select an object.
2. Choose **Objects** → **Information**, or press `i`.



You can also open the Information dialog box as follows:

- Select a message in the Objects Console Summary Panel and choose **Information** on the shortcut menu.
- Click any field in the summary panel to open the relevant dialog box page.
- Click the Information button 4 on the Last Selected Object Panel to open the State Data page.
- Click the current task on the Last Selected Object Panel to open the Information dialog box to the Tasks page.

16.1.1. Information Dialog Boxes for Entity-Level Scenarios

An information dialog box in an entity-level scenario (Figure 16-1) has the following sections:

- Summary:
  - Simulation object icon.
  - Simulation object name.
  - Current task.
  - Sensor information.
  - Speed.
  - Altitude.
- Detailed information.
- Object console.

You can expand and contract the detailed information and object console sections to manage the amount of screen space used by the dialog box.

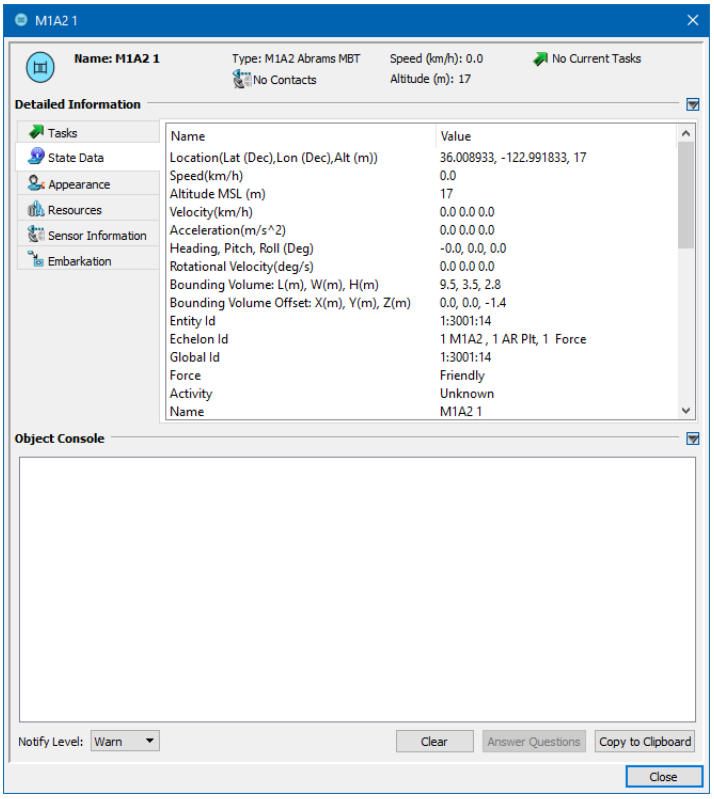


Figure 16-1. Information dialog box for simulation object in entity-level scenario

The Detailed Information section for simulation objects in an entity-level scenario has the following pages:

- Tasks. Shows the currently assigned task and any enabled reactive tasks. Allows you to cancel the current task.
- State Data. Shows location, movement values such as speed and heading, identifying information, and other status information.
- Appearance. Shows the status of appearance attributes, such as lights, fire-power, and concealment.
- Resources. Shows that amount of fuel and ammunition available.
- Sensor Information. Shows information about simulation objects being detected.
- Emitters. Shows details about emitters, if applicable.
- Acoustics. Shows details about active sonar and propulsion noise, if applicable.
- Embarkation. Lists embarked objects, if any.
- IFF. Shows IFF information, if applicable.



The pages listed for Detailed Information are context sensitive. If a page does not apply to an entity type, it is not shown.

The Detailed Information section for units in entity-level scenarios (Figure 16-2) has the following pages:

- Tasks.
- State Data.
- IFF (if applicable).
- Subordinates. Lists the subordinates.
- Acoustics (if applicable).
- Embarkation.

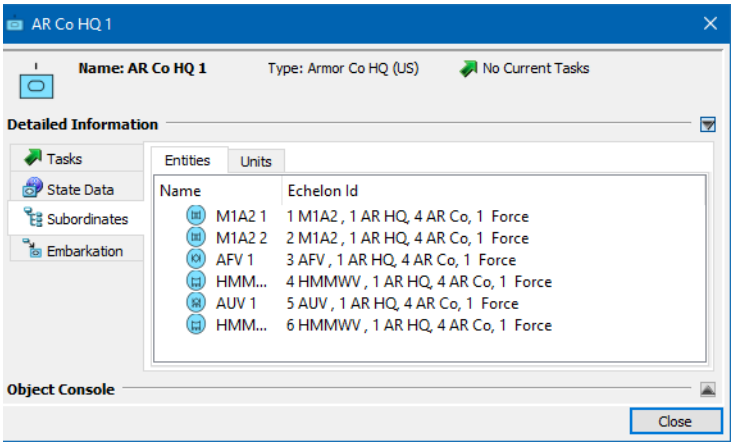


Figure 16-2. Information dialog box for units in entity-level scenarios

16.1.2. Information Dialog Boxes for Aggregate-Level Scenarios

An information dialog box for units in an aggregate-level scenario (Figure 16-3) has the following sections:

- Summary:
  - Simulation object icon.
  - Simulation object name.
  - Current task.
  - Sensor information.
  - Posture.
  - Health status.
  - Health and weapon status gumballs.
- Detailed information.
- Object console.

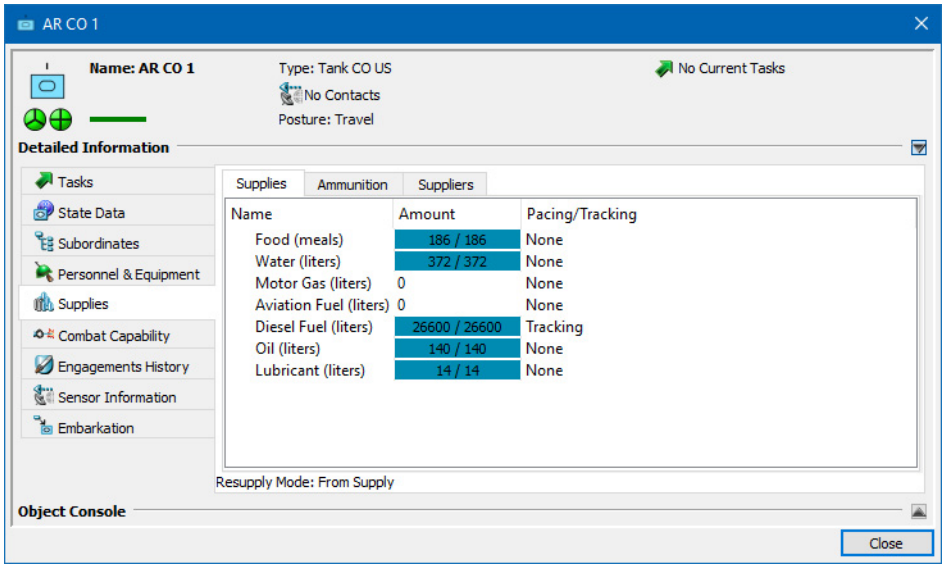


Figure 16-3. Information dialog box for simulation object in aggregate-level scenario

The summary section for units in aggregate-level scenarios (Figure 16-3) includes a health status bar and gumball icons that display information about the health of the simulation object. Simulation object icons can also display gumballs. (Gumballs are disabled by default. Health status is enabled.)

Figure 16-4 describes the status represented by different gumball states.

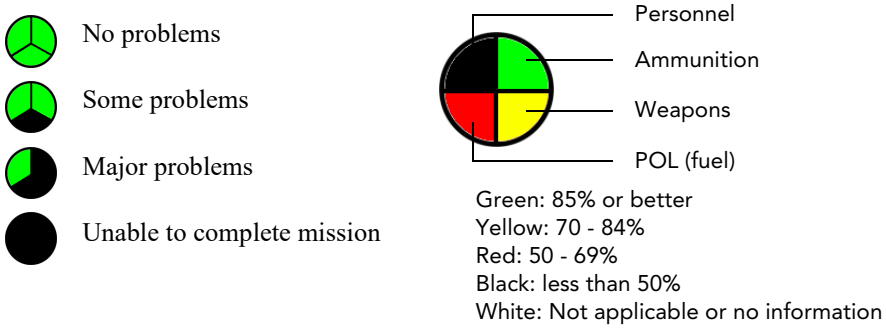


Figure 16-4. Gumball charts

Figure 16-5 illustrates the health status bar and gumball charts for a simulation object icon.

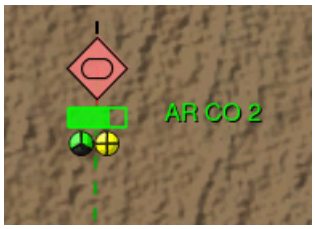


Figure 16-5. Icon with gumballs

You can expand and contract the detailed information and object console sections to manage the amount of screen space used by the dialog box.

The Detailed Information section for simulation objects in aggregate-level scenarios has the following pages:

- Tasks.
- State Data.
- IFF.
- Subordinates. Lists the subordinates.
- Personnel and Equipment. The types of personnel and number of each. Casualties, by category.
- Supplies. Food, water, fuel, and ammunition. The Supplies tab displays information when a simulation object is receiving supplies or providing them to another simulation object. When a simulation object is receiving supplies, an up arrow is displayed next to the supply it is receiving.
- Combat Capability. Combat power by category (anti-air, anti-personnel, and so on) and defensive capability, by category.
- Engagements. Current attack and defense status.
- Sensor Information. The Sensors tab shows information about simulation objects being detected. The Jamming tab shows if the simulation object is being jammed (electronic warfare) or is jamming other simulation objects.
- Emitters.
- Acoustics.
- Embarkation.

16.1.3. Information Dialog Boxes for Tactical Graphics

The detailed information section for an environmental object (Figure 16-6), such as a route, has the following pages:

- State Data. Shows the force and coordinates.
- Attached To. The simulation object this object is attached to, if any.

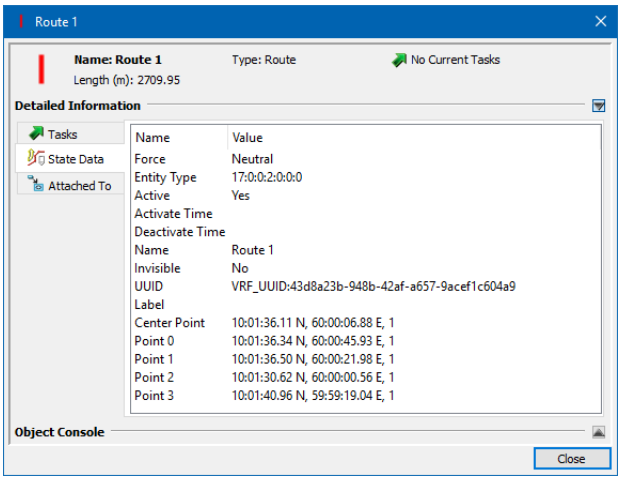


Figure 16-6. Environmental object information dialog box

16.1.4. Viewing Information for Multiple Objects

You can select multiple objects and view summary information for all of them. From the summary information window, you can jump directly to an object and you can open its Information dialog box.

To view information about multiple objects:

1. Select the objects.
2. Choose **Objects** → **Information**, or press `i`. The Multiple Object Information window opens (Figure 16-7).

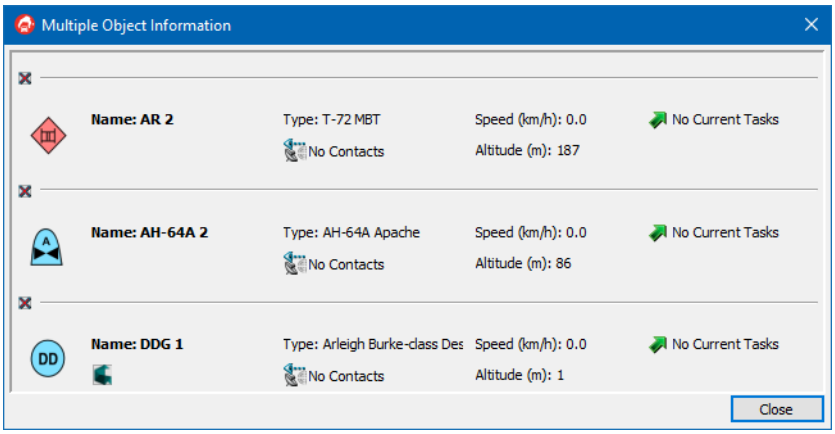


Figure 16-7. Multiple Object Information window

3. To zoom to the extents of an object, click its icon.
4. To open an Information dialog box for an object, click any of the details about the object, such as its name.

16.2. Displaying Entity Labels

VR-Forces provides extensive information about simulation objects in the Information dialog box ("Displaying Information About an Object," on page 16-3). It also provides an abbreviated set of information in the form of simulation object labels.

For 3D models, VR-Forces displays labels that help you locate and identify entities (Figure 16-8). The label consists of a locator circle, which is always visible when labels are enabled, and a text label that is displayed when you mouse over an entity. All entities within the view have locator circles, whether the model is visible or not. The color of an entity's label represents its force – blue for friendly, red for opposing, and green for neutral.

A 3D text label displays the entity's:

- Marking text.
- Entity type enumeration.
- Force.
- Name.
- Speed.
- Heading.
- Latitude and longitude.
- Altitude.



You can configure the attributes of the entity label, such as font size, and the height and width of the background by editing the entityInfo model definition in the Visual Models Editor. For details about editing model definitions, please see "Creating and Editing Model Definitions," on page 87-2.

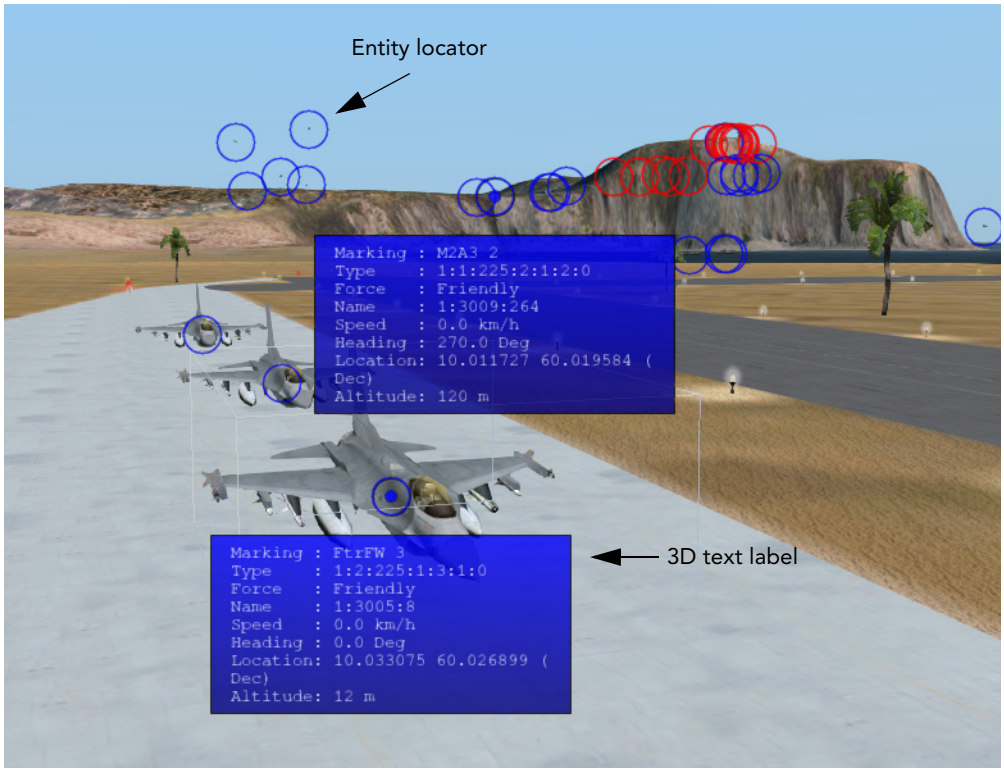


Figure 16-8. Entity labels for 3D models

For 2D icons, labels are a set of individually enabled text labels, called symbol decorations (Figure 16-9). Entity labels for 2D icons are supported both in the 2D view and the 3D view. If you resize 2D icons, the labels also get resized. 2D icons can display the simulation object's:

- Acceleration.
- Altitude above mean sea level.
- Altitude above ground level.
- Heading Indicator.
- Label. If extended labels exist, each label can be displayed separately.
- Location.
- Name.
- Orientation.
- Velocity.
- Weather. Cloud state, precipitation intensity, visibility, wind direction, and wind speed.

If spot reports are enabled, you can display labels for spot reports. For details, please see "Displaying Labels for Spot Reports," on page 9-8.

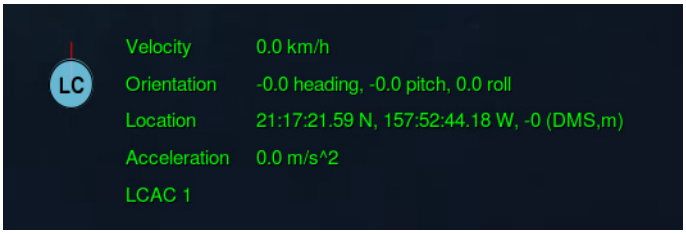


Figure 16-9. Label for a 2D icon

16.2.1. Displaying Text Labels for 3D Models

To view a 3D text label, use one of the following methods:

- Move the cursor over the circle that represents the entity.
- On the Display Settings dialog box, Entity Display Settings page, select Show Information Only For Selected Entities (Figure 16-10). Then select the entity whose label you want to see.

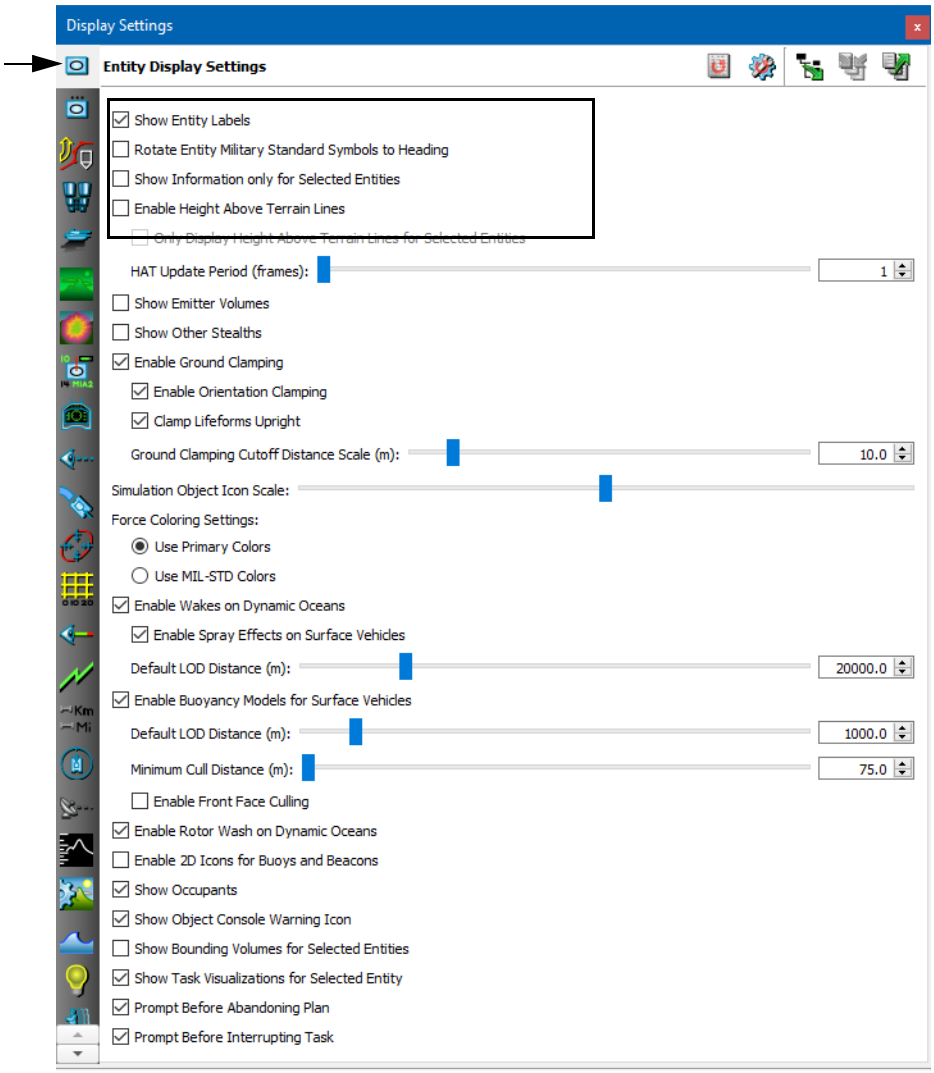


Figure 16-10. Display Settings, Entity Display Settings page

16.2.2. Turning Entity Labels On and Off

To turn the display of entity labels on or off:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 16-10).
3. Select or clear the Show Entity Labels check box.



You can also enable and disable entity labels by choosing **Settings** → **Entity Labels**, pressing **L**, or by clicking the Entity Labels button () on the Display Settings Toolbar.

16.2.3. Pinning 3D Text Labels to the Window

You can pin 3D text labels to the window, making them stay visible after you move the cursor away from the entity.

To pin a 3D text label to the window or hide a pinned label:

1. Right-click an entity's locator circle. A menu is displayed.
2. Choose **Toggle Pinned Label**.

Pinning Multiple 3D Text Labels

You can display multiple text labels at one time.

To pin multiple text labels use either of the following methods:

- Right-click and choose **Toggle Pinned Label** for each entity whose label you want to pin.
- On the Display Settings dialog box, Entity Display Settings page, select Show Information Only For Selected Entities (Figure 16-10). Then, in the Objects List Panel, multi-select the entities whose labels you want to see.

16.2.4. Displaying Labels for 2D Icons

You can enable and disable the display of labels for 2D icons by label type. This allows you to control how much screen space is taken up by the labels. You can control the display of labels in the Display Settings dialog box or on the Display Settings Toolbar.

To enable and disable display of 2D icon labels:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 16-11).

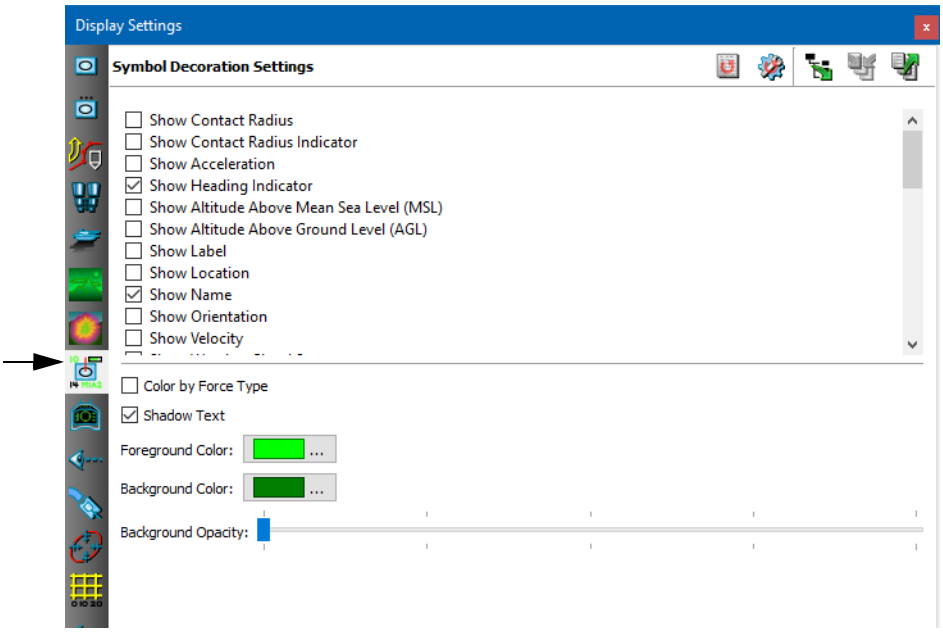


Figure 16-11. Symbol Decoration Settings page

3. Select the check box for each label that you want to display. Clear the check box for each label that you want to hide.

Changing the Color for 2D Labels

The labels for 2D icons use the color specified by the Foreground Color setting on the Symbol Decoration Settings page. You can also specify that labels use the force color.

To change the color of 2D labels:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 16-11).
3. To set the color to any color:
   a. Click the Foreground Color color swatch. A color picker dialog box opens.
   b. Select the color you want to use.
   c. Click OK.
4. To use the force color for labels, select the Color by Force Type check box. This setting overrides the Foreground Color setting.

Enabling 2D Labels on the Display Settings Toolbar

The Entity Labels button on the Display Settings Toolbar lets you open a panel that lists the symbol decorations that you can show or hide. It also lets you color labels by force type.

To enable or disable 2D labels on the Display Settings Toolbar:

1. Click the arrow on the Entity Labels button ( ). A small panel is displayed. It is a persistent window.

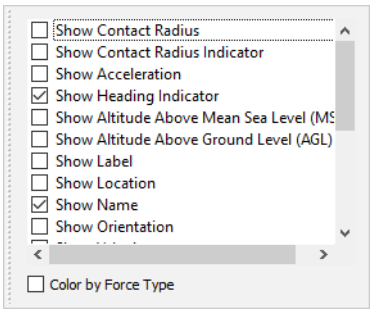


Figure 16-12. Symbol Decoration Panel

2. Select the symbol decorations that you want to display. Clear the ones you want to hide.
3. To hide the Symbol Decorations Panel, click the arrow on the Entity Labels button.

16.2.5. Configuring Shadowing and Background Color for 2D Labels

2D labels are drawn on a background that, by default, is transparent. However, you can increase the opacity of the background to make it visible and can change the background color.

You can add shadowing to the text of 2D labels. This may improve visibility when simulation objects are on light colored terrain.



Figure 16-13. 2D label with background

To set the color and opacity of the 2D label background:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 16-11).
3. To change the opacity of the background, adjust the Background Opacity slider.
4. To change the color of the background:
   a. Click the Background Color color swatch. A color picker opens.
   b. Select the color you want to use for the background.
   c. Click OK.

To configure shadows for 2D labels:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 16-11).
3. Select or clear the Shadow Text check box.

16.3. Using Extended Labels

One of the ways to identify a simulation object is by its label. (For more information, please see "Labels," on page 13-4.) Extended labels are additional labels that you can specify on an application-wide basis and define individually for simulation objects. The contents of each label gets sent over the network as part of the simulation object's information and is displayed in its Information dialog box.

You can add and remove extended labels and change the label name. Changing a label's name does not affect the text that is associated with the label for those simulation objects that use the label.



Extended labels are not scenario-specific. They affect all scenarios. Therefore, if you change a label's name or index, or delete it, be sure that you want this change to apply to all scenarios in which you use the extended label.

Figure 16-14 shows an extended label (called A Special Label) added in the Application Settings dialog box, Simulation Settings page. Figure 16-15 shows the text for the extended label for a specific entity entered in the Edit dialog box and then displayed on the screen.

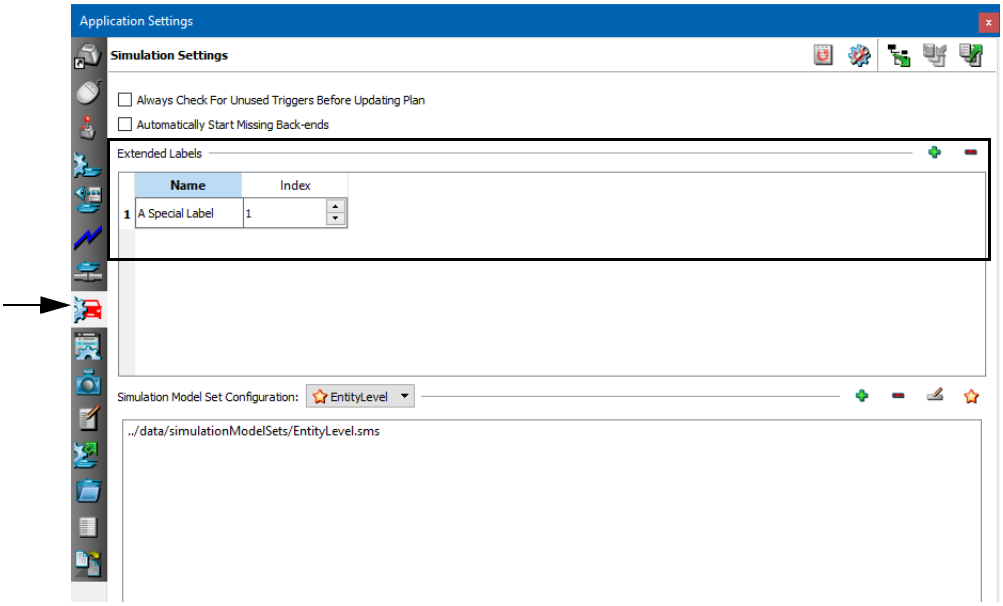


Figure 16-14. Extended label added to session

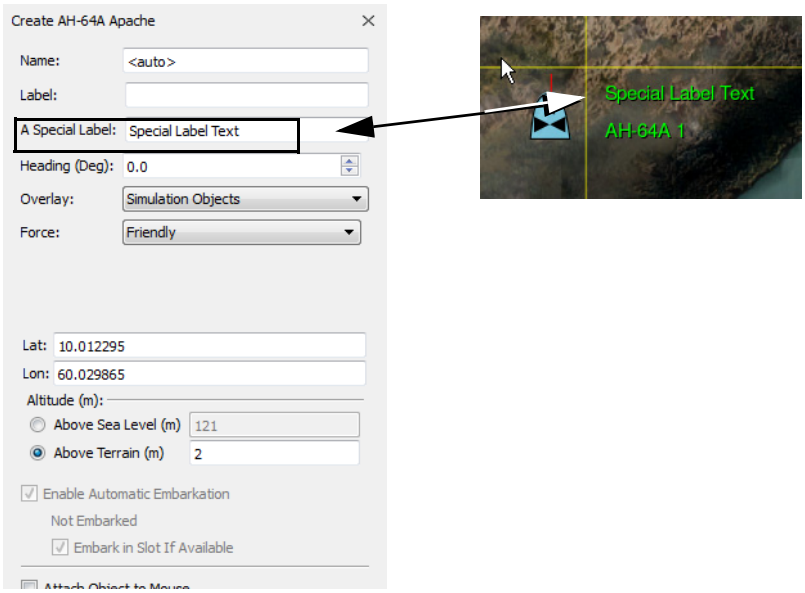


Figure 16-15. Extended label text for a simulation object

You can display extended labels in Plan View mode the same as other symbol decorations. For more information, please see "Displaying Labels for 2D Icons," on page 16-14.

16.3.1. Adding an Extended Label

To add an extended label:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 16-14).
3. Click the Add button (+) on the Extended Labels line. A label is added to the list window. It has a generic name (for example, Label 1).
4. Double-click the label name to make it editable.
5. Type the name you want for this label.
6. Edit simulation objects to specify the text for this label.

16.3.2. Changing an Extended Label's Index

A label's index is used to store the label text and identify it when sent over the network. In Edit dialog boxes, extended labels are listed in indexed order. You can change an extended label's index. However, while changing an extended label's name does not affect the text that you save for a simulation object, changing its index could affect existing scenarios. Therefore, use caution in doing so.

To change an extended label's index:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 16-14).
3. Click the up or down arrow next to a label's index number. The index number changes and, if applicable, the label moves up or down in the list.

16.3.3. Deleting an Extended Label

Deleting an extended label affects all scenarios that use that label. (In other words, it is not a scenario-specific setting.) If you delete an extended label, it is immediately removed from all simulation objects that are using that label.

To delete an extended label:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 16-14).
3. In the Extended Labels window, select the label you want to delete.
4. Click the Delete button ( ).

16.4. Changing Force Colors

By default, labels, detonation lines, and other force-related simulation object graphics (including the 3D Colorized Model model set) are displayed using primary colors (red, blue, and green). You can change the force coloring to use MIL-STD 2525B colors (light blue, light green, and salmon).

You can also change the color of a force in the Simulation Object Editor. For details, please see "Editing a Force," on page 67-33.

To change force coloring:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. For entities and entity graphics, select the Entity Display Settings page (Figure 16-10).

   For units, select the Unit Display Settings page (Figure 24-3).
3. Select the Force Coloring Settings option that you want.

16.5. Displaying HAT Lines for Entities (3D Only)

Height-above-terrain (HAT) lines are vertical lines that run from an entity to the ground and display the altitude of the entity (using whatever units are configured on the Display Units Settings page) (Figure 16-16). They help you understand the relationship of the entity to the terrain.

HAT lines are shown only for entities that could be above or below the ground, such as fixed and rotary-wing vehicles. You can show and hide these lines globally and you can specify that they be shown only for the selected entities.

When you create or edit an entity, VR-Forces displays HAT lines even if they are otherwise disabled.

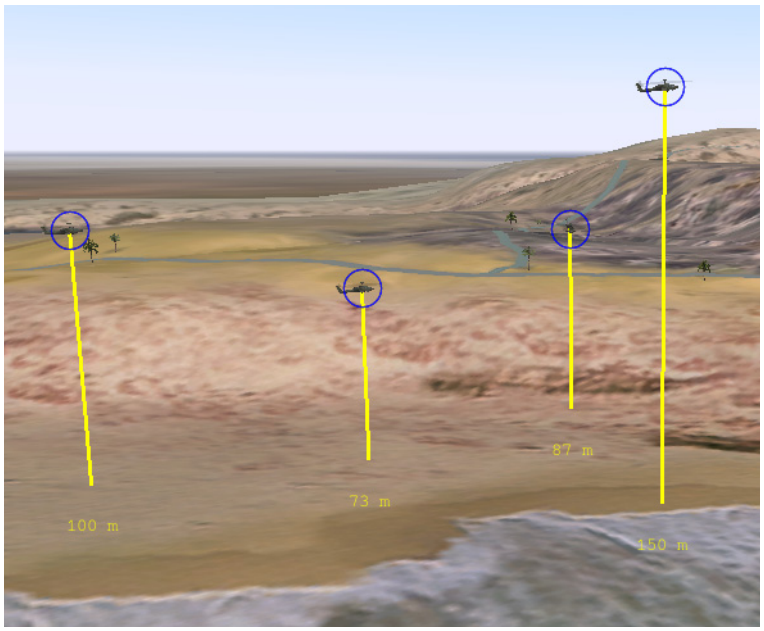


Figure 16-16. Entity height-above-terrain lines

! Displaying height above terrain lines can reduce the frame rate.

To show or hide height-above-terrain lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 16-10).
3. To display lines, select the Enable Height Above Terrain Lines check box.
4. To display lines only for selected entities, select the Only Display Height Above Terrain Lines For Selected Entities check box.



You can also enable and disable entity HAT lines by choosing **Settings** → **Entity Height Above Terrain Lines** or by clicking the Entity Height Above Terrain Lines button () on the Display Settings Toolbar.

16.6. Displaying Entity Resources

VR-Forces tracks the amount of resources that a simulation object has. You can view the status of resources in the Information dialog box.

16.6.1. Viewing Numerical Resource Data

The Information dialog box lists the simulation object's current amount of resources, its starting amount, and the percentage remaining (Figure 16-17).

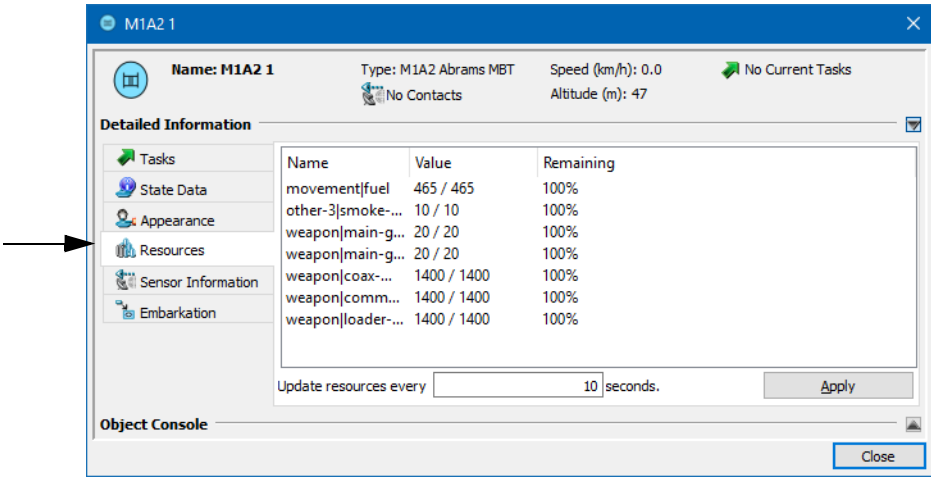


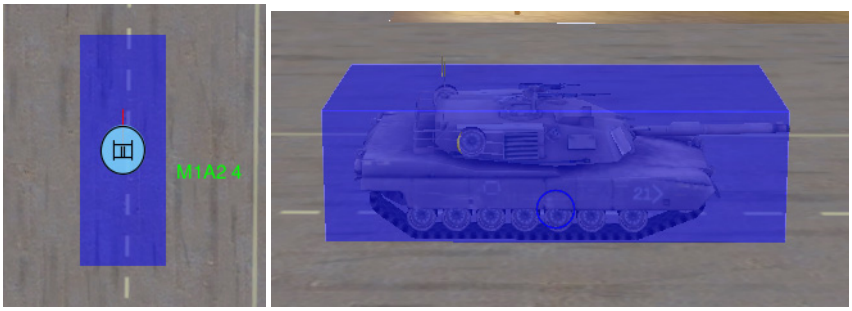
Figure 16-17. Resource Information page

To view a simulation object's resource status:

1. Select the simulation object.
2. Choose **Objects** → **Information**, or press `i`.
3. Expand the Detailed Information section.
4. Select the Resources page.

16.7. Displaying Bounding Volumes

You can display bounding volumes for simulation objects. Bounding volumes show the area around a simulation object that the back-end uses for intersection calculations. Bounding volumes are shown as translucent volumes in 3D and translucent rectangles in 2D (Figure 16-18). For simulation objects in aggregate-level scenarios, the bounding volume is equivalent to the footprint.



2D 3D

Figure 16-18. Bounding volumes

Bounding volumes for units and bounding volumes for individual entities are managed by different settings. For information about displaying bounding volumes for units, please see "Displaying Unit Icons and Bounding Volumes," on page 24-8.

16.7.1. Enabling and Disabling Bounding Volumes for Entities

Display of bounding volumes for entities is an observer setting.

To enable or disable bounding volumes for individual entities:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 44-7).
3. Select the observer mode for which you want to enable or disable bounding volumes.
4. Select or clear the Bounding Volume check box.



You can also enable and disable bounding volumes for entities by choosing **Observer** → **Bounding Volumes** or clicking the Bounding Volumes button ( ) on the Observer Settings Toolbar.

16.7.2. Showing Bounding Volumes Only for Selected Entities

You can configure VR-Forces so that it shows bounding volumes only for selected simulation objects.

To display bounding volumes only for selected entities:

1. Enable bounding volumes, as described in "Enabling and Disabling Bounding Volumes for Entities," on page 16-22.
2. Choose **Settings** → **Display**. The Display Settings dialog box opens.
3. Select the Entity Display Settings page (Figure 16-10).
4. Select the Show Bounding Volumes for Selected Entities check box.

16.8. Configuring Object Console Messages

The Information dialog boxes for simulation objects and tactical graphics have a console that displays messages sent from the simulation engine, from a simulation object's plan, from other simulation objects, and from scripts. You can set the notification level that controls the display of these messages. You can save them to the clipboard, and you can clear the display. These messages are also sent to the Object Console Summary Panel.

Scripted tasks can ask questions. You can answer these questions from the Object Console Summary Panel.

16.8.1. Setting the Notification Level for Console Messages

Console messages are displayed only if they match or exceed the notification level set for that object. The notification levels are:

- 0 – Only fatal messages are printed.
- 1 – Warning messages and fatal messages are printed.
- 2 – Some diagnostic information is printed with warnings and fatal messages.
- 3 – Verbose information is printed.
- 4 – Debug information is printed.

The default notification level is 2.

The notification level is set individually for each object. You can set the notification level in the object's Information dialog box, with the Notify Level set data request (for simulation objects), or with the **objectConsoleNotifyLevel** parameter in *vrfSim.mtl*. (For the set data request procedure, please see "Notify Level," on page 34-28.)

To set the notification level for an object's console in the Information dialog box:

1. Select the simulation object.
2. Choose **Objects** → **Information**. The Information dialog box opens.
3. Expand the Object Console section (Figure 16-19).

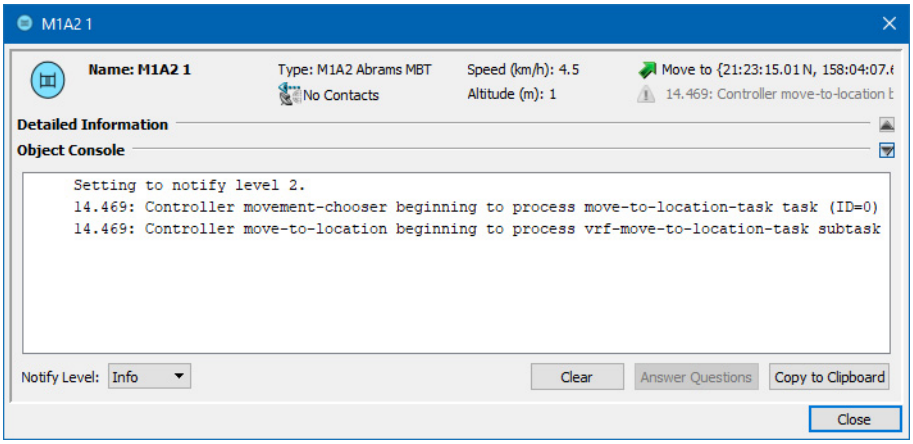


Figure 16-19. Information dialog box, Object Console

4. Select a notification level from the Notify Level list.

16.8.2. Viewing All Console Messages

You can view all object console messages in one place – the Object Console Summary Panel. Messages are displayed based on the notification levels of the individual simulation objects in the scenario. You can set the notification level in a simulation object's Information dialog box or with the Notify Level set data request.

When VR-Forces sends a message to a simulation object's console, it displays an exclamation point next to the entity until you view the message (unless you disable this feature). The messages in the Object Console Summary Panel also display an exclamation point until you open the Information dialog box for that entity or click Acknowledge All.

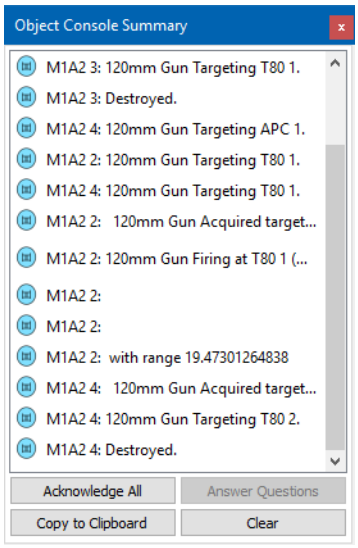


Figure 16-20. Object Console Summary Panel

➤ To display the Object Console Summary Panel, choose **View** → **Object Console Summary Panel**.

➤ To remove the exclamation point from messages in the Object Console Summary Panel, click Acknowledge All.

16.8.3. Answering Questions from Scripted Tasks

Scripted tasks can send questions to the object console. Questions are indicated by a question mark next to the message (Figure 16-21). The script will offer a set of responses as buttons, option buttons, or a list. You can answer questions from the Object Console and the Object Console Summary Panel.

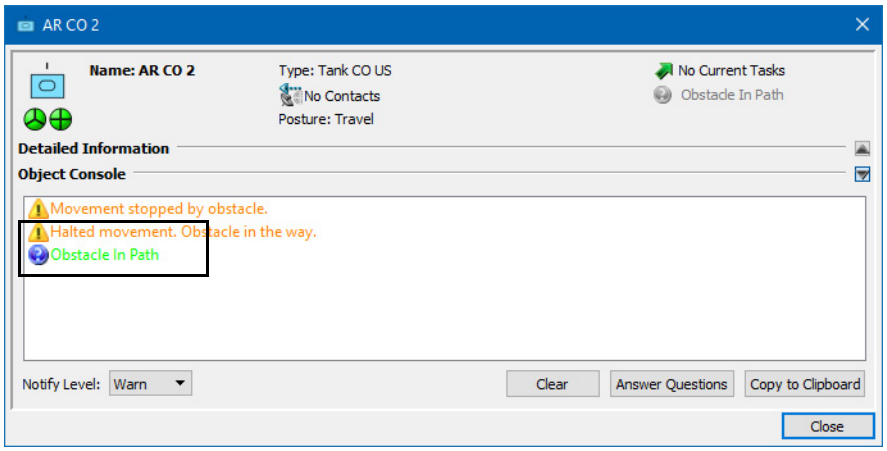


Figure 16-21. Question in console

To answer a question from a scripted task:

1. On the Object Console or Object Console Summary Panel, click Answer Questions. A dialog box with the possible answers is displayed.
2. Choose an option.
3. Click OK.

16.8.4. Saving Console Messages

If you want to save the messages that are sent to the console, you can copy them to the clipboard and then paste them into a file.

To save console messages to the clipboard:

1. Select the simulation object.
2. Choose **Objects** → **Information**. The Information dialog box opens (Figure 16-19).
3. If the Object Console is collapsed, expand it.
4. To save all messages in the console, click Copy to Clipboard. To save specific messages, select them and click Copy to Clipboard.
5. Paste the text into the text editor or word processor of your choice.

16.8.5. Displaying an Object Console Warning Icon

VR-Forces can display a notification icon when an object receives a message on its object console (Figure 16-22). This option is enabled by default, however the default notification level is Warn and objects do not typically get messages at this level. The more likely use for this icon is to set the notification level to Info or lower and then check the console when a notification is received. For details about how to set the notification level, please see "Setting the Notification Level for Console Messages," on page 16-24.



Figure 16-22. Object console warning icon

Icons are displayed next to the entity icon in the display window, next to the entity's listing in the Objects List Panel, and if the entity has recently been selected, on the Last Selected Object Panel.

The icon is not displayed if the Information dialog box is open. If an icon is displayed for an object, when you open its Information dialog box, the icon is removed.

Is you disable the warning icon, it is no longer displayed in the display window or the Objects List Panel. It is still displayed on the Last Select Object Panel, if the entity was the last selected object.

To enable or disable the console warning icon:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 16-10).
3. Select or clear the Show Object Console Warning Icon check box.

16.8.6. Clearing the Object Console

➤ To clear the messages from the object console, click Clear.

16.9. Viewing Object Counts

You can display the number of live and destroyed simulation objects in the simulation you are viewing and the number of control objects, by type.

➤ To display the object counts, choose **View** → **Objects Count Panel**. The Objects Count Panel is added to the display (Figure 16-23).

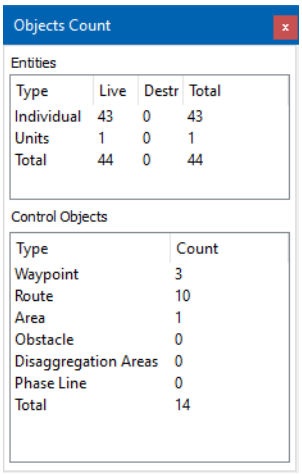


Figure 16-23. Objects Count Panel



17. Embarked and Embedded Entities

This chapter explains how to embark entities on other entities. It also explains how to manage embedded entities, which is another way to embark entities.

Embarking and Disembarking Simulation Objects 17-2
Embarking and Disembarking Simulation Objects Instantly 17-4
Disembarking Simulation Objects Using the Disembark Command 17-7
Embedded Entities 17-8
Configuring an Entity to Support Embedded Entities 17-9
Assigning Tasks and Plans to Embedded Entities 17-11
Restoring an Entity that has Embedded Entities 17-11

17.1. Embarking and Disembarking Simulation Objects

Embarkation places simulation objects into or onto other simulation objects, for example, dismounted infantry in a truck, trucks on an LCAC, or aircraft on an aircraft carrier. While embarked, the simulation objects travel with the simulation object on which they are embarked. Embarked simulation objects are not visible in the 2D view. They are visible in the 3D view if they are embarked on the ourside of the parent, such as on the deck of an aircraft carrier, or inside a parent that has interior geometry, such as some helicopters.

Embarked entities can shoot while they are embarked if they are embarked on an entity that has object geometry enabled.

If you are using HLA and want to use the embarkation feature, you must use RPR FOM 2, draft 17 or later. VR-Forces includes configurations in the Simulation Connections Configuration dialog box that start VR-Forces using the correct FED file and FOM Mapper for HLA 1.3 and HLA Evolved using RPR FOM 2.

You can attach control objects to simulation objects so that embarked simulation objects can use them to move about on the parent simulation object. For example, an aircraft could taxi towards a waypoint on the flight deck of an aircraft carrier.



Attached objects are shown in the Embarkation View on the Objects List Panel.

The embarkation status of simulation objects is displayed in the Embarkation View of the Objects List Panel (Figure 17-1).

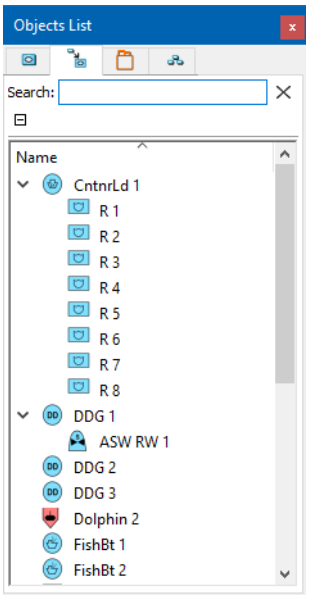


Figure 17-1. Embarkation View

You can embark and disembark simulation objects instantly (described in this section), or you can have them simulate embarkation and disembarkation using the Embark and Disembark tasks, described in Chapter 29, *Embarkation, Wait, Radio, and Other Tasks*.

When you embark a simulation object using the Embark task, the simulation object can embark only on simulation objects that are configured to accept that type of simulation object. For example, LCACs are configured to accept several types of ground vehicles. You cannot give a DI an Embark task to embark on an LCAC. However, when you use the immediate embark method to embark a simulation object, you can use preconfigured slots (spaces) for simulation objects or specify an override. When you specify an override you can embark a simulation object on any other simulation object at the coordinates you enter. However, it is your responsibility to know what the proper coordinates are.



Embarked entities controlled by joystick cannot disembark. For example, if you use a joystick to control a fixed-wing taking off from a carrier, the aircraft will take off, but it will remain in an embarked state.

17.1.1. Embarking and Disembarking Simulation Objects Instantly

You can embark and disembark simulation objects instantly in the following ways:

- Using the Embark On command.
- Using set data requests.
- In the Embarkation View.
- Using automatic embarkation.

Using set data requests lets you include immediate embarkation in plans. This may be particularly useful in a global plan, in which you may want to quickly create and embark simulation objects without paying attention to the realism of the action. For details about using set data requests for embarkation, please see "Embarked," on page 34-16 and "Disembarked," on page 34-16.

Embarking Simulation Objects Instantly Using the Embark On Command

The Embark On command lets you embark simulation objects using preconfigured embarkation spaces or using specific coordinates. If the embarkation spaces on the parent simulation object are full, VR-Forces sends a message to the back-end console and the simulation object is not embarked. To embark simulation objects instantly from a plan, use the Embarked set data request.

To embark a simulation object instantly using the Embark On command:

1. Select the simulation object that you want to embark.
2. Choose **Objects** → **Embarkation** → **Embark On**. The Embark On dialog box opens.

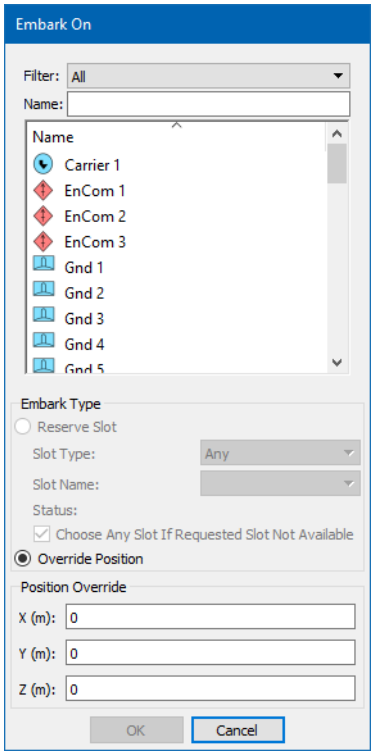


Figure 17-2. Embark On dialog box

3. Optionally, filter the list of potential parent simulation objects.
4. Select the parent simulation object.
5. To embark in a configured space, select the Reserve Slot option. To specify an embarkation point, select Override Position.
6. If you chose the Override Position option, specify the coordinates on the parent, in body coordinates, for the embarkation point.
7. Click OK.

Embarking Simulation Objects in the Embarkation View

To embark an object in the Embarkation View:

1. On the Objects List Panel, select the Embarkation View.
2. Drag the simulation object you want to embark onto the parent simulation object. The Embarked dialog box opens.
3. To specify an embarkation location, specify X, Y, and Z values. If you do not specify a location, the simulation object gets embarked at the center of the parent simulation object.
4. Click OK.

Embarking Simulation Objects Using Automatic Embarkation



Automatic embarkation lets you embark simulation objects as you create them. It has the following conditions:

- It is only available in 3D observer modes.
- The simulation object you want to embark must be capable of embarking (as configured in the OPD).
- The simulation object you want to embark on must allow embarkation.
- There is no option to override embarkation slots by specifying a particular location.
- If slots are configured on the parent simulation object and you select the option to use slots, embarked simulation objects are placed in slots until the slots run out. Then they are placed on the parent simulation object where you click.



You can attach tactical graphics to simulation objects using this method. For more information, please see "Attaching a Tactical Graphic Automatically," on page 40-16.

To embark a simulation object using automatic embarkation:

1. Create the entity on which you want to embark another entity (if it is not already created.)
2. In the Simulation Objects Palette, select the entity that you want to embark.
3. Optionally, configure auto-embarkation, as follows:
   a. If the Create Entity dialog box is not open, click its tab to open it.
   b. To enable or disable automatic embarkation, select or clear the Enable Automatic Embarkation check box.
   c. To embark the entity in a slot if one is available, select the Embark in Slot if Available check box.

4. Move the cursor over the entity you want the new entity to embark on. If it supports embarkation and automatic embarkation is enabled, a green box is displayed (Figure 17-3). (If the entity does not support automatic embarkation, no green box is displayed.) If the create dialog box is open, the Not Embarked label changes to Embark On *entity_name*.

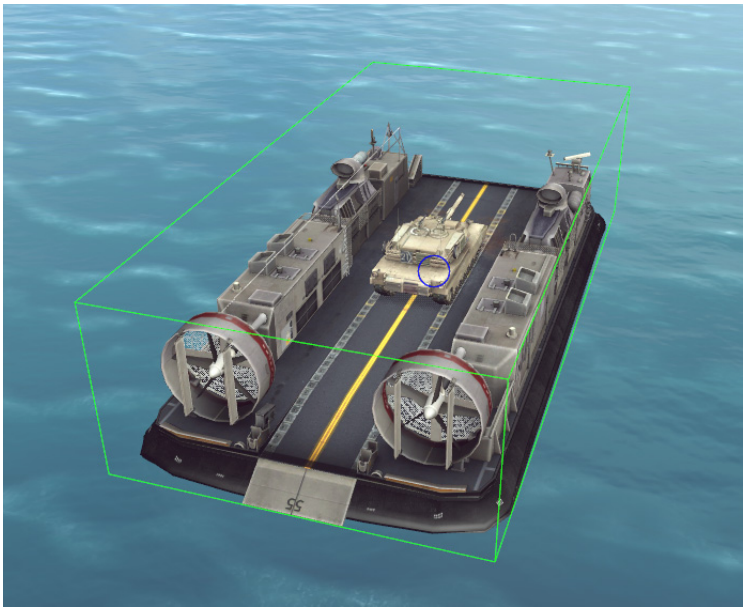


Figure 17-3. Auto embarkation

5. Click to place the entity.

17.1.2. Disembarking Simulation Objects Using the Disembark Command

Like the Embark On command, the Disembark command is typically used as part of scenario development. To simulate disembarkation, you would use the Disembark task. To provide instant disembarkation in a plan, you would use the Disembarked set data request.



Embarked entities controlled by joystick cannot disembark. For example, if you use a joystick to control a fixed-wing taking off from a carrier, the aircraft will take off, but it will remain in an embarked state.

To disembark a simulation object instantly:

1. Select the simulation object that you want to disembark.
2. Choose **Objects** → **Embarkation** → **Disembark**. If the simulation object was embarked using the default embarkation point, it is disembarked next to the former parent simulation object. If the simulation object was embarked using a position override, it is disembarked in the same location as the former parent.

To disembark a simulation object in the Embarkation View:

1. Open the Embarkation View.
2. Drag the simulation object you want to disembark from the parent simulation object to the force level.

Disembarking All Simulation Objects

When you disembark all simulation objects from a parent, the placement of the simulation objects depends on how they were embarked. If the simulation objects were embarked with the Embark On command or the Embark task, they are disembarked to separate locations near the parent. If they were embarked with the Embarked set data request, they are all disembarked in the same location in the center of the parent simulation object. Since this behavior is usually not desirable, you should understand how you embarked the simulation objects before you use this command.

To disembark all simulation objects from a parent entity:

1. Select the parent simulation object.
2. Choose **Objects** → **Embarkation** → **Disembark All**.

17.2. Embedded Entities

Embedded entities are a convenient way to manage entities that are typically embarked on other entities.

Ships, planes, and vehicles often carry other platforms or personnel. You can simulate this relationship by embarking entities on other entities. However, this involves creating the embarking entities, giving them plans, embarking them, disembarking them, and otherwise managing the relationship. This can be inconvenient if you need to simulate large numbers of embarked entities, such as the airplanes and helicopters on an aircraft carrier or multiple personnel on a personnel carrier.

The embedded entity feature lets you configure entities with a set of embedded entities. You can then deploy the embedded entities and give them tasks or plans. Once deployed, there is no difference between an embedded entity and any other entity, except that a parent entity can recover a deployed entity. You do not need to manage the initial process of creating the relationship between the entities via embarkation and disembarkation.

For details about deploying and recovering embedded entities, please see "Embedded Entity Tasks," on page 29-19.

17.2.1. Configuring an Entity to Support Embedded Entities

Embedded entities are configured in systems, which you then add to an entity in the Simulation Object Editor. An entity can have only one embedded entity system. Therefore, if you need different configurations of embedded entities for different parent entities, you must create a system for each configuration.

Within a configuration, you must specify each embedded entity. In other words, you cannot specify a system as containing, for example, 5 helicopters and 40 F/A-18 jets. In this case the system would have to individually specify each helicopter and each jet.

A system definition file for embedded entities specifies the entities and the metadata needed to edit properties in the Simulation Object Editor. You must give each entity a unique name.

An entity is formatted as follows (from *./data/simulationModelSets/EntityLevel/vrfSim/systems/other/ddg-embedded-support-craft.sysdef*):

```
(embedded-entities
  (ee-1
    (entity-name  "SH-60 Seahawk Alpha")
    (entity-type  1 (1 2 225 22 3 1 0))
    (deployment-time 0.0)
    (recovery-distance 35.0)
    (relative-position $rel-pos-lamps-a)
    (relative-heading $rel-heading-lamps-a)
  )
...
  (ee-2
    (entity-name  "RHIB Alpha")
    (entity-type  1 (1 3 225 63 8 0 0))
    (deployment-time 15)
    (recovery-distance 50.0)
    (relative-position $rel-pos-rhib-a)
    (relative-heading $rel-heading-rhib-a)
  )
  ...
)
```

The relative-position and relative-heading parameters are variables that link to the metadata, as follows:

```
(parameter-data-list
  (vector-parameter-data
    (parameter-name  "rel-pos-lamps-a")
    (variable-type  "DtRwVector")
    (display-label  "Seahawk Alpha's relative position")
    (display-units  "meters")
    (source-units  "meters")
    (default-value  -75.000000 0.000000 -6.500000)
    (relative-to  "")
  )
  (vector-parameter-data
    (parameter-name  "rel-pos-rhib-a")
    (variable-type  "DtRwVector")
    (display-label  "Rigid-hull Alpha’s relative position")
    (display-units  "meters")
    (source-units  "meters")
    (default-value  -75.000000 10.000000 -6.500000)
```

```
      (relative-to  "")
    )
    ...
    (real-parameter-data
      (parameter-name  "rel-heading-lamps-a")
      (variable-type  "DtRwReal")
      (display-label  "Seahawk Alpha's relative heading")
      (display-units  "degrees")
      (source-units  "degrees")
      (default-value  0.000000)
    )
    (real-parameter-data
      (parameter-name  "rel-heading-rhib-a")
      (variable-type  "DtRwReal")
      (display-label  "Rigid-hull Alpha's relative heading")
      (display-units  "degrees")
      (source-units  "degrees")
      (default-value  90.000000)
    )
    ...
  )
```

Figure 17-4 illustrates the properties dialog box created for the system shown in the previous paragraphs.

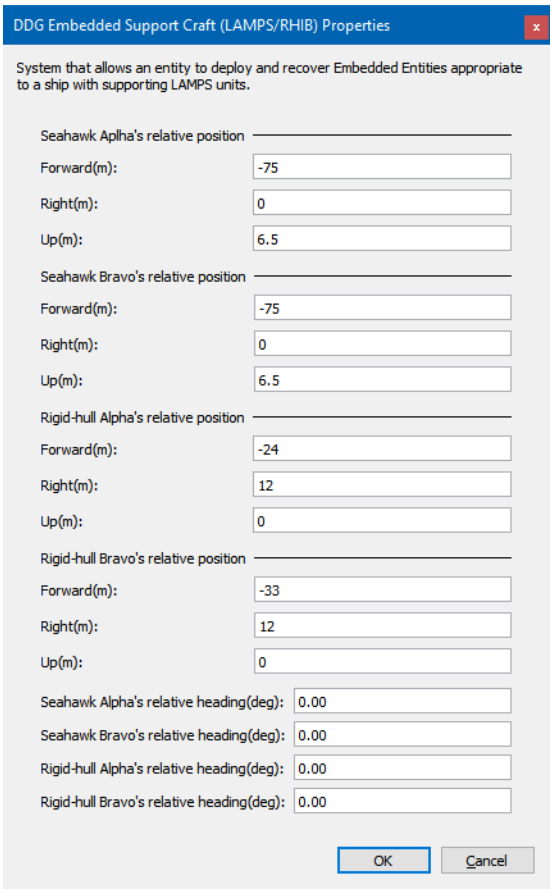


Figure 17-4. Embedded Entity Properties dialog box

17.2.2. Assigning Tasks and Plans to Embedded Entities

Once an embedded entity is deployed, it is just like any other entity in the scenario. You can give it independent tasks or write a plan for it. Follow the standard procedures for these activities.

17.2.3. Restoring an Entity that has Embedded Entities

If you deploy an embedded entity and it gets destroyed, it is removed from the scenario and is not available to be deployed again. However, if you restore a parent entity and it had deployed entities that were subsequently destroyed, it is restored to the full complement of embedded entities. Entities that are deployed at the time the parent is restored are not affected. They are not duplicated in the parent's inventory and their relationship is not affected.



18. Generating Traffic (Pattern of Life)

This chapter explains how to automatically generate pattern of life simulation object traffic and how to create and task pedestrian crowds.

Generating Traffic 18-2
Creating Spawn Points 18-3
Activating and Deactivating Spawn Points 18-4
Creating Sink Points 18-5
Spawn Patterns 18-6
How Spawn Events are Scheduled 18-7
Spawning Entities in Bursts 18-9
Using a Custom Plan 18-10
Default Spawn Patterns and Scenario-Specific Patterns 18-10
Creating a Spawn Pattern 18-11
Copying a Spawn Pattern 18-14
Editing a Spawn Pattern 18-15
Deleting a Spawn Pattern 18-16

18.1. Generating Traffic

The VR-Forces traffic feature (also called spawn and sink or pattern of life) lets you automatically generate background traffic for your scenario.

Automatically generated (spawned) simulation objects are created at spawn points. The default behavior for a spawned simulation object is to randomly choose a destination, called a sink point, and move towards that point. When it arrives at the sink point it is removed from the scenario. This creates the effect of purposeful activity in an area of the terrain without the need to create and task simulation objects individually.

As an alternative to the default behavior, you can write a plan that is assigned to each simulation object created at a particular type of spawn point. This plan can include any tasks and set data requests that are appropriate for the object type.

The behavior of a spawned simulation object, as well as other details such as how frequently they are spawned and how many spawned simulation objects can exist at one time, is configured in a "spawn pattern". VR-Forces comes with two spawn patterns to get you started. You can edit spawn patterns and create as many new ones as you wish. Spawn points and sink points for new and edited spawn patterns are dynamically added to the Create menu and Tactical Graphics Palette so that it is easy to create new spawn and sink points for all of the available patterns.

The SubwayPOL example scenario has spawn patterns that demonstrate continuous simulation object creation, burst traffic, and use of a custom plan.

You can find links to videos that demonstrate spawn patterns at https://www.mak.com/support/tutorials, or click the camera icon next to the procedure whose video you want to view.



Spawn and sink points are automatically added to the Create menu. However, they are not flagged as Favorites on the Tactical Graphics Palette. If you flag a spawn or sink point as a Favorite, a duplicate entry is added to the Create menu.

18.2. Creating Spawn Points

You can create spawn points from the Create menu, the Tactical Graphics Palette, or from the Spawn Pattern Manager. Figure 18-1 illustrates a spawn point.



Figure 18-1. Spawn point

If the plan for a spawn point specifies Use Roads and a spawn point is placed too far away from the road network to use roads, a message is sent to the spawn point's console. If warning icons are enabled, a warning icon is displayed. For details about the Use Roads option, please see "Creating a Spawn Pattern," on page 18-11. For details about configuring console messages and warning icons, please see "Configuring Object Console Messages," on page 16-23.

To create a spawn point from the Create menu:

1. Choose **Create** → **Spawn Points** → *type*, where *type* is one of the spawn patterns. The cursor changes to the waypoint symbol.
2. Click on the terrain where you want to place the spawn point. You can continue clicking to create additional spawn points.
3. Right-click to exit create mode.

To create a spawn point from the Spawn Pattern Manager:

1. Choose **Simulation** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 18-2).

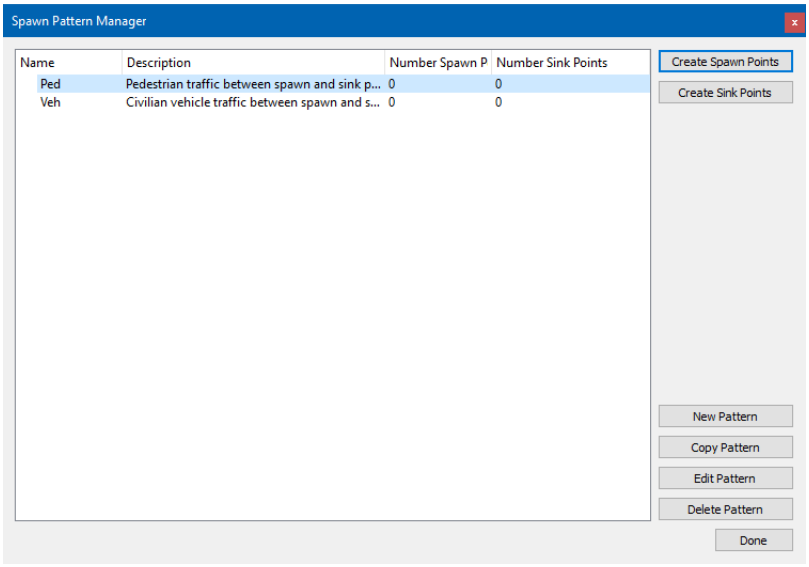


Figure 18-2. Spawn Pattern Manager

2. Select the spawn pattern for which you want to create spawn points.
3. Click Create Spawn Points. The cursor changes to the waypoint symbol.
4. Click on the terrain where you want to place the spawn point. You can continue clicking to create additional spawn points.
5. Right-click to exit create mode.

18.2.1. Activating and Deactivating Spawn Points

You can activate and deactivate spawn points. To do so, use the Activation set data request. For details, please see "Activation," on page 41-2.

18.3. Creating Sink Points

If you are using the default spawned simulation object behavior – pick a sink point and move to it – then you have to create one or more sink points for the simulation objects to move to. Entities created at a given type of spawn point automatically choose among sink points of the same type. (You can configure spawned simulation objects to move to any kind of point by editing the spawn pattern. For details, please see "Creating a Spawn Pattern," on page 18-11.) You can create sink points from the Create menu, the Tactical Graphics Palette, or from the Spawn Pattern Manager. Figure 18-3 illustrates a sink point.

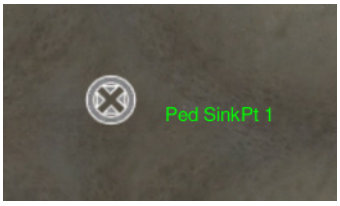


Figure 18-3. Sink point

If a sink point is more than 10 meters from a road and simulation objects move towards it with the Use Roads criteria, they move to the closest point on the road to the sink point and are then removed from the scenario.

To create a sink point from the Create menu:

1. Choose **Create** → **Sink Points** → *type*, where *type* is one of the spawn patterns. The cursor changes to the waypoint symbol.
2. Click on the terrain where you want to place the sink point. You can continue clicking to create additional sink points.
3. Right-click to exit create mode.

To create a sink point from the Spawn Pattern Manager:

1. Choose **Simulation** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 18-2).
2. Select the spawn pattern for which you want to create sink points.
3. Click Create Sink Points. The cursor changes to the waypoint symbol.
4. Click on the terrain where you want to place the sink point. You can continue clicking to create additional sink points.
5. Right-click to exit create mode.

18.4. Spawn Patterns

A spawn pattern configures the following aspects of spawned simulation object creation and behavior:

- The name and description of the spawn pattern.
- The timing of simulation object creation. Entities can be spawned continuously or in bursts.
- The type of simulation object created.
- The spawned simulation object's plan. Entities can follow a default plan or a custom plan.

A spawn pattern's name identifies it on the Create menu and is used to build the label for individual spawn points, for example Ped SpawnPt 1, Ped SpawnPt 2, and so on.

A scenario can have multiple spawn patterns and each spawn pattern can spawn one or more object types. Spawning multiple simulation object types in one pattern is useful when you want heterogeneous traffic. For example, in a stream of pedestrians you might want to include men, women, and children. In other cases you might want a specific simulation object type engaging in specific behaviors and would only spawn one simulation object type in that pattern.

If you use the default plan, you can configure simulation object movement, as follows:

- Maximum and minimum speed. VR-Forces will randomly assign a speed between the maximum and minimum settings. It will not accept a speed that is greater than the maximum speed for the simulation object in the OPD file.
- Use Roads. Enables or disables road-following behavior for vehicles. Road following behavior is described in "Collision, Obstacle, and Feature Avoidance," on page 13-11 and "Entity Movement On Roads and Off Roads," on page 25-21.

When you use the default plan, the Create Spawn Pattern dialog box lists the sink points that this pattern will use.

The SubwayPOL example scenario has spawn patterns that demonstrate spawning simulation objects continuously and in bursts. It also demonstrates use of a custom plan. The following sections describe these different ways to set up spawn patterns.

18.4.1. How Spawn Events are Scheduled

Spawn patterns control the time period during which simulation objects are spawned and the rate at which they are spawned during that time period. A spawn pattern can use one of the following options:

- Continuously. Entities are spawned as soon as the scenario starts and throughout its life.
- Time of Day. Entities are spawned between a starting time of day and ending time of day.
- Simulation Time. Entities are spawned between a specific starting and ending simulation time.

Figure 18-4 illustrates the difference between continuous and timed spawning.

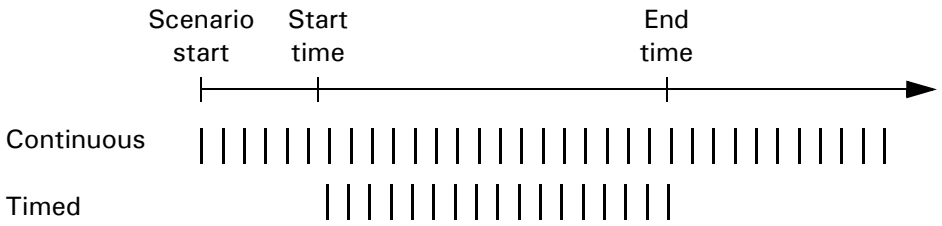


Figure 18-4. Continuous spawning compared to time-based

After you specify a time option, you must specify a spawn interval and a spawn interval variance. The spawn interval determines how frequently simulation objects are spawned. The spawn interval variance introduces some variability to the interval so that there is a more random and naturalistic flow of simulation object creation.

For example, if the spawn interval is 10 seconds, a simulation object is spawned every 10 seconds, plus or minus the variance. The variance is a percentage of the spawn interval. If the variance is 10%, then given a spawn interval of 10 seconds, an simulation object is spawned every 9 to 11 seconds. The spawn intervals are not fixed to an absolute time scale. A new spawn interval starts each time a spawn event occurs.

Figure 18-5 illustrates a series of spawn events with a spawn interval of 10 seconds and a spawn interval variance of 0% and a series of hypothetical events with a spawn interval of 10 seconds and a spawn interval variance of 40%. In the second case, each spawn event can take place anywhere between 6 and 14 seconds after the previous event. Notice that a new spawn interval starts at each spawn event.

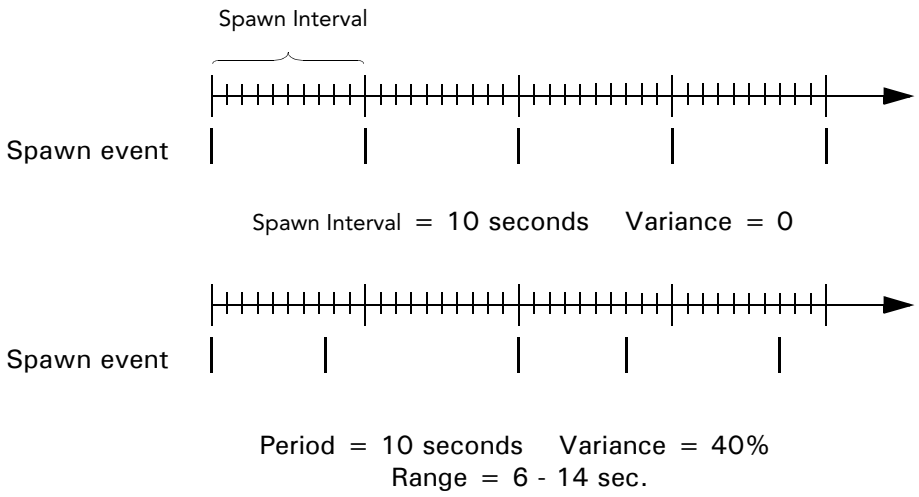


Figure 18-5. Spawn period and variance

The frequency with which a spawn point spawns simulation objects can be affected by the total number of simulation objects that it has spawned. If the maximum number of simultaneous simulation objects for a spawn point is reached, no more simulation objects are spawned from that point until simulation objects from that spawn point are removed from the scenario.

18.4.2. Spawning Entities in Bursts



You can introduce additional variation to the spawn frequency by specifying that simulation objects be spawned in bursts. A burst pattern controls spawning behavior within a fixed period of time. For example, you could use a burst pattern to represent a cycle of train disembarkations or rush hour traffic.

Burst patterns have the following parameters:

- Burst Period. The period of time during which a burst can take place.
- Burst Length. The percentage of the burst period in which to spawn simulation objects.
- Offset Start Time. An offset from the start of the burst period at which to start spawning simulation objects.

Within a burst, simulation objects are created at the rate determined by the spawn interval and spawn interval variance values.

Figure 18-6 illustrates a series of bursts. The burst period is 10 minutes. The offset start time is two minutes, so each burst starts two minutes after the start of the burst period. The burst lasts for 40% of the length of the burst period, in this case four minutes. Within each burst, the frequency of spawn events is controlled by the spawn interval and variance values.

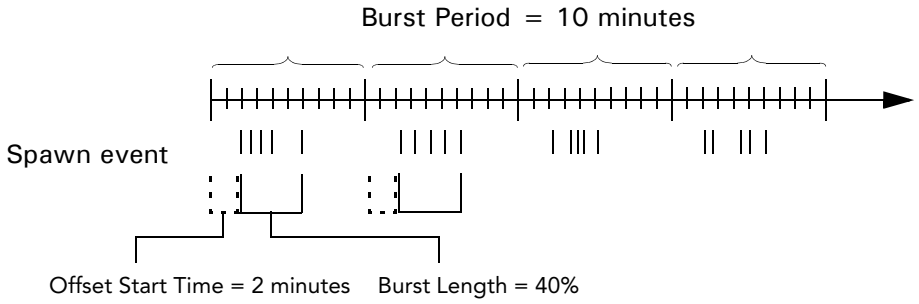


Figure 18-6. Burst periods

18.4.3. Using a Custom Plan



As noted in "Generating Traffic," on page 18-2, the default behavior for a spawned simulation object is to choose a sink point, move to the sink point, and be removed from the scenario. However, you can introduce more complex behaviors for spawned simulation objects by writing a custom plan for a spawn pattern. By creating a custom plan, the simulation objects can engage in a complex set of behaviors, but you still have the benefit of automatic generation. An example of using a custom plan would be to generate a steady stream of jets taxiing to a runway and taking off from an airport.

Custom plans can include all of the tasks, sets, conditions, and global commands that a regular individual plan has. The plan is treated as an individual plan and applied to each simulation object as it is created. When you write a custom plan, simulation objects do not move to sink points. They remain in the scenario unless their plan removes them. You can delete simulation objects without needing to specify the specific simulation object with the Delete Self global command.

18.4.4. Default Spawn Patterns and Scenario-Specific Patterns

The VR-Forces includes default spawn patterns for civilian vehicles and civilian lifeforms. When you create a scenario, these patterns are added to the scenario. When you save a scenario, the spawn patterns get saved with the scenario in the scenario extras file (*.xtr*). (This includes the default spawn patterns if you do not edit them, the edited versions if you have edited them, and any new spawn patterns you have created.) They are now scenario-specific. The next time you load the scenario, it uses the spawn patterns in the scenario extras file, not the default spawn patterns.

If you want to have additional default spawn patterns for your scenarios, you can add entries to *./appData/settings/vrfSim/defaultSpawnTemplates.mtl*. Copy one of the patterns in the file, then change the name and parameter values to suit your needs. Or, create a spawn pattern in a scenario and copy it from the *.xtr* file to *defaultSpawnTemplates.mtl*.

18.5. Creating a Spawn Pattern

To create a spawn pattern:

1. Choose **Simulation** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 18-2).
2. Click New Pattern. The Create Spawn Pattern dialog box opens (Figure 18-7).

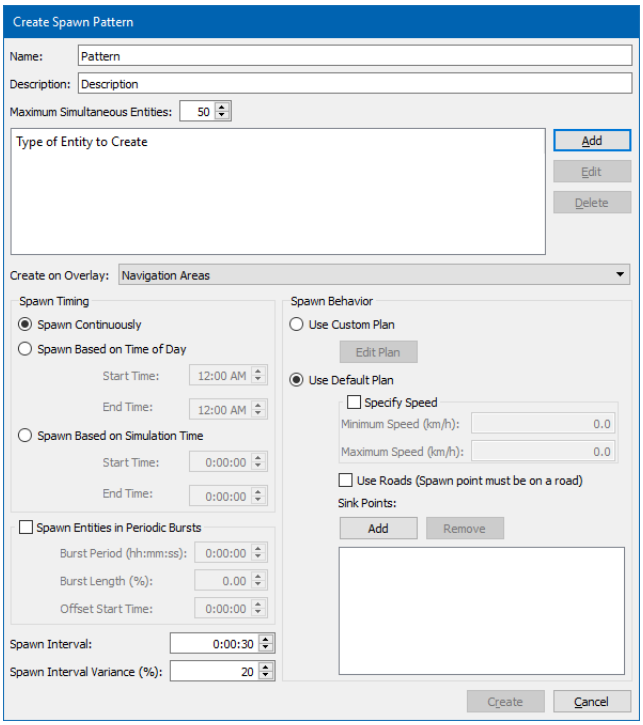


Figure 18-7. Create Spawn Pattern dialog box

3. In the Name box, type a name for the spawn pattern. This name is used for the label for spawn points and sink points created from this pattern. This is a required field.
4. In the Description box, type a description of the pattern. This description is listed in the Spawn Pattern Manager to help identify the purpose of the pattern. This is a required field.

5. Specify the maximum number of simulation objects that can be spawned from a spawn point using this pattern and exist in the scenario at the same time. When the maximum is reached for a given spawn point, no more simulation objects are spawned from that point until a simulation object that was spawned from that point is removed from the scenario.

   For example, if the maximum number of simultaneous simulation objects for a spawn pattern is 200 and you have 10 spawn points of that type, you could have as many as 2000 spawned simulation objects. None of the spawn points will spawn more than 200 simulation objects at one time.

A high maximum number of simultaneous simulation objects along with a small spawn interval can quickly generate tens to hundreds of simulation objects. This may affect performance.

6. Specify the simulation object types to create:
   a. Click the Add button. The Entity Type dialog box opens.
   b. Optionally, filter the simulation object list. (For information about how to filter an objects list using tags, please see "Filtering Object Creation Palettes," on page 14-8.)
   c. Select the simulation object type that you want to add. The simulation object is added to the Type of Entity to Create list.
   d. Add additional simulation object types as desired.



The Entity Type dialog box lets you specify all types of simulation objects, cultural features, and control objects, some of which do not necessarily make any sense to spawn. It is up to you to select types of simulation objects that make sense.

7. In the Create on Overlay list, select the overlay on which you want spawned simulation objects to be created.
8. Choose an option in the Spawn Timing group box, as follows:
   – Spawn Continuously. Entities are spawned throughout the simulation.
   – Spawn Based on Time of Day. Specify the starting time of day and ending time of day during which you want simulation objects to be spawned.
   – Spawn Based on Simulation Time. Specify the starting simulation time and ending simulation time during which you want simulation objects to be spawned.

9. Optionally, select Spawn Entities in Periodic Bursts. (For an explanation of periodic bursts, please see "Spawning Entities in Bursts," on page 18-9.) If selected, configure as follows:
   - Burst Period. The time period for each burst.
   - Burst Length. The length of time, as a percentage of the burst period, during which simulation objects are spawned.
   - Offset Start Time. An offset from the start of the burst period to start spawning simulation objects.
10. Specify the spawn interval. This specifies the amount of time between each spawn event, modified by the spawn interval variance.
11. Specify the spawn interval variance as a percentage of the spawn interval.
12. In the Spawn Behavior group box, select a behavior option as follows:
    - Use Custom Plan:
      a. Click Edit Plan. The Edit Custom Plan dialog box opens.

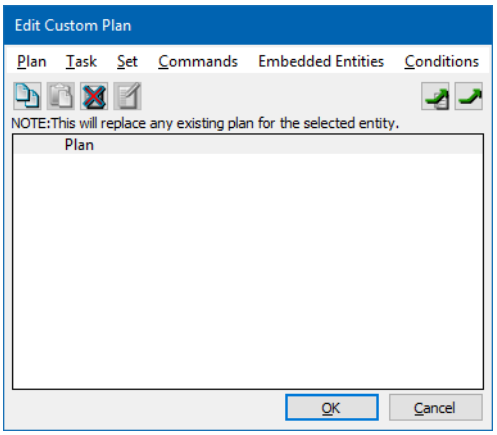


Figure 18-8. Edit Custom Plan dialog box

      b. Write a plan that each simulation object created from this spawn pattern will use. Unless you want the simulation objects to remain in the scenario permanently, end the plan with a Delete Self command. (For details about how to write plans, please see Chapter 37, *Writing Plans*.)
    - Use Default Plan:
      a. Optionally, select the Specify Speed check box and specify the minimum and maximum speeds for spawned simulation objects. If you do not select this option, spawned simulation objects move at the default ordered speed in their OPD file. If you select it, they move at randomly chosen speeds between the minimum and maximum values. This option introduces some variance in behavior. Although the GUI allows you to set any maximum speed, simulation objects will not move faster than the maximum speed in the OPD file.

b. To have a simulation object move on roads, select the Use Roads check box. If this option is cleared, simulation objects move directly towards sink points without regard to vector road networks.

! Use Roads is valid only for ground vehicles. Do not select this option for any other simulation object type.

If Use Roads is selected, spawn points must be within 10 meters of the road. If a spawn point is farther away, the simulation objects are removed from the scenario as soon as they are created.

If a sink point is more than 10 meters from a road, simulation objects that are moving towards that sink point move to the point on the road that is closest to it and are then removed from the scenario.

---

c. Optionally, use the Add button to edit the Sink Points list. A new spawn pattern will not have any sink points listed because until you create the spawn pattern, you cannot create any sink points for that pattern. Entities will automatically use sink points that match the spawn pattern, so after you create them, you do not have to add them to the list. However, they can also move to waypoints. If you want spawned simulation objects to move to waypoints, you must add them to the list.

13. Click Create.

18.5.1. Copying a Spawn Pattern

If you want to create a new spawn pattern that will be similar to an existing spawn pattern, you can copy the existing spawn pattern to a new name.

To copy a spawn pattern:

1. Choose **Simulation** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 18-2).
2. Select the spawn pattern that you want to copy.
3. Click Copy Pattern. The Create Spawn Pattern dialog box opens. It has a default name based on the original pattern name.
4. Edit the spawn pattern. The meaning of each option is described in "Creating a Spawn Pattern," on page 18-11.
5. Click Create.

18.6. Editing a Spawn Pattern

You can change any of the parameters of a spawn pattern. The changes do not affect any simulation object that has already been spawned.

To edit a spawn pattern:

1. Choose **Simulation** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 18-2).
2. Select the pattern that you want to edit.
3. Click Edit Pattern. The Edit Spawn Pattern dialog box opens (Figure 18-9).

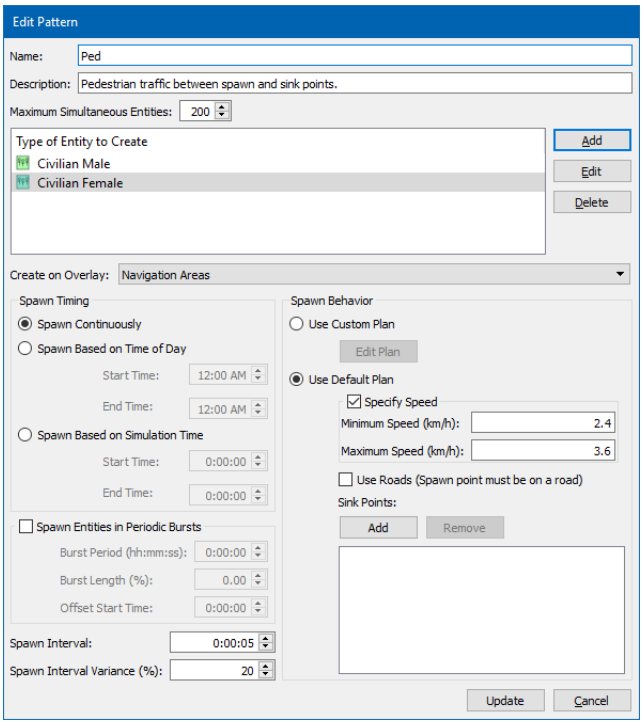


Figure 18-9. Edit Spawn Pattern dialog box

4. Change any value as desired. The meaning of each option is described in "Creating a Spawn Pattern," on page 18-11.
5. Click Update.

18.7. Deleting a Spawn Pattern

If you delete a spawn pattern, all spawn points of that type get deleted. Sink points do not get deleted. The spawn and sink points of this type are removed from the Create menu and Tactical Graphics Palette. If the scenario is running, any simulation objects that were spawned from these points continue executing their plans.

To delete a spawn pattern:

1. Choose **Simulation** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 18-2).
2. Select the spawn pattern that you want to delete.
3. Click Delete Pattern. You are prompted to confirm the deletion.
4. Click Yes.



19. Crowds and Multiple Simulation Objects

This chapter explains how to create many simulation objects at one time and how to create and task pedestrian crowds.

Creating Multiple Simulation Objects and Crowds 19-2
Creating Crowds Using a Pedestrian Area 19-2
Adding Entities to a Pedestrian Area 19-3
Creating Crowd Units 19-3
Crowd Tasks 19-4
Creating Multiple Simulation Objects 19-5
Adding an Area for Multiple Simulation Object Creation 19-8
Specifying the Simulation Objects to Create 19-9
Pedestrian-Oriented Tactical Graphics 19-10
Using Civilian Visit Points 19-10

19.1. Creating Multiple Simulation Objects and Crowds

VR-Forces can quickly create multiple simulation objects within an area. This allows you to create crowds, traffic, or other multi-object groups without having to place each simulation object by hand. You can use any of the following methods to create multiple objects:

- Pedestrian area. A pedestrian area gets populated by a random number of male and female civilians. They execute a wander task.
- Multiple simulation objects. You can specify the number of simulation objects to create, the type of objects to create, and you can assign tasks and set data requests to them.
- Crowd unit. Create a crowd unit from the Simulation Objects Palette or aggregate individual humans into a crowd unit. Assign crowd-specific tasks to the crowd.

19.2. Creating Crowds Using a Pedestrian Area

When you create a pedestrian area, VR-Forces populates it with a random number of male and female civilian entities. They are created as a ground unit. They are all assigned a wander task. You cannot specify the number of entities to create or the mix of entities to create.

! If you delete a pedestrian area, all of the entities that were created for it are also deleted.

You can create pedestrian areas only in navigation areas.

To create a crowd using a pedestrian area:

1. Open the Tactical Graphics Palette.
2. Optionally, filter the list by selecting the Simulation Effect Object button ().
3. Select Pedestrian Area.
4. Draw an area on the terrain as you would any other areal tactical graphic. When the area is completed, VR-Forces creates entities to populate the area.

19.2.1. Adding Entities to a Pedestrian Area

After you create a pedestrian area, you can automatically add additional entities to the area.

To automatically add entities to a pedestrian area:

1. Select the area.
2. Choose **Task** → **Create Pedestrians**. The Create Pedestrians dialog box opens.
3. Specify the number of pedestrians to create.
4. Select a placement option.
5. Select or clear the Restrict To Area check box.
6. Click OK. The pedestrians get added to the area.

19.3. Creating Crowd Units

VR-Forces has three preconfigured crowd units: small, medium, and large. They create a unit of civilian males and females. You can also create crowd units by aggregating individual human entities. However, some of the animations used in crowd tasks are specific to the civilian characters used in the supplied units.

When you place a crowd, if the terrain has pedestrian paths or other pedestrian-specific locations, the members of the crowd are placed in these locations.

Crowd members get created within a configurable radius of the location you specify, with a configurable density. By default, crowds are created using the parameters in Table 19-1. You can change the values in the Simulation Object Editor.

Table 19-1: Crowd units

Unit	Radius (m)	Density	Approximate Size*
Civilian Crowd (Small)	15	2	50
Civilian Crowd (Medium)	30	2	100
Civilian Crowd (Large)	50	2	175

*If there are not enough valid locations to place entities, the size of the unit may be smaller than the theoretical maximum.

Crowd units are supported only in entity-level scenarios.

You can create preconfigured crowds only in navigation areas and crowd tasks can be executed only in navigation areas. Although you can aggregate individual entities into a crowd outside of a navigation area, they will not be able to execute any crowd tasks.

To create a crowd:

1. Click the Simulation Objects Palette.
2. Clear all filters (click the All top level tag).
3. Optionally, click the Civilian tag. Then click the Crowd tag.
4. Select the Civilian Crowd (Small), Civilian Crowd (Medium), or Civilian Crowd (Large) unit.
5. Click on the terrain to place the unit.

19.3.1. Crowd Tasks

Crowds can execute some of the tasks available to other ground units. They have the following movement tasks:

- Crowd around a location.
- Crowd around an object.
- Gather in front of an object.
- Flee.
- Wander.
- Protest.
- Disperse Crowd.



When a crowd gathers around a location or object, VR-Forces calculates a target destination for each member of the crowd and each member of the crowd tries to move to that destination. If you send two or more crowds to the same location or object, these target destinations may overlap. Individual entities may give up on their movement tasks or may end up very close to members of the other crowd.

For details about each task, please see its section in Chapter 28, *Human and Crowd Tasks*.

19.4. Creating Multiple Simulation Objects

You can add multiple simulation objects to a scenario in one operation and assign them a common task. The simulation objects are added to the scenario as members of a selection group. (For information about selection groups, please see "Using Selection Groups," on page 15-8.) Creation of multiple simulation objects has the following constraints:

- The simulation objects are added within a specified area. If an appropriate area does not already exist, you can create one as part of the simulation object creation process.
- If the area in which you want to place simulation objects overlaps navigation data, the simulation objects are placed only in the portion of the area that has navigation data, as illustrated in Figure 19-1.
- If the area you choose for placing simulation objects is too small to contain the number of simulation objects that you specify, some simulation objects might not be created.

When you create multiple simulation objects, VR-Forces automatically puts them in a selection group named group_*number*, where *number* starts at 0 and is incremented for each group created.

In Figure 19-1, Area 1 overlaps the navigation data. If, for example, you try to place 25 entities in this area, they are placed only in the portion of Area 1 that overlaps navigation data, and only the number of simulation objects that fit in this subset of Area 1 are created.



- If VR-Forces cannot create the number of simulation objects specified (due to one of the reasons mentioned in this section), there is no notification. It is your responsibility to check to see how many simulation objects were created.
- Creating multiple simulation objects as described in this section is not the same thing as pattern of life traffic generation or creating simulation object groups. It is just a way to create a lot of simulation objects at one time and give them a task. And unlike creating crowds, which only contain civilians, you can create any type of simulation object.

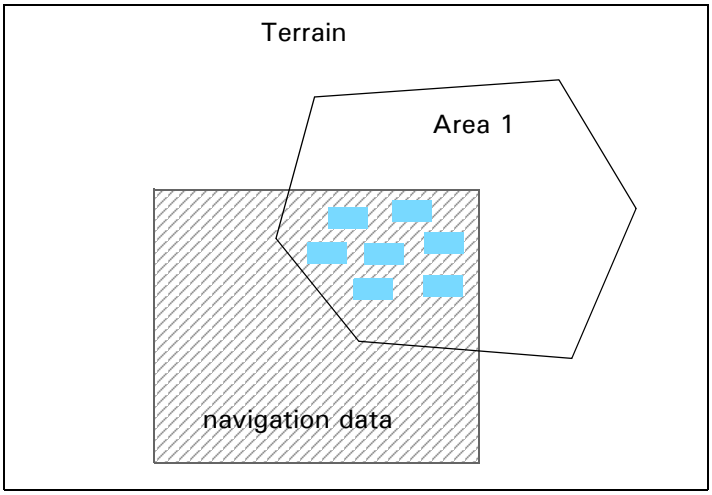


Figure 19-1. Placement of multiple simulation objects

When you create multiple simulation objects you can ask VR-Forces to run an accessibility check before it creates each simulation object. VR-Forces tries to ensure that the simulation object being created can reach a specified waypoint from a candidate location in the specified area. This check is primarily designed to avoid putting simulation objects on top of buildings that are unreachable and would not allow the simulation object to leave the building. It works as follows:

- If the terrain does not have navigation data, the accessibility check is disregarded and the simulation objects get created.
- If the terrain has navigation data and if the candidate location cannot use navigation data to reach the waypoint, VR-Forces does not create the simulation object at that location. It tries to find another location at which to place the simulation object. If after multiple attempts to place a simulation object the accessibility check still fails, VR-Forces proceeds as follows:
  - If this is the first simulation object that VR-Forces has tried to create, the accessibility check is disregarded and VR-Forces creates the simulation objects (within the constraints previously mentioned).
  - If the accessibility check fails for any simulation object after the first simulation object was created, VR-Forces assumes there is no more space in which to create simulation objects and stops trying to create them.

To add multiple simulation objects to a scenario:

1. Choose **Create** → **Multiple Simulation Objects**. The Create Multiple Simulation Objects dialog box opens (Figure 19-2).

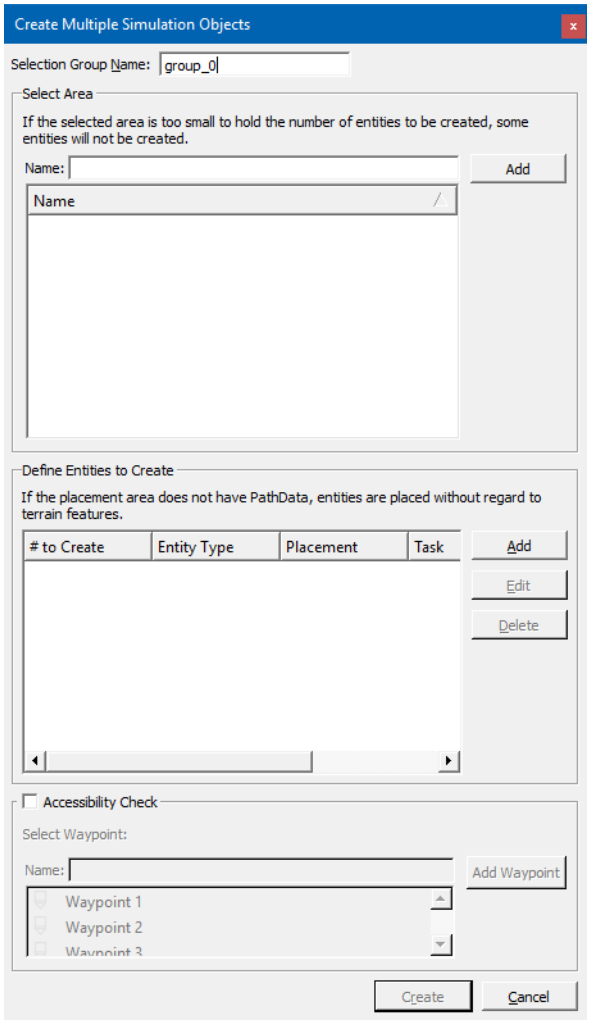


Figure 19-2. Create Multiple Simulation Objects dialog box

2. Optionally, change the name of the selection group.
3. In the Select Area group box, type the name of the area in which you want to create the simulation objects, or select the area from the list or on the map. If the area in which you want to create the simulation objects does not exist, add a new area as described in "Adding an Area for Multiple Simulation Object Creation," on page 19-8.
4. In the Define Entities To Create group box, add the simulation objects that you want to create. For details about this part of the procedure, please see "Specifying the Simulation Objects to Create," on page 19-9.

5. Optionally, edit or remove any of the entity sets listed in the Define Entities To Create list.
6. Optionally, select the Accessibility Check check box. If you select this check box, select a waypoint. You can create a waypoint from this dialog box if necessary.
7. Click Create. The entities are created in the specified area.

19.4.1. Adding an Area for Multiple Simulation Object Creation

To add an area from the Create Multiple Entities dialog box:

1. In the Select Area group box, click Add. A Create Area tab is added to the window.
2. Create an area following the standard VR-Forces procedure for creating areas. For details, please see "Placing Objects," on page 14-13.

19.4.2. Specifying the Simulation Objects to Create

When you add multiple simulation objects to a scenario, you can specify any of the supported object types and assign a common task or set data request to a specific set of simulation objects.

To specify the simulation objects to add in the Create Multiple Entities dialog box:

1. In the Define Entities to Create group box, click Add. The Add Group of Entities dialog box opens (Figure 19-3).

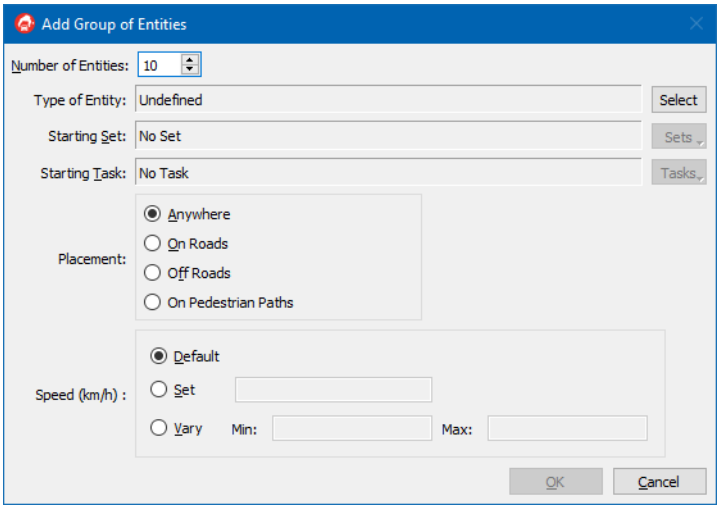


Figure 19-3. Add Group of Entities dialog box

2. In the Number of Entities box, specify the number of simulation objects you want to add for this set.
3. Click the Select button that is next to the Type of Entity text box. The Entity Type dialog box opens. The list of object types available is the same list of simulation objects provided on the Simulation Objects Palette.
4. Select the object type you want to add. The dialog box closes and the object type is listed in the Type of Entity box.
5. Optionally, specify a starting set data request by clicking the Sets button and selecting a set data request. The list is not context sensitive. You must ensure that the selected set data request is appropriate to the object type that you are adding.
6. Optionally, specify a starting task by clicking the Tasks button and selecting a task. The list is not context sensitive. You must ensure that the selected task is appropriate to the object type that you are adding.
7. Optionally, select an option in the Placement group box. VR-Forces will try to place the simulation objects in the desired location.
8. Optionally, set a speed and variance range.
9. Click OK. The object set is added to the list in the Define Entities To Create group box (Figure 19-2).

10. Optionally, add additional object sets.

19.5. Pedestrian-Oriented Tactical Graphics

VR-Forces has the following tactical graphics that are particular to pedestrians. These objects only work in navigation areas.

- Pedestrian path. When humans plan paths in navigation areas that have navigation data, if a pedestrian path is present that can take them to their destination, they will use it. You can think of a pedestrian path as a route that a human automatically chooses to follow instead of being told to follow it. Pedestrian paths are supported only for entity-level scenarios.
- Civilian visit points. Waypoints that attract the attention of civilians.
- Crosswalks. Crosswalks are similar to pedestrian paths. Pedestrians prefer them when path planning. The unique aspect of crosswalks is that if an entity is configured with a human crosswalk controller, it checks to see if it is safe to cross the street before proceeding. (Crosswalks are an experimental feature. Please contact support@mak.com if they do not work as you expect.)
- Pedestrian areas. Pedestrian areas automatically generate pedestrians when you create them. For details, please see "Creating Crowds Using a Pedestrian Area," on page 19-2.

19.5.1. Using Civilian Visit Points

Civilian visit points are waypoints that attract the attention of civilian entities that are walking nearby. The civilians can be members of a crowd or individual entities. When a civilian comes within a configurable distance of a civilian visit point, it stops and faces the point a configurable percentage of the time. If it stops, it waits a few seconds and then continues its task.

This behavior is controlled by the Visit Interest Points reactive task. You can edit the parameters that control an entity's behavior in the Manage Reactive Tasks dialog box. For more information about reactive tasks, please see "Reactive Tasks," on page 25-11.



20. Creating and Using an Order of Battle

An Order of Battle is a description of the simulation objects and simulation object hierarchy that you can create in a scenario.

Introduction to the Order of Battle 20-2
The Order of Battle Panel 20-3
Creating an Order of Battle 20-3
Adding Objects Directly to the Order of Battle 20-4
Adding Existing Objects to the Order of Battle 20-5
Creating Units in the Order of Battle Panel 20-5
Instantiating Members of an Order of Battle in a Scenario 20-6
The Order of Battle Palette 20-6
Removing Objects from an Order of Battle 20-6
Editing the Objects in an Order of Battle 20-7
Changing the Order of Objects in an Order of Battle 20-7
Exporting an Order of Battle 20-7
Importing an Order of Battle 20-8

20.1. Introduction to the Order of Battle

In the abstract, an Order of Battle (OOB) describes the hierarchical structure of the personnel and resources that are available to a military commander. The term can also be used to refer to the hierarchical structure of the personnel and resources deployed in a particular engagement.

VR-Forces implicitly supports the second sense of the term by listing all simulation objects in a scenario in the Objects List Panel. It also supports the abstract sense of the term by allowing you to construct an Order of Battle that represents a hierarchy of simulation objects that can be instantiated (created) in a scenario.

You create an Order of Battle in the context of a scenario. However, once you create an OOB, you can export it and then import it into other scenarios. This lets you quickly create new scenarios that use the same OOB for training or scenario development. You can configure VR-Forces so that users do not have access to the Simulation Objects Palette and can only add simulation objects to a scenario if they are in the OOB.

An Order of Battle can include any simulation object that you can create from the Simulation Objects Palette. However, unlike the Simulation Objects Palette in which each object is essentially a template for creating multiple instances of that object type, each member of an Order of Battle has its own unique ID (UUID). When you instantiate a member of the OOB, you only create that one instance of the object type.

In VR-Forces you can:

- Create an Order of Battle, which involves:
  - Adding members.
  - Removing members.
  - Changing the name or object type of members.
  - Changing the order of members.
  - Organizing members into units.
- Instantiate members of the OOB in a scenario.
- Import and export OOBs.

When you create an OOB, it gets saved with the scenario.

20.1.1. The Order of Battle Panel

You can view and manage the Order of Battle on the Order of Battle Panel (Figure 20-1). The Order of Battle Panel lists the members of the OOB, by force. Each object in the OOB list indicates its status in the scenario as follows:

- Gray text. A member of the OOB that is not instantiated in the scenario.
- Black text. A member of the OOB that is instantiated in the scenario.

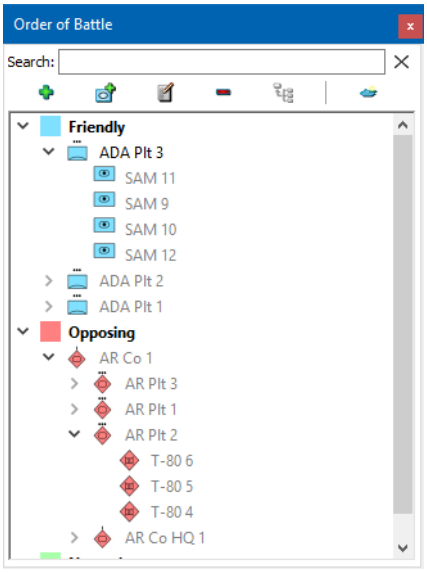


Figure 20-1. Order of Battle Panel

➤ To open the Order of Battle Panel, choose **View** → **Order of Battle Panel**.

20.2. Creating an Order of Battle

You create an Order of Battle by adding members. You can add any simulation object that is available on the Simulation Objects Palette.

You can add objects to the OOB in either of the following ways:

- Add objects to the Order of Battle without adding them to the scenario.
- Add objects that exist in the scenario, but are not part of the OOB, to the OOB.

20.2.1. Adding Objects Directly to the Order of Battle

You can add a simulation object to the Order of Battle, without creating it in the scenario.

When using this method, if you want to have more than one instance of an object in the OOB, you must add a discrete member for each one. For example, if you want to have 10 HMMWVs in the OOB, you must add 10 HMMWVs, one at a time.

If you want to add many instances of the same simulation object to the Order of Battle, it may be quicker to create them in the scenario from the Simulation Objects Palette and then add them to the OOB.

To add simulation objects directly to an Order of Battle:

1. Open the Order of Battle Panel.
2. Click the Add button (+). The Add to Order of Battle dialog box opens (Figure 20-2). It is similar to the Simulation Objects Palette, except that you can also specify a name for the object.

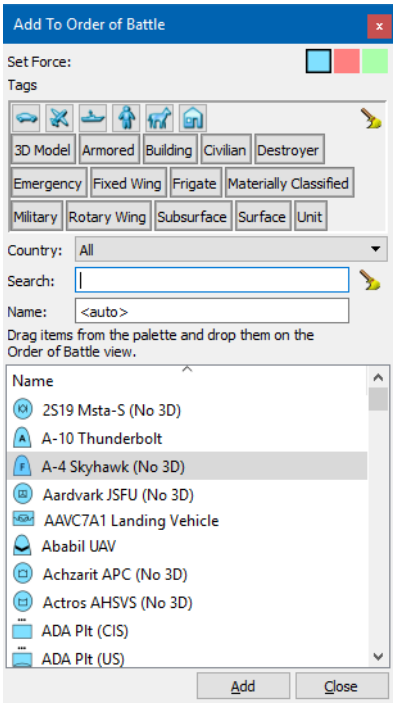


Figure 20-2. Add to Order of Battle dialog box

3. On the Set Force line, select the force to which you want to add this object.
4. Optionally, filter the list.
5. In the list of simulation objects, select the object you want to add to the OOB.
6. Optionally, specify a name for the object you want to add to the OOB.

7. In the Order of Battle Panel, select the force or unit under which you want to add the object. The following rules apply:
   - If you add to a force, it becomes a top level subordinate of the force.
   - If you add to a unit, it becomes a subordinate of that unit.
   - If you add it to an entity, it becomes a sibling of the entity. If the entity is part of a unit, the new member becomes part of the unit.
8. In the Add to Order of Battle dialog box, click Add.



You can quickly add objects to the OOB by dragging them from the Add to Order of Battle dialog box to the Order of Battle Panel. The same rules for adding members apply. Also, regardless of the force selected on the Set Force line, when you drag an object onto the Order of Battle Panel, it becomes part of the force you drag it to.

20.2.2. Adding Existing Objects to the Order of Battle

You can create objects from the Simulation Objects Palette and then add them to the OOB. You can also use this method to add objects from existing scenarios to an OOB. You can add objects individually and you can add all objects at once.

When you add an existing object to the OOB, it gets added to the same force and its place in the force hierarchy is preserved. If you add a member of a unit, the entire unit gets added to the OOB.

To add an existing object to the OOB:

1. Select the objects that you want to add to the OOB.
2. Choose **Simulation** → **Order of Battle** → **Add to Order of Battle**.

➤ To add all objects in the scenario to the OOB, choose **Simulation** → **Order of Battle** → **Add All Simulation Objects to Order of Battle**.

20.3. Creating Units in the Order of Battle Panel

You can aggregate objects listed in the Order of Battle Panel into units. The process and criteria for creating units is the same as described in "Creating a Unit by Combining Existing Simulation Objects," on page 23-3.

To aggregate members of an Order of Battle into units:

1. Select the objects that you want to aggregate.
2. Choose **Objects** → **Unit** → **Aggregate As**, or click the Aggregate As button () on the Order of Battle Panel. The Aggregate As dialog box opens.
3. Select the unit type you want from the list.
4. Optionally, change the order of the unit members.
5. Click OK.

20.4. Instantiating Members of an Order of Battle in a Scenario

If an Order of Battle includes simulation objects that have not been instantiated (created) in the scenario, you can add them to a scenario. Since each member of the OOB represents a specific simulation object with its own unique ID (UUID), when you add a simulation object from the OOB, you only add that specific object. You cannot add multiple instances of the object type as you would from the Simulation Objects Palette.

To add a simulation object to a scenario from the Order of Battle:

1. In the Order of Battle Panel, select the simulation object you want to add to the scenario. (You can only add objects that are not already in the scenario. That is, objects listed in gray text.)
2. Click the Add Object button ( ).
3. Click on the terrain where you want to place the object. The entry for the object in the Order of Battle Panel changes from gray to black.

20.4.1. The Order of Battle Palette

There may be cases where you want to set up a training exercise in which participants can only add simulation objects from the Order of Battle. VR-Forces has an Order of Battle Palette that contains the Order of Battle as listed in the Order of Battle Panel, but without any of the controls for adding, removing, or editing the OOB. This palette is hidden by default, but you can enable it from the View menu.

Using the window layout feature you could save a layout in which the Simulation Objects Palette is hidden and the Order of Battle Palette is visible. Taking things a step further, you could edit the GUI configuration file so that the Simulation Objects Palette and the Order of Battle Panel are not even available in the GUI. For information about saving window layouts, please see "Saving Window Layouts," on page 83-3. For information about configuring the menu options in the GUI, please see Chapter 84, *Configuring Toolbars, Dialog Boxes, and Menus*.

20.5. Removing Objects from an Order of Battle

You can remove objects from the OOB. When you remove an object from the OOB, if it is instantiated in the scenario, it stays in the scenario. If you delete an object from a scenario, if it is a member of the OOB, it is not removed from the OOB. The object's entry in the Order of Battle Panel changes from black to gray.

If you remove a unit, all its subordinates get removed. If you remove a member of a unit, the rest of the unit is not affected.

To remove an object from the Order of Battle:

1. Select the object in the Order of Battle Panel or on the map.
2. Click the Delete button ( ) on the Order of Battle Panel or press the **`Delete`** key.

20.6. Editing the Objects in an Order of Battle

You can edit the members of an Order of Battle as follows:

- If an object is created in the scenario, editing it opens up the standard Edit dialog box for a simulation object. For details, please see "Editing Simulation Objects," on page 14-25.
- If an object is not created in the scenario, you can change its name, object type, and force.

To edit a member of the Order of Battle:

1. Select an object in the Order of Battle Panel.
2. Click the Edit Object button ( ).

   If the object has been created, a standard Edit dialog box opens.

   a. Make changes as desired.

   b. Click Update.

   If the object has not been created, the Edit Name / Object Type dialog box opens.

   a. To change the name, type a different name in the Name box.

   b. To change the object type, select a different object in the list.

   c. Click OK.

   In either case you can change a simulation object's name by clicking the name in the Order of Battle Panel to make it editable.

20.7. Changing the Order of Objects in an Order of Battle

You can reorganize the members of the OOB. This lets you populate the OOB in the most convenient manner and then organize it into the desired hierarchy. You can move members from units into other units, from units to the force level, and from the force level to units. You cannot change the order in which sibling objects or units are listed.

➤ To reorganize an object in the Order of Battle, in the Order of Battle Panel, drag the object to its new location.

20.8. Exporting an Order of Battle

You can export an Order of Battle so that you can import it into other scenarios.

To export an Order of Battle:

1. Choose **Simulation** → **Order of Battle** → **Export Order of Battle**. The Export Order of Battle dialog box opens.
2. Type a name for the Order of Battle.
3. Click Save.

20.9. Importing an Order of Battle

You can import an Order of Battle into a scenario. When you import an OOB, you can merge it with the existing OOB, if any, or you can selectively import members of the OOB.

To import an Order of Battle:

1. Choose **Simulation** → **Order of Battle** → **Import Order of Battle**. The Import Order of Battle dialog box opens.
2. Select the OOB you want to import.
3. Click Open. The Order of Battle Selector dialog box opens.
4. To import all members of the selected OOB, click Import All.

   To import selected members of the OOB, drag them from the Order of Battle Selector to the Order of Battle Panel.
5. If necessary, close the Order of Battle Selector.



IV. Modeling Units

VR-Forces has two, quite different, ways to model simulation objects – entity-level modeling and aggregate-level modeling. These modeling approaches affect how units are modeled.

The chapters in this section provide conceptual background about modeling units, how to create them, and how to display them.



21. Entity-Level Unit Modeling

This chapter describes how units are modeled in entity-level scenarios.

Units in Entity-Level Scenarios 21-2
How Units Are Organized 21-3
How Units Move 21-4
How Units Carry Out Tasks 21-5

21.1. Units in Entity-Level Scenarios

In entity-level scenarios units are simulated in either of two states:

- Disaggregated: Subordinates are simulated and published as individual entities (Figure 21-1). The members of a unit can move individually through the scene, have their own plans, interact with other simulation objects, and do all the things you would expect of an individual entity. When a unit engages in combat, the individual members of the unit fight the opposing forces using their weapon systems and survive or die as individuals.

  Units in this state can be tasked and given plans, but usually pass responsibility for execution to their subordinates. For example, if you give a disaggregated unit a movement task, each member of the unit plots a path and follows it.

- Aggregated: Subordinates are not simulated separately. Tasks and plans are executed by the unit. For example, if you give an aggregated unit a movement task, the object representing the unit moves.

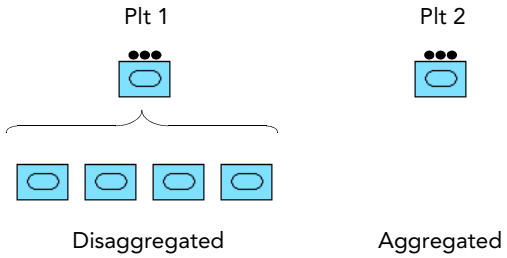


Figure 21-1. Disaggregated and aggregated units

- The disaggregated and aggregated states are not the same as expanded and collapsed units (which are explained in "Expanding and Collapsing Units," on page 24-2.). Expanded and collapsed units are simply different visual representations of the unit. Aggregated and disaggregated states are different simulation states.
- An aggregated unit in an entity-level scenario is similar to a unit in an aggregate-level scenario in that there is no representation of subordinates. However, an aggregated unit can disaggregate and model subordinates, whereas a unit in an aggregate-level scenario cannot do so. Furthermore, their underlying modeling is completely different.
- Although it is theoretically possible for an entity-level scenario to have units that cannot be disaggregated into individual vehicles and persons, *EntityLevel.sms* does not have any units of this type.

A unit can support either of these states. At any given time, one subset of components and systems is enabled, and the other is disabled. At runtime, you can switch dynamically between these states.

You can configure VR-Forces so that unit state is determined automatically according to a set of rules, or to have unit state controlled manually.

The choice of whether or not to simulate units in the aggregated or disaggregated state depends on your goals for the simulation. For example, if you need to support large numbers of simulation objects, and do not want or need to simulate individual entity behavior, then you may want to simulate some or all of your units in the aggregated state. Conversely, if you need to simulate the detailed behavior of subordinates, then you will want to simulate units in the disaggregated state.

A unit in the aggregated state only simulates movement behavior. It does not model combat behavior. If you want a unit to model attrition, simulate it in the disaggregated state. A unit in the aggregated state keeps track of its current resource and attrition state. If a unit later transitions to a disaggregated state, this state information is preserved.

For more details about the differences between aggregated and disaggregated units, please see Chapter 23, *Creating and Controlling Units*.

21.1.1. How Units Are Organized

The organization within a disaggregated unit is based on the echelon ID of each member. For example, in Table 21-1, four unique entities (identified by name) are assigned designators in `2 Plt`.

Table 21-1: Example of names and echelon IDs

Name	Echelon ID
Tank 1	1 M1A2, 2 Plt, 1 Force
Tank 2	2 M1A2, 2 Plt, 1 Force
Tank 3	3 M1A2, 2 Plt, 1 Force
Tank 4	4 M1A2, 2 Plt, 1 Force

Assignment of echelon IDs is handled automatically by VR-Forces when you create a unit.

Reorganizing Units

Reorganization is the reassignment of echelon IDs to the members of a unit when a member gets destroyed. If reorganization is enabled, when a member of a unit gets destroyed, the destroyed member is assigned the highest numerical ID in the unit, and the surviving members' echelon IDs are moved down numerically by one. For example, assume a platoon as shown in the leftmost table in Figure 21-2. If `2 M1A2` (currently the entity named Tank 2) gets destroyed, reorganization would result in the new echelon ID assignments illustrated in the rightmost table. The former `3 M1A2` is now `2 M1A2`, former `4 M1A2` is now `3 M1A2`, and the former `2 M1A2`, is now `4 M1A2`. Their echelon IDs have changed. Their names and object IDs have not.

Echelon ID	Name
1 M1A2	Tank 1
2 M1A2	Tank 2
3 M1A2	Tank 3
4 M1A2	Tank 4

Echelon ID	Name
1 M1A2	Tank 1
2 M1A2	Tank 3
3 M1A2	Tank 4
4 M1A2	Tank 2

Figure 21-2. Reorganization of Platoon 2

You can configure VR-Forces to reorganize units automatically, or you can reorganize manually. Manual reorganization does not require you to reassign each designator; you simply order the unit to reorganize with the Reorganize command.



Reorganization only applies to disaggregated units.

21.1.2. How Units Move

When an aggregated unit receives a movement command, VR-Forces calculates the path it must follow. Then it moves to the location similarly to any individual entity. The unit icon is placed at the center of the bounding box of the area that would be encompassed by the unit if it was disaggregated.

When a disaggregated unit receives a movement command, VR-Forces calculates the path the unit leader must follow for movement. Then it calculates parallel paths (taking into account the formation) for each member of the unit. The unit members are then responsible for following these paths until the unit completes the movement task.

Closing Formation

VR-Forces can close up holes in formations that are created when a unit member is removed from a formation. When a unit is ordered to form up, it assigns formation positions to each of its members. Each position in the formation has an internal ID. If at some point one or more of the unit's members are destroyed, removed, or independently tasked, the unit reassigns the formation IDs for its list of members. Members that are removed, destroyed, or independently tasked, are not given formation IDs. The active members are moved up in the formation to fill the vacancies. Assignment to a different formation position has no effect on echelon designator.

The process of closing formation is internal to a unit. Formation position is not reported as part of simulation object information. The only way to determine a simulation object's formation position is to observe the formation as the unit moves.

Closing formation is an inherent feature of units in the VR-Forces application. You do not need to do anything to put it into effect and you cannot turn it off.

Closing Formation Versus Reorganization

Closing the formation of the entities in a unit is not the same thing as reorganization. Reorganization changes the echelon designators of its members. Closing formation does not change the echelon designators of unit members, or the tasks a simulation object is assigned. It only affects the relative positioning of members within a formation.

21.1.3. How Units Carry Out Tasks

Like other simulation objects, each unit has a plan and you can assign independent tasks to units. When a disaggregated unit is given a task, it sends radio messages to its subordinates directing them to carry out their role in the task, for example, getting into formation and moving to a waypoint.

If a unit is in the aggregated state, it does not send task messages to subordinates. However, if while carrying out its task the unit disaggregates, it sends appropriate task messages to its subordinates to continue with the task execution. For example, suppose a unit is moving to a point and its path to the point passes through a disaggregation area. When it enters the disaggregation area, it would disaggregate, calculate routes for its subordinates, and send them Move Along Route tasks. When the unit exits the disaggregation area, it aggregates again and the tasks for its subordinates are abandoned.



Disaggregation areas are special control objects. If automatic aggregation and disaggregation is enabled, when a unit enters a disaggregation area, it automatically disaggregates. When it leaves the area it aggregates. For more information, please see "Using Disaggregation Areas," on page 23-12.

The precise way in which a unit sends out messages to its subordinates is part of the modeling of the unit that is built into the VR-Forces application.



22. How Aggregate-Level Modeling Works

This chapter describes the aggregate warfare model and how simulation objects created using *AggregateLevel.sms* work.

The Aggregate Warfare Model 22-3
- Unit Combat 22-4
- Unit Footprints 22-5
- Unit Posture 22-6
- Unit Movement 22-8
- Limited Munition Attack 22-9
- Contamination Areas 22-9
- Unit Sensors 22-10

Combat Engineering Objects 22-11
- Concealed Obstacles 22-12
- Creating Combat Engineering Objects 22-13
- How Engineering Units Create Engineering Objects 22-13

Logistics 22-15
- Receiving Supplies 22-17
- Providing Supplies 22-17

Electronic Warfare 22-18
- Jamming Communications 22-18
- Jamming Radar 22-19
- Sensing Electronic Emissions 22-20

Air Combat 22-20
- Loadouts 22-21

Air Missions 22-22

Fuel Use and Aerial Refueling 22-23
- Aerial Refueling Systems 22-23
- Refueling Missions 22-24

Air Bases and Air Missions 22-25
Preparing Missions 22-26
Landing Air Units 22-27
Repairing Aircraft 22-27
Quick Reaction Alert Status 22-28

22.1. The Aggregate Warfare Model

VR-Forces supports a warfare model for aggregate-level scenarios. The aggregate warfare model requires use of *AggregateLevel.sms* or a similar SMS and only works with HLA Evolved and the MAK FOM extensions.

The aggregate warfare model is data-driven. Simulation objects have combat power and combat vulnerability. Their overall state is expressed as their health. These are abstract values that represent relative strengths of different units.

When opposing force simulation objects come into contact, they inflict damage on each other until one of the simulation objects is destroyed or breaks off combat. A unit is considered killed when its health is 0.

Some of the features that distinguish aggregate-level scenarios from entity-level scenarios include:

- Footprint. The area occupied by the unit.
- Posture. The unit's mode of interaction with the environment.
- Logistics. Management of personnel and matériel.
- Combat engineering objects. Tactical graphics that affect simulation object movement and health.
- Reports. Some data tracked in a simulation, such as personnel levels and pacing and tracking, is not used by the back-end. It is sent over the network to be used by command and control systems and human participants.



AggregateLevel.sms includes some individual entities, such as surface vessels and aircraft. However, these entities also use the aggregate warfare model. They do not behave like they would if you were using the same entity types in entity-level modeling.

22.1.1. Unit Combat

Combat power is the ability of a simulation object to cause casualties and damage, and is a function of how many weapons it has, the size (destructive power) of their munitions, the rate of fire of the weapons, the accuracy of the weapons, and the effectiveness of the subordinate simulation objects in conducting an attack (tactics, coordination, target selection, and so on).

A simulation object's ability to engage in combat is expressed as a strength value in one or more of the following domains:

- Anti-air
- Anti-tank
- High explosive
- Anti-personnel
- Anti-ship.

The strength value describes the attrition rate per second imposed on the target by the attacking unit.

A combat domain has a range. When an opposing force simulation object is within the range of a combat domain for which a simulation object has strength, it attacks the simulation object.

Health is a function of the amount of equipment, personnel, or both in the simulation object, and the difficulty in destroying them. For example, a unit with 10 tanks would have twice the health of a unit with 5 of the same kind of tank. However, note that the initial health for a unit is set by the SMS designer in the Simulation Object Editor. It is independent of other parameters or resource variables. It is not computed based on the equipment in the unit. Also, health values in different domains are not usually comparable to each other. They are only meaningful relative to units in the same domain.

Simulation objects also have a vulnerability in each of these domains. Base vulnerability describes how susceptible a simulation object is to a given category of weapon. In particular, it describes how easy it is to hit and kill the equipment in the simulation object with the given category of weapon. For example, infantry is very susceptible to high explosive munitions, light armor fairly susceptible, and heavy armor not very susceptible.

Vulnerability is expressed as a modifier of the combat power (attrition rate) of the attacker. It is multiplied by the attrition rate to calculate the actual damage. For example, a Mech Inf CO US has an Anti-personnel strength of 78. A Mech Inf CO RU has an Anti-personnel vulnerability of .1. If the US company attacks the Russian company, the actual damage for personnel will be 78 * 0.1 = 7.8 per second.

Both combat power and vulnerability are modified by a variety of factors, including posture, MOPP level, sector, and morale. The combination of all modifiers results in the final attrition rate.

The attrition across all domains is applied to the simulation object's health. If health declines to 0, the simulation object is killed.

22.1.2. Unit Footprints

A unit footprint is an abstract representation of the area the unit occupies with vehicles, personnel, and equipment. A unit takes damage from events that occur within its physical footprint, such as explosions or other simulation objects firing into that area.

The physical footprint has four sectors: front, rear, left flank, and right flank. The interactions of the unit with other simulation objects in the scenario varies based on which sector another entity is encountered on. For example, a unit is typically much more vulnerable to attack on the rear and flank sectors.

Both the physical footprint size and the size of the sectors for a unit are configured with a base value, but can then change during the scenario depending on actions of the unit.



The footprint is always circular. The footprint parameter describes the radius.

22.1.3. Unit Posture

Posture is an abstraction of the positioning of the equipment and personnel of a unit within its physical footprint, and its state of readiness for various activities.

The supported postures are:

- Deliberate-Defense. Planned, prepared. Lower mobility, lower vulnerability, moderate attack power. It is a positional defense.
- Hasty-Defense. Reactionary. Moderate mobility, moderate vulnerability, lower attack power. Happens when the unit first makes contact with the enemy. This is a positional defense, but with less time to prepare than deliberate-defense.
- Deliberate-Attack. Planned, prepared. Moderate mobility, moderate vulnerability, higher attack power. A fully planned attack executed as a deliberate advance using fire and maneuver techniques.
- Hasty-Attack. Reactionary. Higher mobility, higher vulnerability, lower attack power. Used for movement when enemy contact is expected.
- Reconnaissance. Mobile maneuver, locate and report on enemy, avoid detection, avoid engagement (if you can). Sensor signature and vulnerability are low, but so are combat power and speed.
- Travel. Very high mobility, but very vulnerable to attack and low attack power. Useful when the unit is moving from one location to another in a safe area.
- Rout. Health is less than 50% and unit is under attack. It cannot change to another posture until its health exceeds 50%.

A unit's posture affects:

- Footprint. The amount of space that the unit occupies.
- Sensing. For example, in reconnaissance posture the unit detects objects more quickly and at a greater distance than in a defensive posture.
- Sensor signature. The ability of other units to detect a unit is affected.
- Movement speed.
- Sector sizes.
- Combat power. Many weapon systems can only be used in certain postures. In particular, most cannot be used in Travel posture (which is the posture a simulation object is in when it is created).
- Vulnerability.

When a unit transitions between postures, it does so over time. Transitions may take up to several hours. The transition delay models the planning and preparation necessary to perform the task expected in the new posture. For example, deliberate defense is a well prepared defense that assumes planning fires, digging fighting positions, and so on. It significantly reduces the vulnerability of the defender. However, it may require 3 hours to transition to this posture. Hasty attack is the only posture that does not require a long transition time, as it is the posture for quick actions on contact, meeting engagements, pursuits, and so on. Transition times between each posture are configured in the Simulation Object Editor.

Because transitioning between postures can take hours of simulation time, when you set up a scenario you must decide if you want units to start in a given posture or transition to it as part of the simulation.

Defense postures assume stationary defense.

Aircraft and ships generally do not distinguish postures, except that they are not battle-ready in Travel posture.

22.1.4. Unit Movement

Units try to move at their ordered speed. The movement speed of a unit over the terrain depends on many factors, including the following:

- The slope of the terrain and the broad categories of the terrain underneath the unit, alter the maximum possible travel speed.
- Roads allow units to move more quickly, while restricted areas like forests or cities slow them down.
- Rivers and water bodies may stop a unit completely, although bridges allow them to cross.
- Overlapping footprints cause units to slow down.



The movement speed modifiers for slope, terrain, footprint overlap, MOPP, posture, and so on only affect maximum speed. If ordered speed is less than the net modified maximum speed, you will not see a change in speed due to these factors.

For movement speed calculations, the center point of the unit is used to determine what type of terrain the unit is on.

If the footprints of two units overlap, their movement is slowed based on their Max Speed Modifier When Overlapping Another Unit parameter. This parameter is set on the Movement tab for a unit in the Simulation Object Editor. The lower the value, the greater the movement penalty. When unit footprints overlap, the lowest modifier among all overlapping units is applied to all of them.



Aggregate-level scenarios do not support collision avoidance among simulation objects.

The effect of terrain features on movement is configured in a unit's movement system. For details, please see "Configuring Aggregate-Level Movement Restrictions," on page 65-9.

VR-Forces supports a variety of combat engineering objects that affect entity movement. They may cause a unit to go more slowly or they may damage the unit. For information about combat engineering objects, please see "Combat Engineering Objects," on page 22-11.

22.1.5. Limited Munition Attack

Most aggregate-level combat is automated and continues until a combatant is killed or combatants run out of ammunition. There are no specific fire or detonation interactions involved and combat essentially takes place in the background. The aggregate warfare model also supports limited munition attacks, in which a limited amount of munitions are used and the attack then ends. In the case of artillery, this is a specified number of rounds from a battery, delivered over a short time. In the case of large munitions such as missiles and bombs, the attack for each munition is generally instantaneous. VR-Forces does not model the time it takes for a munition to reach its target.



Not all missile attacks are modeled as limited munition attacks. Anti-tank missiles in ground units, for example, are launched by multiple launchers in the unit against multiple targets in the defender, and are modeled with normal combat power.

Individual missile attacks typically make use of a hit factor from the attacker and a defense factor from the defender to determine a hit probability. The missile must hit before its combat power is applied in an attack.

22.1.6. Contamination Areas

The aggregate warfare model supports nuclear, biological, and chemical contamination areas (NBC areas) that may damage or impede a unit. You can create contamination areas as tactical graphics, or simulation objects with nuclear, biological, or chemical weapon systems can create them as a result of attacking with these weapons.

Contamination areas have parameters that are similar to other areal tactical graphics, plus the following parameters:

- Agent type. The type of contamination. It determines the configuration used for damage calculations.
- Terrain type. Informational only; not used by VR-Forces.
- Topography type. Informational only; not used by VR-Forces.
- Vegetation type. Informational only; not used by VR-Forces.

The terrain, topography, and vegetation types are published by VR-Forces, but do not affect unit modeling or interaction. They may be useful to other applications that interact with VR-Forces.

By default, NBC hazard areas are placed on the Contamination Areas overlay.

Contamination (NBC) Areas and MOPP Level

When a unit is attacked by NBC weapons or enters an NBC area, it suffers attrition. It stops its current task and increases its mission oriented protective posture (MOPP) level until it stops suffering attrition. Then it resumes its task. However, any contamination that took place persists.

If a unit is moving and it detects an NBC hazard area in its path, it stops moving and increases its MOPP level to a point that will allow it to pass through the hazard area without suffering attrition.

Increasing MOPP level affects a unit's speed, combat effectiveness, sensor sensitivity, posture transition time, and vulnerability.



Some types of contamination are benign enough that they cause little, if any, damage to simulation objects. Other types, such as gamma rays, cause attrition even at the highest MOPP level.

22.1.7. Unit Sensors

Simulation objects in aggregate-level scenarios use the same sensor signature mechanism as simulation objects in entity-level scenarios. Sensor systems allow them to detect other simulation objects in various domains. Sensor signatures indicate how detectable they are by other simulation objects in the applicable domains. The main difference between simulation objects in the two types of modeling is in the detection checks. Simulation objects in entity-level scenarios do a line-of-sight check from the center of the simulation object . Simulation objects in aggregate-level scenarios do three checks – from the center of the outer ring of the footprint and from the edges of the circumference of the footprint, as illustrated in Figure 22-1.

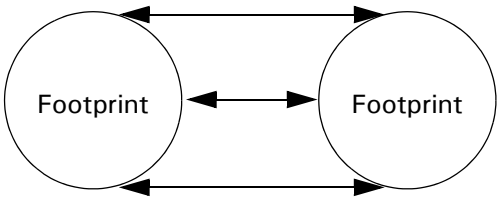


Figure 22-1. Aggregate-level sensing

22.2. Combat Engineering Objects

VR-Forces supports a variety of combat engineering objects that affect unit movement. They may cause a unit to go more slowly or they may damage the unit. Combat engineering objects (CEOs), such as fortifications, can also protect a unit, in that they affect the effectiveness of opposing forces.

CEOs are displayed as tactical graphics. You can create combat engineering objects from the Tactical Graphics Palette. Also, units can create combat engineering objects in response to combat engineering object tasks.

Units can counter the presence of combat engineering objects by breaching them, bridging them, defusing them, or destroying them.

Combat engineering objects can have the following effects on units:

- Mobility. CEOs can decrease entity mobility. Mobility modifiers are configured by mobility type. For instance, an object might affect the mobility of soldiers on foot more than tanks. When a simulation object comes in contact with an engineering object, it checks the object type's mobility modifier against the entity's mobility type. The entity slows down based on the modifier value.

  The modifier is scaled based on the completion percentage of the object. Modifiers only affect the maximum speed, so speed is affected only if the modified maximum speed is less than the current speed of the unit.

  If a unit encounters an obstacle that obstructs its movement or may cause damage, it stops moving and sends a message to the console indicating that it is obstructed (Figure 22-2). The user must then tell it what to do. The console message may use the question and answer capability. (For information about console questions, please see "Answering Questions from Scripted Tasks," on page 16-26.)



The unit may not be able to move until you respond to the question.

If an obstacle is concealed, the unit does not stop. For information about concealed obstacles, please see "Concealed Obstacles," on page 22-12.

- Mobility enabling. Certain types of engineering objects are treated as terrain features by a moving unit. For instance, a simulation object may determine that an engineering object represents a road or bridge, giving it additional path planning options. This capability is either on or off.
- Vulnerability. Vulnerability modifiers are configured by weapon type. For instance, a fortification might do very well at stopping anti-personnel weapons, but may do little or nothing to stop high explosives. When a simulation object under fire contacts an engineering object, it checks the object type's vulnerability modifier against the attacking weapon's category. This may reduce the amount of damage the entity takes from that weapon.

  The modifier is scaled based on the completion percentage of the object.

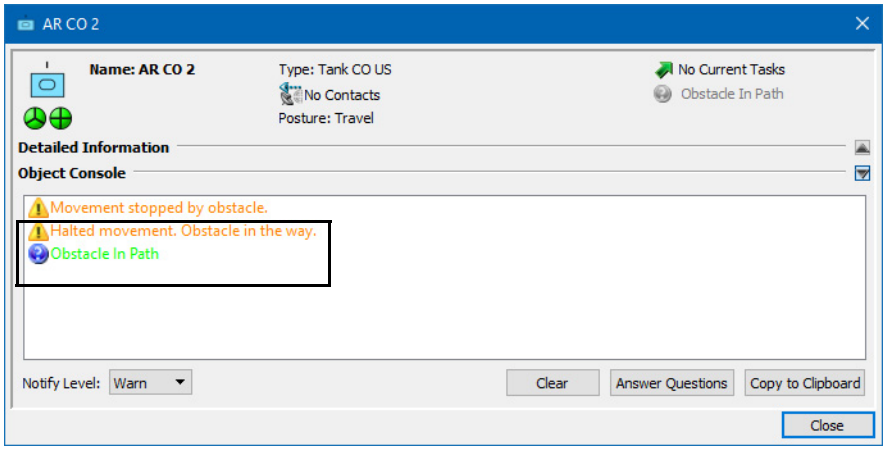


Figure 22-2. Question in entity console

- Visibility. When a simulation object contacts an engineering object with a visibility modifier, its visibility to opposing simulation objects is increased or decreased accordingly.

  The modifier is scaled based on the completion percentage of the object.

- Attrition over time. When a simulation object contacts an engineering object that causes damage (for example a minefield), that object continues taking damage over time as long as it remains in contact. The damage is configured as one of the existing damage/weapon categories, for example, high-explosive. Simulation objects with more or less vulnerability to that damage type take more or less damage accordingly.

  The attrition is scaled based on the completion percentage of the object.

- Immediate attrition. Some engineering objects cause one-time, immediate damage to any unit that comes in contact with them. For instance, a simulation object might trigger an IED. Usually the engineering object is destroyed in the process.

  The modifier is scaled based on the completion percentage of the object.

  The various modifiers are configured for each CEO in the Simulation Object Editor.

22.2.1. Concealed Obstacles

By default, minefields, booby traps and unexploded ordnance objects are concealed. (This attribute is configurable in the Simulation Object Editor, so you can set it for other objects if you want to.) The engineering object sensor does not automatically detect concealed objects within its sensing radius. The sensor has a random (configurable) chance of detecting the object each sensor cycle that the object is within range.

If a sensor detects a concealed minefield, the Mark Minefield reactive task marks it as detected. Thereafter, the minefield is always 100% detectable. (The Mark Minefield reactive task is enabled by default.)

Unit movement systems check the concealed flag for obstacles. For non-friendly obstacles that are concealed, the "stop and ask" behavior is suppressed unless the object has been specifically detected by the sensor. Therefore, if the sensor has not detected the object, the unit enters it without first stopping and alerting you to the presence of the object.

Concealed objects are always discovered if they are causing damage to a simulation object.

22.2.2. Creating Combat Engineering Objects

You can create combat engineering objects directly from the Tactical Graphics Palette or you can task simulation objects with the appropriate systems, such as combat engineering units, to create them.

Combat engineering objects are implemented as points, lines, and areas, so you can create them the same way that you would create tactical graphics. Most areal CEOs are fixed size, which is configurable in the Simulation Object Editor. You just have to specify the lower left corner of the area.

When you create a CEO from the Tactical Graphics Palette, you can give it a name, specify a layer and set all the parameters you would expect for that type of object. When you task a unit to create a CEO, it is given a default name and you can only specify its location.

Many tactical graphics have the same default names as CEOs, for example, fortified lines and minefields. They may also look the same on the map. However tactical graphics do not affect unit movement and combat the way that CEOs do. To help distinguish tactical graphics from CEOs, CEOs created by units are placed on the Combat Engineering Object layer. It is recommended that if you create CEOs from the Tactical Graphics Palette, that you place them on the CEO layer.

22.2.3. How Engineering Units Create Engineering Objects

When a unit with engineering capabilities creates a CEO, it expends construction material at a rate specified in the object's model (configured in the Simulation Object Editor). Depending on the type of objects, the process can take hours of simulation time.

While an object is partially complete, its effectiveness is reduced. Some effects, such as enabling mobility over previously impassable terrain (for example by using bridges), do not take effect until the object is complete.

You can follow the progress of object construction in two ways. You can view the engineering unit's use of construction material on the Supplies page of its Information dialog box. You can view the percent of the object's completion on the Engineering Object Information page of the Information dialog box for the object (Figure 22-3).

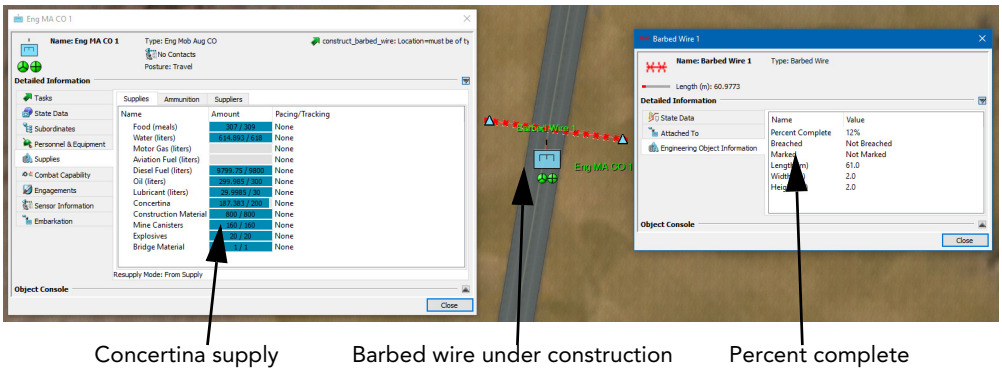


Figure 22-3. Barbed wire under construction

Units create combat engineering objects as follows:

1. The unit moves within range of the location to create the object.
2. The unit creates a new object that is 0% complete. The object is displayed on the terrain.
3. The unit begins constructing the object. As it does so, its supply of construction material decreases.
4. When the object is complete or the unit runs out of construction material, it ends the task.

If a creating unit is interrupted and must halt construction, it notifies that object that construction has stopped. The object stays at its current level of completion, which may or may not be useful, depending on the object type. A constructing unit might also reduce its rate of construction if it takes damage.

If the constructing unit is able to resume construction or another unit is tasked to improve the object, it moves within range of the object and construction starts again.

Multiple units with construction capability can work together. To task a second unit to assist the first, task it to improve the object under construction.

You can also set the completion percentage of a combat engineering object with the Percent Complete set data request.

22.3. Logistics

Simulation objects have resources, such as fuel, ammunition, and food. Over time, and particularly during combat, these resources are consumed. Logistics is the process by which a unit resupplies itself.

Units can resupply the following resources:

- Health
- Food
- Water
- Motor gas
- Aviation fuel
- Diesel fuel
- Oil
- Lubricant
- Anti-Tank
- High-Explosive
- Anti-Personnel
- Anti-Air
- Anti-Ship
- Large-Munition
- Other.

Anti-Tank, High-Explosive, Anti-Personnel, Anti-Air, and Anti-Ship refer to ammunition in that category. There is no distinction made between different ammunition names within a category. Large-munition covers all other categories of ammunition, such as indirect fire munitions, bombs, and missiles. Other supplies covers any other type of supply, including engineering materials.

The Supplies page on the Information dialog box indicates when a simulation object is receiving or providing supplies. You can display supply lines that show when a supply unit is supplying another unit (Figure 22-4). In Figure 22-5, green arrows show the supplies that the entity is receiving.

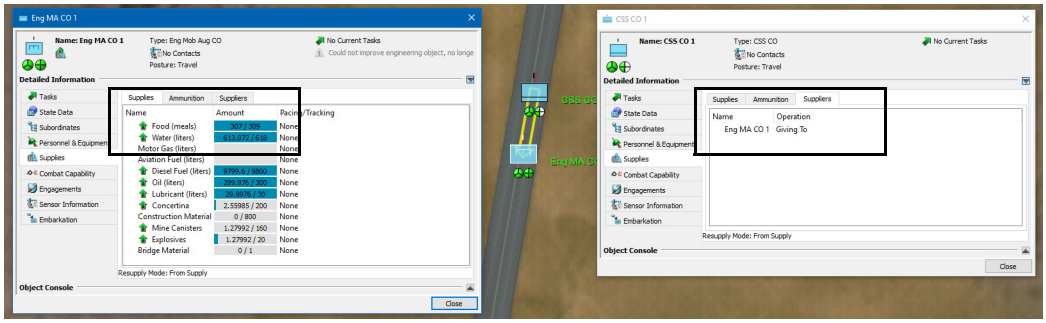


Figure 22-4. Supply information and supply lines

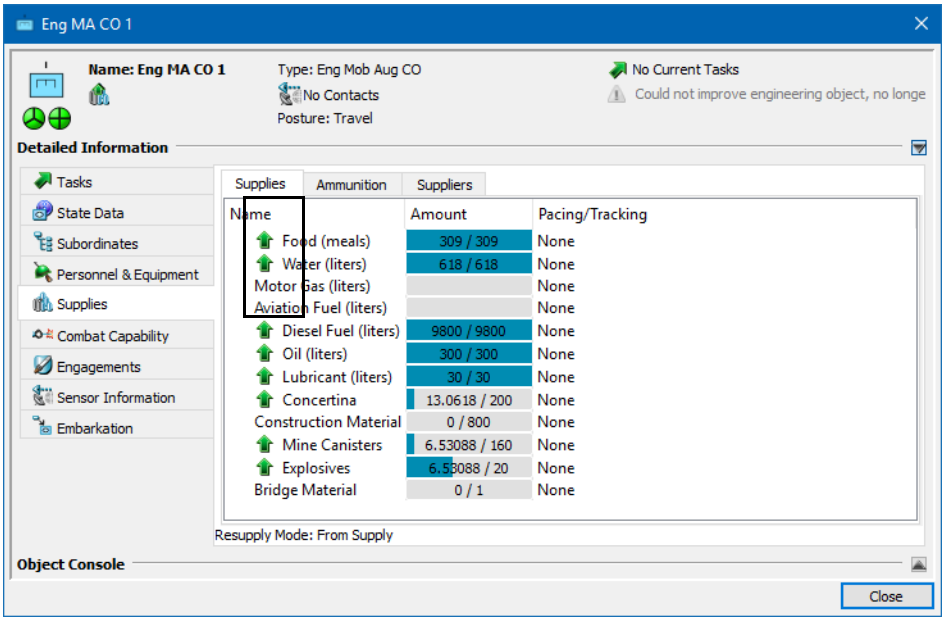


Figure 22-5. Supplies being received

22.3.1. Receiving Supplies

Units can receive supplies in the following ways:

- Take supplies from a supply unit. Units take supplies from a friendly supply unit that is within range and has notified them that it can provide supplies. The resupply range is specified in the Simulation Object Editor as part of a supply system.
- Auto Resupply Continuously. Supplies increase continuously at a configurable rate.
- Auto Resupply Periodically. Simulation objects refill all supplies at specified intervals. The interval is configurable.
- Refueling. Air units can refuel from tanker units. For details, please see "Fuel Use and Aerial Refueling," on page 22-23.

The ability to resupply can be restricted if a simulation object is moving, attacking, or defending. This is configured in the Simulation Object Editor on the Supplies tab. You can change how a simulation object receives supplies using the Set Resupply Mode task. If they are configured for airborne refueling (Can Receive Fuel While Airborne parameter), airplanes can receive fuel while they are flying, but must be on the ground to receive any other type of supply.

22.3.2. Providing Supplies

Simulation objects can provide supplies to other simulation objects if they have a resupply system (*resupplier.sysdef*). A supply unit can be configured to provide any of the supplies that are supported. A supply unit can provide supplies to any number of other simulation objects. It is assumed to have its own supply source, so it never runs out. You can change a supply unit's status with the Change Supplying Command task.

A supply unit runs a background process that tests for friendly simulation objects in its area and notifies them that it is available to provide supplies. It checks for conditions that might prevent a simulation object from receiving supplies. It stops providing supplies to simulation objects that move out of range.

Aircraft units that have a refueling capacity can refuel other air units that request refueling.

22.4. Electronic Warfare

The aggregate warfare model supports three types of electronic warfare – communications jamming, radar jamming, and sensing electronic emissions.

22.4.1. Jamming Communications

Each unit has an EW-Comms-Dependence and an EW-Defense-Strength property. The dependence property indicates how much it can be affected if its communications are jammed. The defense factor indicates how well its technology and practices protect it from communications jamming. Units also have, in their state properties, a list of jamming attacks being made against them (type and strength) and a net EW-Comms-Degradation-Percentage.

When a simulation object attacks another entity with jamming, it sends EW attack information to the target.

Simulation objects jam all enemy simulation objects in range; there is no target selection. Friendly simulation objects are not affected.

When a simulation object suffers communications jamming attacks, the strengths of all attacks are added together and compared to the defensive strength. The ratio determines a modifier between 0 and 1, which is multiplied by the EW-Comms-Dependence. The result is a degradation percentage. This degradation percentage is applied to reduce combat power, and increase vulnerability and posture change time (by a maximum factor of 2.0).

Units can have a system called an aggregate-comms-jammer. This system has parameters for strength, maximum range, and a set of allowed postures. Strength falls off with $1/r^2$ (r is range in kilometers). This system can trigger EW attacks using the Make EW Attack background scripted task. It has a jammer controller that can publish an emitter.

You can enable or disable the emitter using the Emitter set data request. Turning it on activates jamming, and turning it off deactivates jamming. If the emitter is turned on while the simulation object is not in an allowed posture for jamming, a warning is printed to the console. In this case the emitter is on, but it is not used for jamming attacks.

When jamming is enabled the Make EW Attack script determines which jamming systems on the simulation object are on, and then uses them to attack all targets within range. It updates (new) EW-Attacks and Jamming-Strength-On state properties on the simulation object. The Jamming-Strength-On property summarizes the strength from all jammers the simulation object has turned on, and is used for emissions sensing. The Update_EW_Degradation background script computes the effects of EW attacks on a simulation object.

The MI PLT (military intelligence platoon) unit is configured to have an aggregate-comms-jammer.

22.4.2. Jamming Radar

Each unit has a state property Radar-Jamming-Strength-Receiving, which is updated by the Update_EW_Degradation background script. This is the total radar jamming strength being received.

Radar jamming can have the following effects:

- Reduce the sensitivity of radar sensors so they cannot detect targets.
- Reduce the hit factor of radar-homing missiles so they have a smaller hit probability.

The active-radar-sensor system has a parameter called jamming-defense-factor. This factor reflects the technology of the radar (not power) and is used with the jamming strength received to compute a sensitivity modifier for the sensor. The modifier varies from 0 to 1.

The weapon systems that use a controller that uses an aggregateReleaseBomb-ControllerDescriptor have parameters can-be-radar-jammed and jamming-defense-factor. This includes the following weapon systems:

- aggregate-antiair-missile
- aggregate-antiship-missile
- aggregate-land-attack-missile
- aggregate-release-bomb
- aggregate-unguided-bomb.

If can-be-radar-jammed is true, then jamming-defense-factor is used with jamming strength received to calculate a modifier for the hit factor of the weapon.

Weapon systems are configured with either a constant or variable for can-be-radar-jammed, to provide examples of different guidance systems:

- Anti-air missiles use a variable, to allow modeling of IR or radar-homing missiles.
- Anti-ship missiles set can-be-radar-jammed true, thus assuming radar guidance.
- Land attack missiles use a variable.
- Release-bomb omits the parameter (default is false), to simulate optical guidance.
- Unguided bomb omits the parameter (default is false).

The aggregate-radar-jammer system can publish an emitter and can be turned on and off using Emitter set data request. It enables the EW_Attack script.

The radar jammer strength falls off with $1/r^2$ (r is range in kilometers).

The EA-18G entity is configured with a radar jamming system.

22.4.3. Sensing Electronic Emissions

The emission sensor domain allows simulation objects to sense RF electromagnetic emissions from simulation objects. Each simulation object has an emissions signature, which is a parameter for each object type. This can be detected by the SIGINT sensor. This is the "normal background signature".

There is a sensor signature modifier for emissions. VR-Forces determines if specific emitters are turned on, thus adding to the normal background signature for the simulation object .

The emissions signature modifier has a parameter, jam-strength-factor, that converts jamming strength to a signature value. It is set in the platform files.

There is a sensor system emissions-sensor for emissions. It is called a SIGINT (signals intelligence) sensor. The MI PLT unit is configured with a SIGINT Sensor.

22.5. Air Combat

All air-to-air and air-to-ground engagements are executed using the limited munitions attack capability in the aggregate model. The core of this model is that a single munition is fired at a time, and then a hit calculation is performed. The hit calculation uses the hit-factor of the shooter and compares it to the defense factor of the target to determine a hit chance. A random draw is done and compared to the hit chance to determine if an actual hit takes place. If the hit occurs, then the attack power of the munition is applied to the target.

Aircraft units can represent one or more aircraft. Each aircraft has a primary equipment attribute, which is usually the airframe. When aircraft units engage in combat, damage is calculated against the primary equipment. For example, if an F-16 aircraft has the F-16 airframe as its primary equipment, a unit that represents four F-16s has a primary equipment count of four. When that unit takes damage, no more than one F-16 can be damaged by each hit.

Each aircraft type has a loadout that is defined in the Simulation Object Editor. A loadout specifies the equipment and weapons for the aircraft. At runtime, you can set the loadout for an aircraft unit. The loadout for a unit determines how effective it is when it attacks another simulation object.

The results of air-to-air engagements are determined by the following rules:

- Sensor capabilities. The sensor range and sensitivity, and the sensor signature of the target determine the maximum range for detection, and therefore the maximum range for engagement of a target.
- Weapon type. The weapon types on the aircraft determine the maximum engagement range when engaging with a weapon. The weapon type also determines the hit factor that affects the chance of a weapon hitting, and the total damage done if the weapon does hit.
- Tactical Data Link. If the aircraft unit is equipped with a Tactical Data Link combat system, the unit receives a bonus to its defense factor.
- Number of aircraft. Defending aircraft units receive a combat maneuvers bonus to their defense factor. Attacking aircraft have a smaller attack interval when the number of aircraft is larger, causing them to fire more missiles in a shorter period of time. This bonus is based on the current number of aircraft in the unit. It is reduced if some of the aircraft are destroyed during combat.
- Battle management support. If the attacking unit has active spot-report data on the target aircraft from other units in the scenario, it receives a hit-factor bonus when engaging the target. A defending unit also receives a defense-factor bonus if it has active spot reports on the attacking entity.

The details of an engagement are displayed in the Information dialog box.

The AWACS unit is configured with long range radar. It can provide spot report tactical data to other aircraft units, which allows them to engage units that are beyond the sensor range of their aircraft.

22.5.1. Loadouts

Aircraft loadouts are defined in the Simulation Object Editor for each aircraft type. Each defined loadout specifies the quantity of weapons and equipment in the loadout. At scenario runtime, you can set the loadout for an aircraft unit and the unit will be equipped with the predefined loadout items.

The weapons and equipment determine the effectiveness of the attacks a unit can make. Additional fuel tanks can increase the range of aircraft. Jammer pods increase defense against radar guided weapons.

22.6. Air Missions

Missions are abstract command objects that link together aircraft that will be launched as a flight. A mission has the following attributes:

- Name (call sign).
- List of base aircraft in mission.
- Aircraft loadout.
- Target launch time.
- Time to start preparing for launch. This is computed from the target launch time and an estimate of the aircraft preparation time.
- Simulation object type for the air unit that will be created for this flight.

Air missions are carried out by aircraft units. A mission can be carried out by a single unit (which can represent multiple aircraft) or by multiple units. A multiple unit mission consists of multiple single units and a single "Mission Superior" object that is the superior of all the single units that are part of the mission.

The icons for air units have graphics that indicate the number of aircraft in the unit (Figure 22-6). Information dialog boxes also show the number of aircraft. If a unit suffers attrition, the number of graphics is reduced (Figure 22-7).



Figure 22-6. Aircraft unit icon

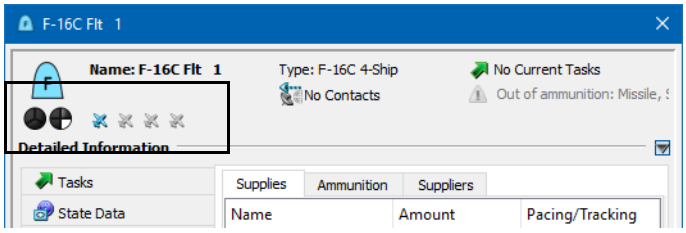


Figure 22-7. Aggregate information after attrition

The units that execute a mission can be created as part of the initial scenario laydown, they can be created during runtime by plans, players, or instructors, or they can be launched by an air base.

For single unit missions, the mission orders are issued directly to the unit. For multi-unit missions, the mission orders are issued to the mission superior.

Mission orders may include a task to land at an air base at the end of the mission. The units executing the mission are deleted when they land, so the mission ceases to exist. Any report of mission status should be made by the mission plan before this occurs.



VR-Forces does not have any specific mission management features. Users can manage missions with the global and unit plans, through player interaction, or through externally developed applications that use the remote control API.

22.7. Fuel Use and Aerial Refueling

Fuel is a supply that is configured in the SOE (on the Supplies tab) and consumed based on aircraft activity. Air units use fuel based on their altitude and speed. When an aircraft unit is in a holding pattern, fuel use is calculated based on the default speed. VR-Forces interpolates values for altitudes and speeds that fall between the configured values.

For tankers, the fuel available for refueling other aircraft is assumed to be the same fuel that the tanker itself uses to fly. When refueling is complete, aviation fuel supplies are transferred from the tanker to the receiving aircraft.

22.7.1. Aerial Refueling Systems

An aircraft unit can refuel other aircraft if it is configured with an aerial refueling system. The fueling system can be a flying boom type, a drogue-and-probe type, or a combination. The configuration cannot be changed during a simulation run.

The boom system specifies a maximum fuel transfer rate. It can also be configured to have drogue lines on it, with a different transfer rate for the drogues.

Drogue systems define the number of drogues and the fuel flow rate for each fuel line.

Aircraft that are to receive fuel must be configured with an aerial refueling receiver system. The aerial refueling receiving system specifies whether the system is a boom or drogue system and what the maximum fuel transfer rate is. Air units may only refuel from tankers that have compatible refueling equipment.

Refueling systems and fuel receiving systems get assigned on the Additional Systems tab in the SOE. *AggregateLevel.sms* has two tanker units, a KC-135A with a boom system, and a KC-130J with two drogues.

22.7.2. Refueling Missions

Air units can be tasked to refuel from a tanker unit with the Refuel from Tanker task. Tanker units provide fuel in response to a Refueling Mission task. When a tanker unit receives a Refueling Mission task, it flies to the designated location, flies a race track pattern, and services aircraft that need fuel. It can continue to refuel aircraft until its fuel level falls below a configurable threshold. (For details about the Refueling Mission task, please see ("Refueling Mission," on page 30-31). For details about the Refuel from Tanker task, please see ("Refuel from Tanker," on page 30-30).)

The tanker coordinates with the refueling aircraft by exchanging text radio messages. The aircraft unit initiates a refueling request and says what type of system it is using and how much fuel is required. The tanker then either rejects the request (based on refueling system type and amount of fuel available) or accepts it. The aircraft unit is expected to move close to the tanker and report in with a message. When the tanker has a refuel line available, it tells the unit to move to the tanker and begins fueling it. The transfer rate is dependent on the transfer rates of the tanker and the aircraft, and the number of aircraft and drogue lines available. Supply transfer messages are sent to the refueling aircraft periodically, and corresponding aviation fuel resources in the tanker are reduced.

When refueling is complete, the tanker notifies the refueling unit.

If multiple aircraft units request refueling, the tanker coordinates with each of them and refuels them each in turn.

22.8. Air Bases and Air Missions

An air base is a special type of entity that is placed on the ground. Air bases have a variety of resources that they can use to prepare, launch, and recover air missions. They do this using a background process (Manage Air Base) and several tasks and set data requests.

An air base represents the following physical features:

- Apron. The aircraft that are assigned to an air base wait on the apron to be assigned to missions, prepared for launch, or sent to a hangar to be repaired. The apron has a maximum capacity of aircraft.
- Hangar. When a damaged aircraft is actively undergoing repair, it is moved to a hangar. Hangars have a maximum capacity of aircraft.
- Runway. Aircraft that are prepared for a mission taxi on the runway area for a few minutes before taking off.
- Operational Readiness Platform (ORP). The location at the end of a runway where aircraft on Quick Reaction Alert (QRA) are stationed. The ORP has a maximum capacity of aircraft.

Air bases have the following types of resources:

- Aircraft. All of the aircraft at an air base are on the ground. Air bases do not track the aircraft that are executing missions. When an aircraft takes off, it is deleted from the air base records. Aircraft have the following attributes:
  - Type (equipment name).
  - Health. Health is expressed as a percentage and represents the status of the air frame and the amount of fuel on board.
  - Number of munitions loaded.
  - Location on the air base.

  An aircraft can be in one of the following states:
  - Available. The aircraft is at 100% health and is available to be assigned to a mission.
  - Assigned. The aircraft has been assigned to a mission. Some time before the mission launches, an assigned aircraft begins fueling and arming and its air crew gets ready.
  - In Repair. The aircraft is in this state if its health is less than 100%. It can only be actively repaired (have its health improved) if it is in a hangar and has ground crew assigned to it.
  - QRA. The aircraft has been assigned and prepared, and is waiting at the ORP.

- Air crews. Each aircraft must be manned by one air crew. An air crew can man any type of aircraft. Air crews can be available or assigned. Air crews are assigned (to missions) when the air base is tasked to launch aircraft.
- Ground crew personnel. Ground crew personnel are used to repair, fuel, and arm aircraft. They may be in one of the following states and locations:
  - Available. Not assigned to any activity.
  - In-Hangar. Repairing aircraft.
  - On-Apron. Fueling and arming aircraft.
- Fuel. A single type of fuel is used to fill all aircraft prior to launching them on a mission.
- Munitions. Large munitions (bombs, missiles, and rockets) are aggregated into one count of munitions for the air base. When an aircraft is assigned a mission and given a loadout, the quantity of munitions in the loadout is deducted from the base munition count.
- Runways. An air base can have one or more runways. Runways have a length and a heading. The length of a runway determines whether aircraft can land at the air base.

You can specify the resources of an air base with the following set data requests:

- "Assign Aircraft," on page 35-4.
- "Facilities," on page 35-8.
- "Personnel and Supplies," on page 35-13.

22.8.1. Preparing Missions

When an air base has pending missions, the Manage Air Base process handles the preparation of the aircraft as follows:

1. It estimates the preparation time and sets a time to begin preparation so that the flight can launch at the desired time.
2. When the preparation time arrives, the process assigns ground crew to the aircraft and computes the time at which fueling, arming, and preparing the air crew will complete. These tasks occur in parallel.
   - Fueling time requires a setup time and depends on how much fuel is taken on board.
   - Arming time is constant. It is independent of the loadout.
   - Crew preparation time is constant. It is independent of the type of aircraft.
3. When all preparation tasks are complete, the ground crew is made available to the air base.
4. When all aircraft in the flight are ready, they taxi onto the runway.
5. An air unit is created and the aircraft counts in the air base are decremented to remove the flight from the air base.

The *AirBase.ope* platform file includes parameters for the Manage Air Base script that determine:

- The size of the ground crew needed to fuel an aircraft.
- The size of the ground crew needed to arm an aircraft.
- The time required to arm an aircraft.
- The time required to set up refueling.
- The rate at which fuel can be transferred to the aircraft.
- The time required for the air crew to prepare.
- The time required for the flight to taxi and take off.

22.8.2. Landing Air Units

When an air unit wants to land at an air base (Land at Air Base task), it sends a message to the air base requesting to land. It tells the air base the number of aircraft in the unit and how long the runway needs to be for it to land. The Manage Air Base background process handles this request.

If the air base has room for the aircraft and its runway is long enough, it tells that air unit that it can land. The air unit flies to the air base and then requests final permission to land.

The air base determines the properties of the air unit, as follows:

- The aircraft type.
- Number of aircraft.
- Air frame health percent.
- Fuel in the tanks.
- Number of munitions.

When the air unit lands, the unit is removed from the scenario. The air base adds the aircraft to its list, and makes them either available or in-repair (if they are damaged). An air crew is added to the available air crews for each aircraft. Fuel and munitions are added to the air base's fuel and munition counts. The count of aircraft on the apron is incremented for each aircraft.

22.8.3. Repairing Aircraft

The Manage Air Base background process manages aircraft repair. When an air base has damaged aircraft, it moves them into the hangar and assigns ground crew personnel to them. The process can only start this repair if there is space in the hangar and ground crew available. Aircraft in the hangar are repaired at a rate that is proportional to the number of ground crew assigned. When the aircraft is at 100% health, the ground crew are made available again, and the aircraft is moved back to the apron and becomes available.

The *AirBase.ope* platform file includes parameters for the Manage Air Base script that determine:

- The repair rate (% repair / ground crew / second).
- The maximum number of ground crew that can be assigned to an aircraft.

22.8.4. Quick Reaction Alert Status

When aircraft are assigned Quick Reaction Alert (QRA) status, they prepare to take off and taxi to the Operational Readiness Platform (ORP), which is a set of parking spaces at the end of the runway. When events require it, one or more aircraft are ordered to take off and intercept a target. Aircraft and crews cannot remain prepared indefinitely, so after some time they must be returned to unprepared status at the base.

You can specify a level of quick reaction capability. For example, some QRA aircraft might keep their engines running, while others are fueled and armed on the apron with the crew in a ready room. In general the faster the reaction capability, the less time the aircraft can be on QRA.

Aircraft in QRA status remain part of the air base inventory rather than being formed into a unit.

You assign aircraft to the QRA status with the Prepare QRA task. You launch them with the Launch QRA Aircraft task. For details, please see "Prepare QRA," on page 30-30 and "Launch QRA Aircraft," on page 30-27.



23. Creating and Controlling Units

This chapter explains how to create and manage units.

Creating Units 23-2
Creating a Unit by Combining Existing Simulation Objects 23-3
Creating a Preconfigured Unit 23-5
Configuring the Unit Creation State 23-6
Selecting a Unit 23-6
Adding Simulation Objects to a Unit 23-7
Removing a Simulation Object from a Unit 23-8
Units and Unit State in Entity-Level Scenarios 23-8
How a Unit's State is Shown 23-9
How Changing Aggregation State Affects a Unit 23-10
Triggering Unit State Transitions 23-11
Aggregating and Disaggregating Units Manually 23-12
Configuring Automatic Aggregation and Disaggregation 23-12
Using Disaggregation Areas 23-12
Deleting a Unit 23-13

23.1. Creating Units

You can create a unit by selecting simulation objects (which can be individual entities or units) and combining them to form a unit, or by placing a preconfigured unit from the Simulation Objects Palette. In entity-level scenarios, most preconfigured units can be disaggregated to the level of individual vehicles and persons. In aggregate-level scenarios, most preconfigured units cannot be disaggregated.

When you create a unit, it is created as a subordinate to the force level. You cannot create units that are subordinates of an existing unit. Once you create a unit, you can subordinate it to another unit. (For details, please see "Adding Simulation Objects to a Unit," on page 23-7).



Except where noted, the procedures for creating and editing units apply to entity-level scenarios and aggregate-level scenarios.

23.1.1. Creating a Unit by Combining Existing Simulation Objects

When you create a unit by combining simulation objects, you can specify the order of subordinates in the unit. This determines which subordinate is considered the unit leader and affects the assignment of echelon IDs. (It also affects the icon used to represent the unit.) You cannot change the subordinate order after you create the unit.

- When you create a unit, the members of the unit get new echelon IDs to reflect the new entity hierarchy.
- If the simulation objects that you are aggregating have plans and your unit setting is to aggregate new units, the plans are deleted. However since individual plans get deleted whenever a unit enters the aggregated state, in entity-level scenarios you cannot count on simulation objects keeping their plans even if you change this setting.
- When you aggregate individual entities into a unit, you can include non-VR-Forces entities. These entities will be included for the purposes of calculating the bounding volume of the unit and for expanding and collapsing the display of unit members. However, VR-Forces actions that would not typically apply to non-VR-Forces entities, such as tasks and set data requests, will not affect the non-VR-Forces entities.

To create a unit by combining existing simulation objects:

1. Select the simulation objects that you want to become part of a unit.
2. Choose **Objects** → **Unit** → **Aggregate As**. The Aggregate As dialog box opens (Figure 23-1).

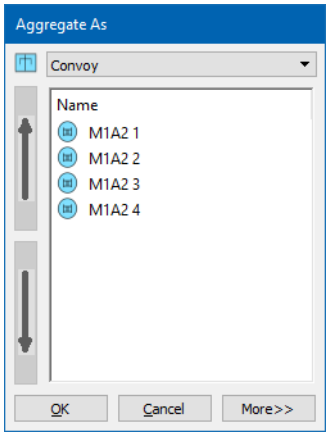


Figure 23-1. Aggregate As dialog box

3. Select a unit type from the list.

- The Convoy unit can only be assigned convoy tasks. Its only purpose is to allow you to quickly aggregate miscellaneous ground vehicles into a convoy. You cannot aggregate or disaggregate a convoy.
- The default list of unit types is limited and is different for entity-level scenarios and aggregate-level scenarios. You can specify the unit types that are on the list in the Simulation Object Editor. For details, please see "Specifying an Aggregate As Option," on page 70-3.

4. Optionally, customize the entity type enumeration for the unit, as follows:
   a. Click More. The dialog box expands to show the entity type enumeration for the unit type selected (Figure 23-2).

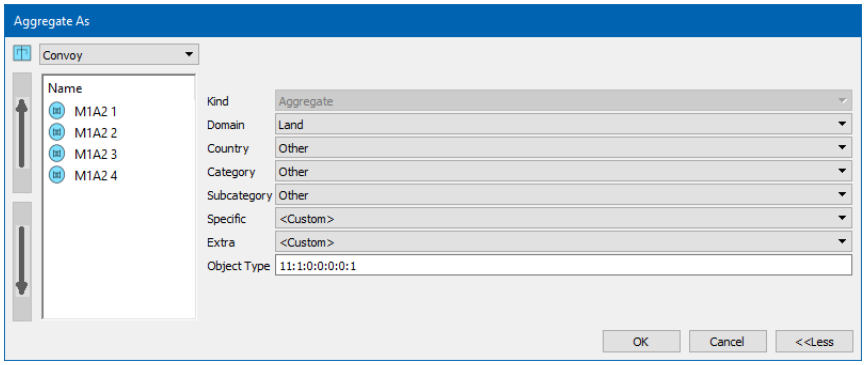


Figure 23-2. Aggregate As dialog box

   b. Edit the enumeration by selecting values from the lists, or by typing an enumeration.

If an edit to the enumeration results in an entity type that does not match the unit type selected in the list, the list entry does not change. However, the unit will be created with the entity type specified by the custom enumeration.

5. Optionally, change the order of subordinates, as follows:
   a. Select the subordinate whose order you want to change.
   b. Click the up or down arrow to move it to a new position in the list of subordinates.
6. Click OK.



Creating Higher Echelon Units in Aggregate-Level Scenarios

In aggregate-level scenarios, you can combine entities and units to create higher echelon units just as you can in entity-level scenarios. However, unlike in entity-level scenarios, you cannot switch these units between an aggregated and disaggregated state. They are always disaggregated.

23.1.2. Creating a Preconfigured Unit

EntityLevel.sms and *AggregateLevel.sms* include preconfigured units at several hierarchical levels. You can configure them and add additional units in the Simulation Object Editor. (For details, please see "Editing Unit Composition," on page 70-4.)

You create preconfigured units by selecting them on the Simulation Objects Palette and follow the standard procedures for creating simulation objects. The abbreviations in the unit names in *EntityLevel.sms* have the following meanings:

- ADA. Air defense artillery.
- COLT. Combat observation lasing team.
- CSS. Combat service support.
- FA. Field artillery.
- MI. Mechanized infantry.



By default, units in entity-level scenarios are created in a disaggregated state. You can change the default aggregation state. For details, please see "Configuring the Unit Creation State," on page 23-6.

23.1.3. Configuring the Unit Creation State

This procedure only applies to units in entity-level scenarios.

To configure the aggregation state used when units are created:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Entity Behavior Settings page (Figure 23-3).

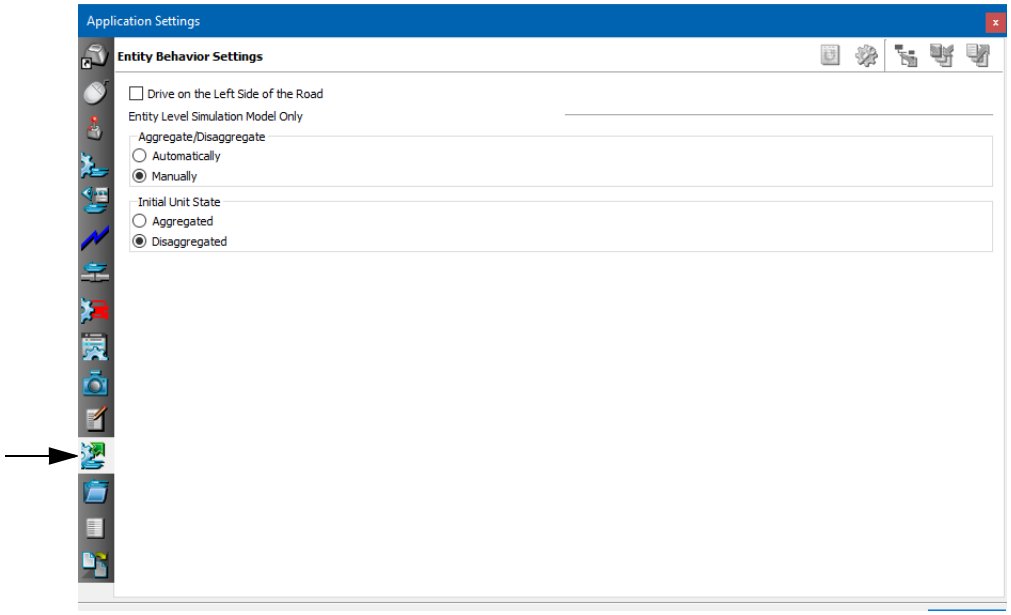


Figure 23-3. Entity Behavior Settings page

3. In the Initial Unit State group box, select the creation option that you want.
4. Click OK.

23.2. Selecting a Unit

➤ To select a unit, select its icon on the terrain, or select it in one of the views on the Objects List Panel.

23.3. Adding Simulation Objects to a Unit

➤ To add a simulation object to a unit, in the Objects List View, select the simulation object you want to add and drag it onto the unit as illustrated in Figure 23-4. The name of the simulation object stays the same, but its echelon ID changes. (You can see the echelon ID in the object's Information dialog box.)

- You can add simulation objects to a unit in an entity-level scenario only if it is in the disaggregated state.
- You can also add simulation objects to a unit with the Superior set data request. For details, please see "Superior," on page 34-42.

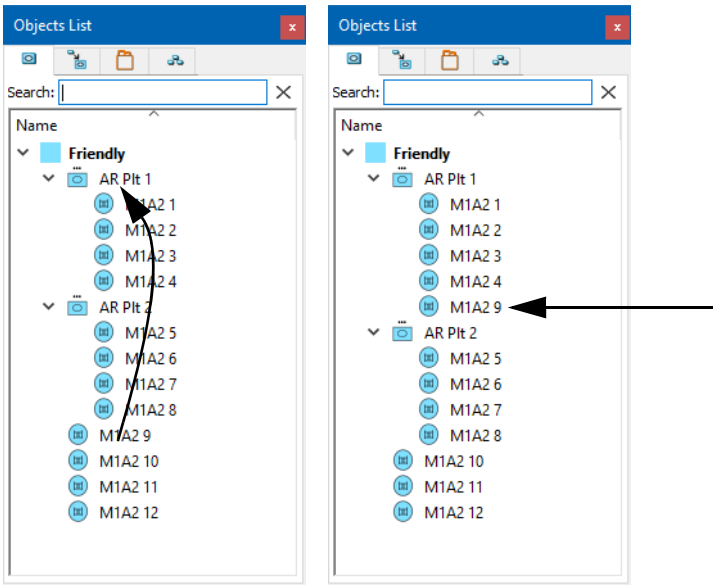


Figure 23-4. Adding a simulation object to a unit

23.4. Removing a Simulation Object from a Unit

➤ To remove a simulation object from a unit, in the Objects List View, drag the simulation object from the unit to the force level in the window, as illustrated in Figure 23-5, or into another unit. The name of the simulation object stays the same, but its echelon ID changes. (You can see the echelon ID in the object's Information dialog box.)

- You can remove simulation objects from a unit in an entity-level scenario only if it is in the disaggregated state.
- You can also remove simulation objects from a unit with the Superior set data request. For details, please see "Superior," on page 34-42.

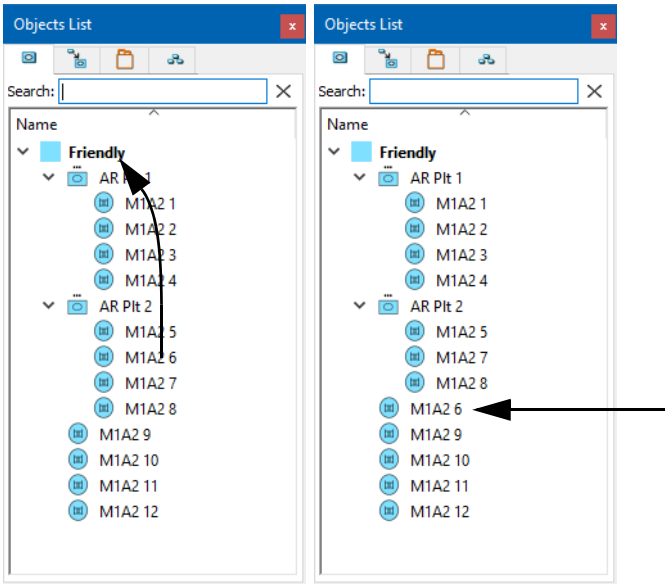


Figure 23-5. Removing a simulation object from a unit

23.5. Units and Unit State in Entity-Level Scenarios

In entity-level scenarios, units can switch back and forth between the aggregated state and the disaggregated state. It is important to understand the differences between these states. The types of units are introduced in Section 21.1, "Units in Entity-Level Scenarios". This section provides additional details about how units function in the two different aggregation states.

The Convoy unit does not support aggregation and disaggregation.

23.5.1. How a Unit's State is Shown

Units publish their aggregation state using the DIS/RPR FOM aggregate state enumeration. The current value for this parameter is displayed on the State Data page in a unit's Information dialog box (Figure 23-6). It can be accessed in the unit's *DtVrfAggregateStateRepository* member.

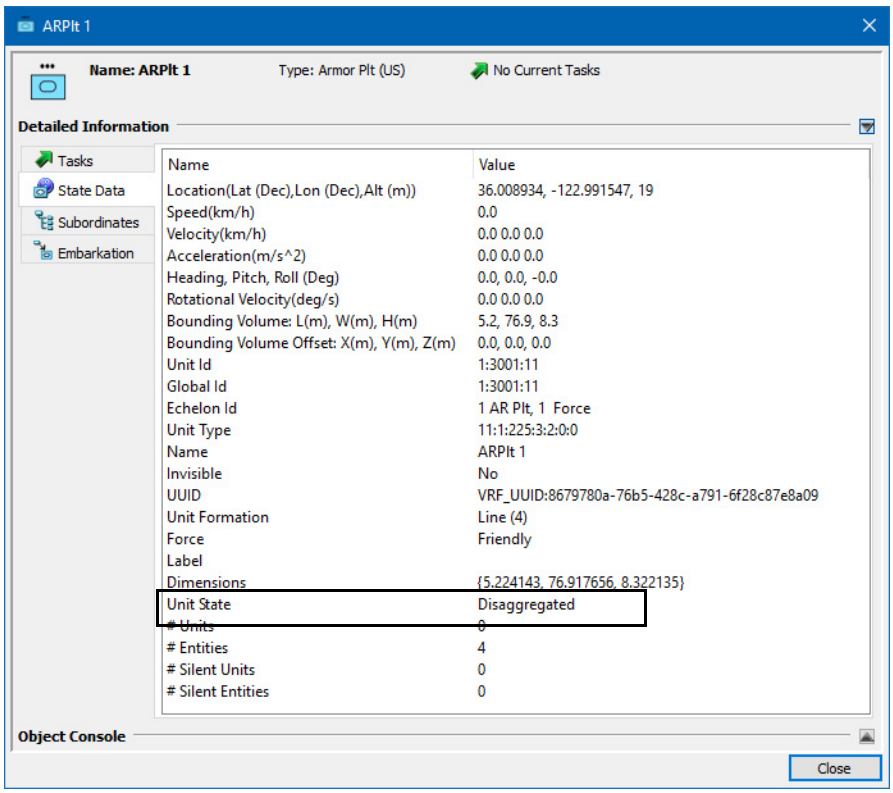


Figure 23-6. State Data page

Aggregated State

When a unit is in the aggregated state, its subordinates do not exist as distinct simulation objects. It does not have any subordinates shown in the Objects List View or on the Subordinates page of its Information dialog box. It does not display ghosted icons for subordinates. You cannot expand or collapse it.

Disaggregated State

When a unit is in the disaggregated state, its subordinates are simulated as distinct simulation objects. (However, it is possible to have a disaggregated unit with no subordinates.) Subordinates are displayed under the unit in the Objects List View and are listed on the Subordinates page of its Information dialog box. You can expand or collapse the unit.

23.5.2. How Changing Aggregation State Affects a Unit

Units can change their aggregation state at runtime. All state information maintained by the unit (for example, the current task it is executing, plan state, current resource levels) is maintained across the transition between aggregated and disaggregated states. This section describes how units handle transitions for the most important activities.

Movement

All movement tasks can be executed in either aggregated or disaggregated state. Units can transition between states at any time, without interrupting movement tasks. For example, suppose a unit in the disaggregated state, is executing a move-to-location task. While it is executing the task, you issue a Unit State set data request, indicating it should switch to the aggregated state. The unit transitions into the aggregated state (subordinates are deleted, and the state associated with them is mapped and stored internally in the unit), and continues moving toward its destination. Similarly, making a transition in the opposite direction while executing a movement task, does not interrupt the execution of that task.

Units preserve their formation across unit state transitions.

Combat

Only disaggregated units can engage in combat. However, a unit remembers any loss of resources or subordinates while it is aggregated. If a subordinate is destroyed while the unit is disaggregated, it is not created again if the unit aggregates and then disaggregates again.

Resources

Resources are tracked and restored across state transitions. However, if a subordinate is destroyed while in the disaggregated state, then all of its resources are lost. They are not redistributed to other subordinates.



When you restore a unit that is in the aggregated state (using Restore), the original configuration of the unit is recreated. If you restore a unit that is in the disaggregated state, then only those subordinates still present in the simulation are restored. Any destroyed subordinates that were previously removed from the simulation as result of a disaggregate-->aggregate-->disaggregate transition are not restored. To fully recover all subordinates, restore the unit while in the aggregated state.

Plans and Tasks

While a unit is disaggregated, its members can be given tasks and plans. However, when the unit aggregates, all tasks and plans are discarded. If the initial unit creation state is set to Aggregated and you create a unit from existing simulation objects, if the simulation objects have plans, the plans are discarded. (For information about settings the initial aggregation state, please see "Configuring the Unit Creation State," on page 23-6.)

23.6. Triggering Unit State Transitions

You can initiate unit state transitions manually or automatically, as follows:

- Manually: You can send a Unit State set data request or use Aggregate and Disaggregate commands on the Objects menu. This immediately forces the unit to change state. As a set data request, state changes can be included in plans.
- Automatically: On the Entity Behavior Settings page of the Application Settings dialog box, you can configure automatic aggregation and disaggregation. In automatic mode, units automatically aggregate or disaggregate according to the following rules:
  - If a unit is within disaggregation range of a hostile entity or unit, the unit is in the disaggregated state. Disaggregation range is the range around a simulation object or unit in which you want to force any nearby aggregated units to disaggregate so you can detect or engage them. This range is configurable for every type of entity and unit in the Simulation Object Editor (Maximum Disaggregation Range). If the disaggregation ranges of two units are not the same size, the larger of the ranges triggers disaggregation of both units.
  - If the unit's formation footprint (projection of its bounding extent in a topographic plane) overlaps a disaggregation area, the unit is in the disaggregated state. (A disaggregation area is a specific type of control object.)
  - If neither of the above conditions is in effect, the unit is in the aggregated state.

23.6.1. Aggregating and Disaggregating Units Manually

To aggregate a disaggregated unit:

1. Select the unit.
2. Choose **Objects** → **Unit** → **Aggregate**.



You can also change a unit's state with the Unit State set data request (which also means that you can change state from within a plan).

To disaggregate an aggregated unit:

1. Select the unit.
2. Choose **Objects** → **Unit** → **Disaggregate**.

23.6.2. Configuring Automatic Aggregation and Disaggregation

To configure automatic aggregation and disaggregation:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Entity Behavior Settings page (Figure 23-3).
3. In the Aggregate/Disaggregate group box, select Automatically or Manually.
4. Click OK.

23.6.3. Using Disaggregation Areas

When an aggregated unit that has automatic aggregation and disaggregation enabled enters a disaggregation area, it disaggregates. When it leaves the area, it aggregates again. A disaggregation area is a control object. You create one just like any other area, as described in Chapter 14, *Creating and Placing Objects*.

23.7. Deleting a Unit

If you delete a unit, all its members get deleted.

To delete a unit:

1. Select the unit.
2. Choose **Objects** → **Delete**.



24. Displaying Units

Many aspects of managing units are the same as for individual entities. This chapter focuses on features that are unique to units.

Selecting Units 24-2
Expanding and Collapsing Units 24-2
Displaying Ghosted Icons 24-4
Displaying Unit Icons and Bounding Volumes 24-8
Specifying the Color Scheme for Unit Bounding Volumes 24-10

24.1. Selecting Units

You can select units in the same ways that you can select entities. (For details about selecting entities, please see "Selecting Simulation Objects, Tactical Graphics, and Props," on page 15-4.) However, to select a unit, you must select the unit icon, not individual members of the unit. If a disaggregated unit is expanded, the unit icon might not be visible. To select the unit, you can collapse it, or you can select the unit in the Objects List View.

24.2. Expanding and Collapsing Units

You can display disaggregated units in expanded or collapsed mode. When a unit is collapsed, it is displayed as one icon. When a unit is expanded, all the members of the unit are visible as individual entities or subordinate units and the unit icon is hidden. Multi-level units can expand and collapse at each level of aggregation. For example, a company could be collapsed to the company level (one icon), the platoon level (an icon for each platoon), or fully expanded.



Aggregate-level scenarios do not support aggregation and disaggregation of units. Units are always disaggregated. Therefore, they can always be expanded and collapsed.

When you Expand or Collapse a unit, you can expand or collapse one level of aggregation or all levels. When you Expand or Expand All, expansion is downward from the current unit and does not affect any sibling units. When you Collapse, contraction is upward in the hierarchy. Collapse All collapses all members of the unit to the topmost level.

Figure 24-1 illustrates these relationships. Assume a company with three platoons, each of which has four vehicles.

- If you Expand the top level unit, it expands to the platoon level. (1)
- If you select Plt C and choose Expand or Expand All, it expands just Plt C. (2)
- If you select the top level unit and expand all, it expands to the leaf level – 12 individual vehicles. (3)
- If the unit is fully expanded and you select one member of Plt C and choose Collapse, just Plt C collapses. (4)
- If you select any member of an expanded unit and select Collapse All, the entire unit collapses. (5)

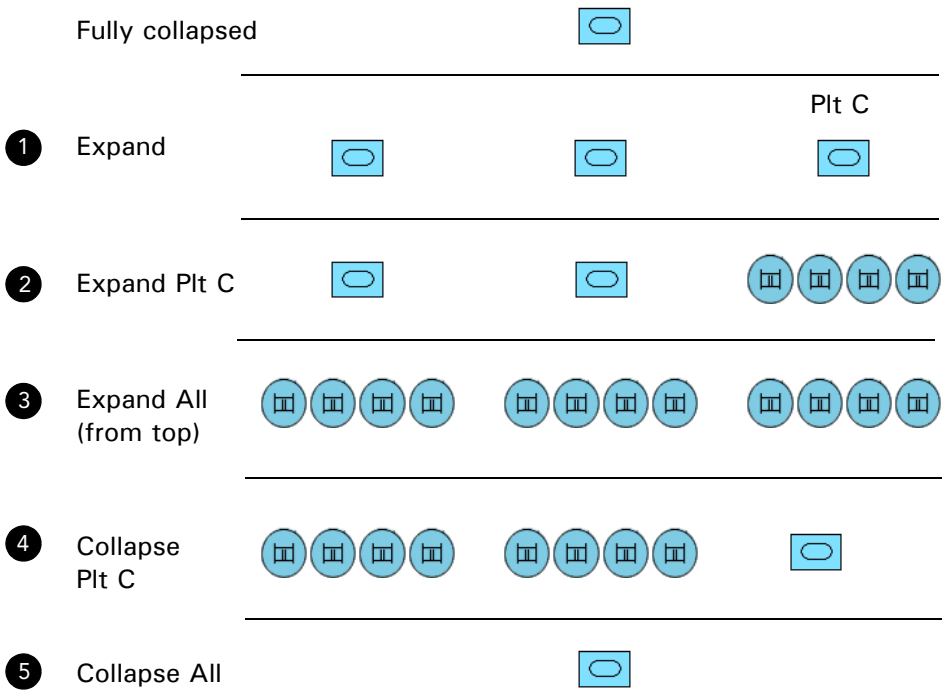


Figure 24-1. Unit expand and collapse

A unit's icon does not indicate whether it is aggregated or disaggregated. However you can easily infer its state from the Objects menu or the Objects List View. The unit commands on the Objects menu are context sensitive. If you select a unit and the Disaggregate command is available, then the unit is in the aggregated state. Similarly, if you view a unit in the Objects List View, if it is aggregated, you cannot expand it in the tree. If it is disaggregated, you can expand it to see its subordinates.

You can also view its state explicitly on the State Data page of the Information dialog box.

To expand a unit:

1. Select the unit icon or its entry in the Objects List View of the Objects List Panel.
2. Choose **Objects** → **Unit** → **Expand**.

To expand a unit and all its subordinate units:

1. Select the unit icon or its entry in the Objects List View of the Objects List Panel.
2. Choose **Objects** → **Unit** → **Expand All**.

To collapse a unit one level:

1. Select any member of the unit.
2. Choose **Objects** → **Unit** → **Collapse**.

To collapse a unit to its root level:

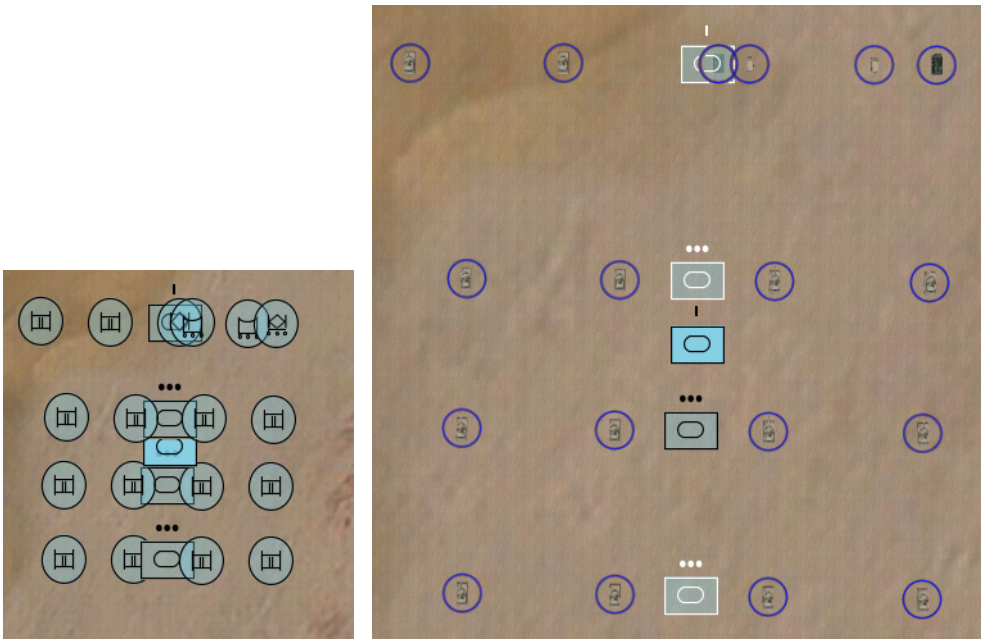
1. Select any member of the unit.
2. Choose **Objects** → **Unit** → **Collapse All**.

24.3. Displaying Ghosted Icons

If you want to select units, you need to collapse them so that you can select the unit icon. However, when you do this, you can no longer see where individual subordinate simulation objects are located and if subordinates are destroyed, you cannot see which ones are destroyed and which are functional. VR-Forces provides the following options for displaying unit icons so that you can work with them in a way that best meets your visualization and simulation object management needs:

- Display ghosted icons for hidden simulation objects.
- Allow ghosted icons to be selected.
- Display all unit members.
- Manually expand and collapse the display.
- Display leaf-level unit members.

In the Plan View observer mode, when you display ghosted units, the unit icon is displayed normally, and the subordinate members are displayed using semi-transparent icons. In 3D observer modes, units that are subordinate to the top level unit are displayed using semi-transparent icons. Leaf-level entities are displayed using the appropriate model for the observer mode (Figure 24-2).



2D 3D

Figure 24-2. Ghosted icons

To configure the display of unit icons:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 24-3).

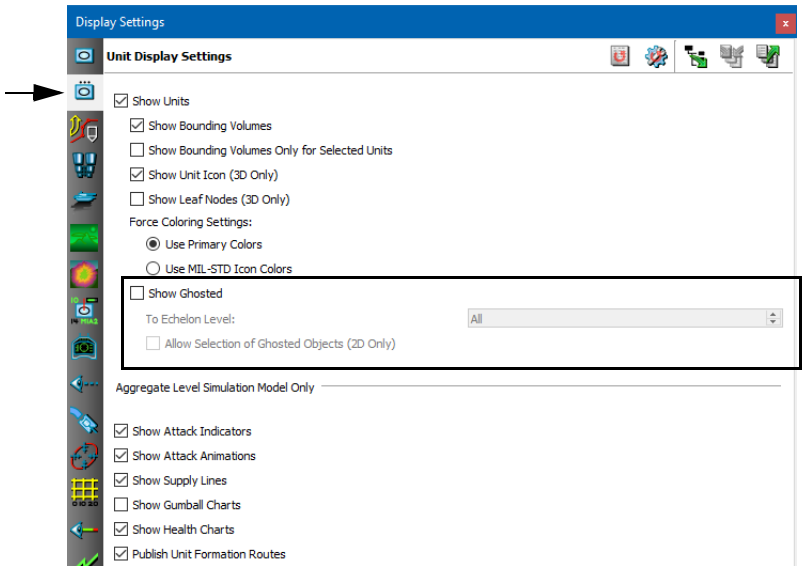


Figure 24-3. Unit Display Settings page

3. To display unit members based on the current level of aggregation only, clear Show Ghosted.
4. To display ghosted icons:
   a. Select Show Ghosted.
   b. In the To Echelon Level box, select the level of aggregation to show ghosted icons for. To show all levels without needing to consider how many levels there are, select All. Figure 24-4 illustrates the effect of changing the echelon level for displaying ghosted icons for the VR-Forces unit hierarchy shown.
   c. To allow selection of ghosted icons (2D view only), select Allow Selection of Ghosted Icons. (In 3D views, you can always select entities at the leaf level.)

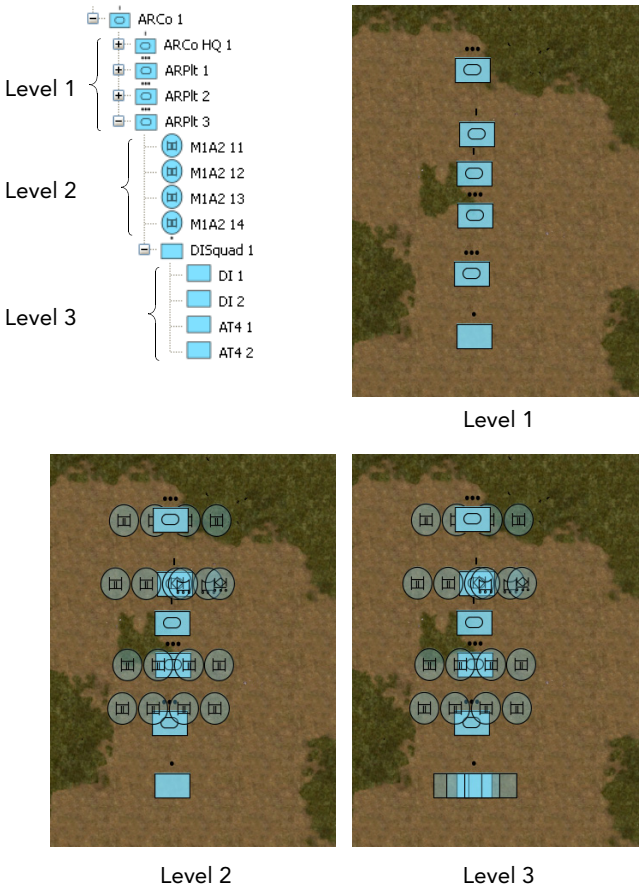


Figure 24-4. Displaying ghosted icons by echelon level

24.4. Displaying Unit Icons and Bounding Volumes

VR-Forces lets you configure when to display unit icons and bounding volumes. Bounding volumes are translucent polygons that show the extents of the unit. This is useful when the individual entity icons or models are not displayed. You can configure icons and bounding volumes as follows:

- Enable and disable the display of units. When disabled, only the 2D icons or 3D models for leaf level unit members are displayed.
- Enable and disable the display of bounding volumes. Bounding volumes are displayed only for collapsed units. You can restrict the display of bounding volumes to the selected units.
- In 3D observer modes, enable or disable the display of unit icons.

Figure 24-5 shows a unit bounding volume in Stealth observer mode, with leaf nodes displayed.

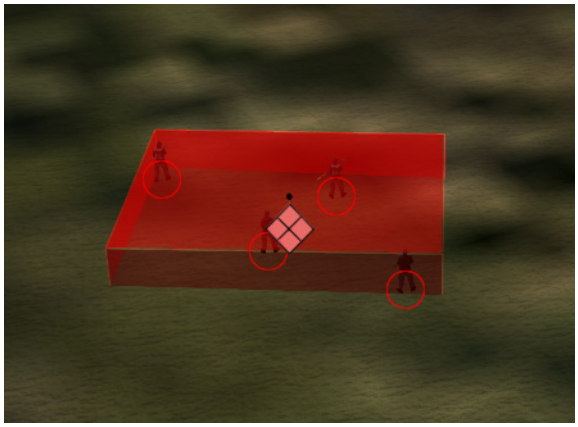


Figure 24-5. 3D bounding volume

Figure 24-6 shows a unit bounding volume in Plan View observer mode, with ghosting enabled.

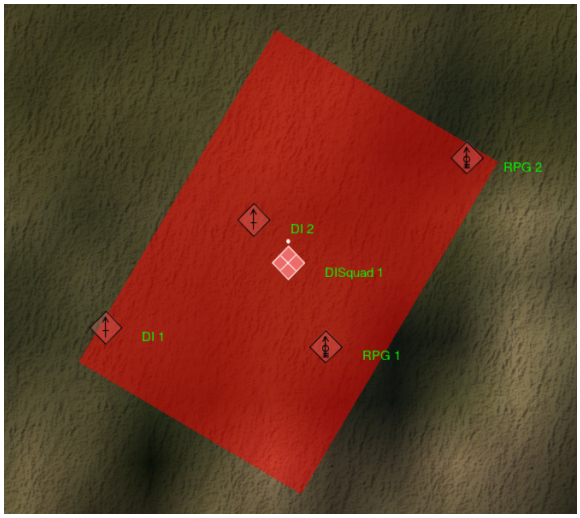


Figure 24-6. 2D bounding volume

You can display unit icons without showing bounding volumes and vice versa.

To configure the display of unit icons and bounding volumes:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 24-3).
3. To enable the display of unit icons, bounding volumes, or both, select Show Units.
4. To display bounding volumes for collapsed units, select Show Bounding Volumes.
5. To restrict the display of bounding volumes to the selected units, select Show Bounding Volumes Only for Selected Units.
6. To display unit icons in 3D observer modes, select Show Aggregate Icon (3D Only).



You can also enable and disable units by choosing **Settings** → **Units** or by selecting the Units button () on the Display Settings Toolbar.

24.4.1. Specifying the Color Scheme for Unit Bounding Volumes

Unit bounding volumes can be displayed using primary colors or MIL-STD colors.

To specify the color scheme to use for unit bounding volumes:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 24-3).
3. Select the Force Coloring Settings option that you want (Use Primary Colors or Use MIL-STD Icon Colors).



V. Tasks

This section describes tasks and scripted tasks and sets.

V.1. Introduction to Tasks

VR-Forces simulation objects can execute a variety of movement and tactical tasks. These tasks can be assigned in plans or through the Task menu. Tasks assigned outside of a plan are called independent tasks.

Some tasks are mutually exclusive with other tasks, for example, a simulation object cannot execute two different movement tasks at the same time. Other tasks, such as Send Text Message, can run concurrently with movement tasks. When you give a task to a simulation object, if the new task is mutually exclusive with the current task, the simulation object immediately stops the current task and begins the new one. If the new task is not mutually exclusive with the current task, the simulation object executes the new task and continues with the previous task. For information about how to configure mutually exclusive tasks, please see "Configuring Task Execution Rules," on page 25-33.

If a simulation object has a plan, when you give it an independent task, it abandons the rest of the plan.

Independent tasks are useful for assigning tasks to simulation objects without the need to edit a plan. You can assign them interactively, so simulation participants can respond immediately to events in a simulation by assigning new tasks to a simulation object as they might have to do in the field. General issues for assigning tasks are described in detail in Chapter 25, *Assigning Tasks*. Specific task procedures are in the succeeding chapters.

Many tasks are implemented as part of the VR-Forces code (C++ tasks) and cannot be added to or changed by end-users. Other tasks, called scripted tasks, are implemented using Lua scripts. You can edit scripted tasks and write new ones. For details, please see "Creating Scripted Tasks and Sets," on page 31-1.

VR-Forces has a special type of scripted task, called a reactive task, that does not get explicitly assigned to simulation objects. Reactive tasks monitor the simulation and run only if certain conditions are met. For information about reactive tasks, please see "Reactive Tasks," on page 25-11.

A third type of scripted task is the background process. Background processes run continuously to handle certain types of ongoing aggregate-level activities, such as consuming or receiving supplies, calculating a unit's footprint, and automatically attacking opposing forces. For information about background processes, please see "Background Processes," on page 32-32.

Figure 24-1 illustrates the many ways that a simulation object can have its task, behavior, and state changed.

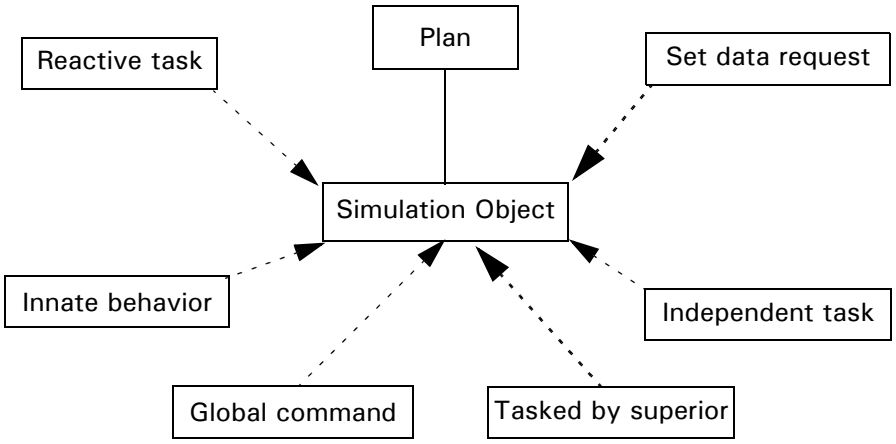


Figure 24-1. Inputs to simulation object actions



25. Assigning Tasks

This chapter provides general details about how to assign tasks to simulation objects. For the procedures for assigning tasks, please see the rest of the chapters in this section.

Assigning Tasks to Simulation Objects 25-3
- Task Procedures 25-3
- C++ Tasks and Scripted Tasks 25-3
- Concurrent Task Execution 25-4
- How Do I Know Which Simulation Objects Can Execute a Task? 25-6
- Escaping the Task Assignment Process 25-6
- Specifying Parameters for Tasks 25-6
- Viewing Task Status 25-8
- Filtering the Object Selection Lists 25-9
- Skipping (Stopping) a Task 25-9

Assigning Tasks to Units 25-10
- Convoy Tasks 25-10
- Independently Tasking Unit Members 25-11

Reactive Tasks 25-11
- Enabling Reactive Tasks 25-13
- Disabling Reactive Tasks 25-14
- Setting the Priority of a Reactive Task 25-15
- Managing Reactive Tasks 25-16
- Canceling a Reactive Task 25-17

Using Behavior Sets to Manage Scripts 25-18
- Creating Behavior Sets 25-19
- Editing Behavior Sets 25-20
- Assigning a Behavior Set to a Force 25-20

Entity Movement On Roads and Off Roads 25-21
- Road Driving Behavior 25-22
- Specifying the Side of the Road to Drive On 25-22

Fixed-Wing Entity Tasks and Behaviors 25-23
Placement of Newly Created Fixed-Wing Entities 25-23
Fly Task Behavior is Different from Move Task Behavior 25-24
Specifying and Maintaining Altitude for Fixed-Wing Entities 25-24
Fixed-Wing Entity Movement on the Ground and in the Air 25-25
How Fixed-Wing Entities Take Off 25-26
How Fixed-Wing Entities Land 25-27
Rotary-Wing Entity Tasks and Behaviors 25-28
Suppressive Fire Tasks 25-30
Configuring Suppression Effects 25-32
Configuring Task Execution Rules 25-33
Configuring Action Categories 25-34

25.1. Assigning Tasks to Simulation Objects

You can assign tasks to a simulation object as part of its plan, as part of a global plan, and independently of any plan. For conceptual information about tasks, please see "Introduction to Tasks," on page V-1. For information about how to add a task to a plan, please see "Adding a Task or Set Data Request to a Plan," on page 37-4.

You cannot assign tasks to non-VR-Forces simulation objects. However, you can use a remote simulation object as a parameter in a task, for example, following a remote simulation object, or targeting a remote simulation object. For more information, please see "Using Non-VR-Forces Simulation Objects in Plans," on page 36-18.

25.1.1. Task Procedures

For the procedures for assigning tasks, please see chapters 26 through 30.

If your installation uses the VR-Forces API to add additional tasks, consult with your software engineers for procedural information. Please see "Fixed-Wing Entity Tasks and Behaviors," on page 25-23 and "Rotary-Wing Entity Tasks and Behaviors," on page 25-28, for additional information about assigning tasks to these simulation object types.

The instructions for creating tasks direct you to use options on the Task menu. However, all the options available on the main menu are also available on context-sensitive menus. A limited number of tasks are available on the Tasks Toolbar and on the Last Selected Object Panel.

25.1.2. C++ Tasks and Scripted Tasks

The VR-Forces application includes a basic set of tasks for movement, weapons fire, and other actions, that were developed using the VR-Forces Toolkit and the C++ API. System integrators often use the C++ API to add additional tasks before delivering VR-Forces to their customers. These tasks are called C++ tasks to distinguish them from another type of task – the scripted task. Scripted tasks, written in the Lua programming language, use the lower-level C++ tasks and the VR-Forces Lua API to build higher-order tasks for simulation objects. You do not need the VR-Forces Toolkit or a developers license to write scripted tasks.

To the VR-Forces end user, scripted tasks, most of which are listed on the Task menu, are indistinguishable from C++ tasks. For information about scripted tasks, please see Chapter 31, *Creating Scripted Tasks and Sets*.

VR-Forces also supports two special categories of scripted tasks called reactive tasks and background processes. These tasks are not on the Task menu. Reactive tasks execute when a condition becomes true. For details, please see "Reactive Tasks," on page 25-11. Background processes run continuously. (For details, please see "Background Process List," on page 30-34.)

25.1.3. Concurrent Task Execution

Some tasks can execute at the same time. Some are mutually exclusive. If you assign a simulation object a task that is mutually exclusive with its current task, the current task is abandoned. If you assign a task that is not mutually exclusive with the current task, it executes the new task while continuing to execute the previous task.



When you assign a mutually exclusive independent task, VR-Forces displays a prompt that asks you to confirm interruption of the current task. You can disable this prompt if you always want a new task to interrupt the current task. For details, please see "Enabling and Disabling the Task Confirmation Prompt," on page 25-5.

C++ tasks are organized into the following groups:

- Weapon
- Movement
- Radio
- Depth control.

All tasks within a group are mutually exclusive among themselves. For example, all movement tasks are mutually exclusive. Simulation objects that are moving can usually execute a task from one of the other groups at the same time. For example, simulation objects can execute the various Send tasks while they are executing movement tasks. When you create a scripted task, you can specify which groups it conflicts with. For information about how to configure mutually exclusive tasks, please see "Configuring Task Execution Rules," on page 25-33. For information about scripted tasks, please see Chapter 31, *Creating Scripted Tasks and Sets*. For information about how to use concurrent tasks in plans, please see "Using Concurrent Tasks in Plans," on page 36-17.

Enabling and Disabling the Task Confirmation Prompt

When you give a simulation object an independent task, if it is executing a task or a plan, VR-Forces informs you that the new task will override the task or plan and asks you to confirm the action. You can enable and disable the confirmation prompt.

To enable or disable the task override confirmation prompt:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 25-1).

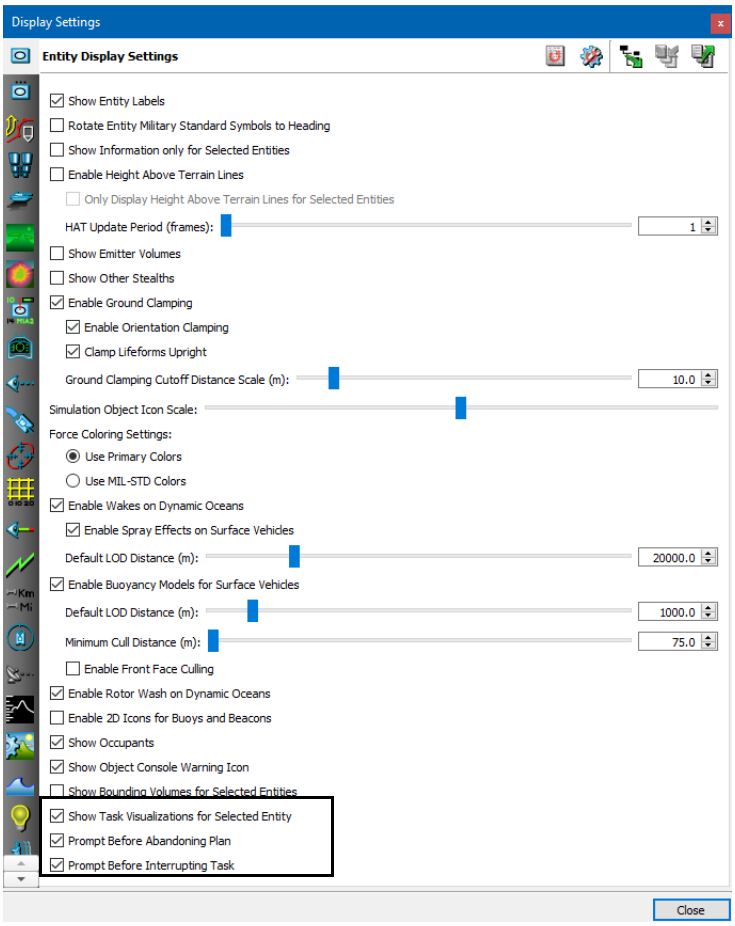


Figure 25-1. Entity Display Settings page

3. To enable or disable the prompt for interrupting a task, select or clear the Prompt Before Interrupting Task check box.
4. To enable or disable the prompt for abandoning a plan, select or clear the Prompt Before Abandoning Plan check box.

25.1.4. How Do I Know Which Simulation Objects Can Execute a Task?

Some VR-Forces tasks apply to almost all simulation objects, such as Radio tasks and some movement tasks. Others are valid only for particular platforms or even particular entity types. Sometimes it is obvious which entity types can execute a task. For example, you would expect that Fixed-Wing Land only applies to fixed-wing entities. Similarly, Raise Periscope applies to submarines. However, for other tasks, it is not always evident which simulation object can execute a task.

Simulation objects can execute tasks for which they have an appropriate controller. They access most task controllers through systems. Therefore, simulation objects can execute the tasks supported by their various movement, weapon, sensor, and other systems. For example, to deploy naval mines, an entity needs a Naval Mine Deployment system.

To see which systems a simulation object has, and thereby infer which tasks it can execute, you can examine the simulation object's systems in the Simulation Object Editor. You can also refer to *VR-Forces Entity Model Catalog*, which lists the systems for each simulation object configured in the Simulation Object Editor. Appendix E, *Systems and System Usage* lists each system and the scripted tasks associated with it.

25.1.5. Escaping the Task Assignment Process

You can escape the task assignment process at any point before you complete it.

➤ To escape the task assignment process, in the task dialog box, click Cancel.

25.1.6. Specifying Parameters for Tasks

VR-Forces lets you specify task parameters in dialog boxes or by selecting them on the terrain. Many tasks require you to select a tactical graphic or a simulation object.

If you select the appropriate object, the selection is reflected in the task dialog box.

If you prefer, you can specify all parameters in the task dialog box. Depending on the zoom level of the terrain and the number of simulation objects and tactical graphics you have created, it may be easier to select an object from a list than to select it from a densely packed group of icons.

Selecting Objects in Multiple List Dialog Boxes

Some task dialog boxes, such as Patrol Between, require you to select multiple objects, each of which is selected from a different list.

You can select the objects in the list window or in the display window. At any given moment, one of the lists is the active list and a selection in the display window is reflected in that list. The active list is indicated by the button next to the Filter list (Figure 25-2).

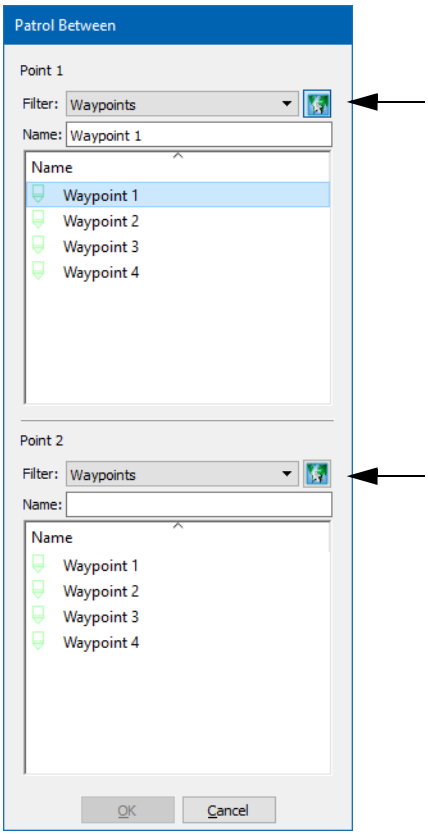


Figure 25-2. Multiple list dialog box

The active list is determined as follows:

- When you first open the dialog box, the first list window is automatically active. If you select an object in the display, it is selected in the first list window. Then the second list window automatically becomes active and the next object you select in the display is selected in the second list window. This sequence continues until all list windows have a selection. Then the last window stays as the active list.
- Selecting an object in a list window makes it active.
- Clicking the Filter button ( ) next to the Filter list makes that list window active. It stays active until you click the button to inactivate it or click in another list window.
- If all lists are inactive (by clicking the Filter button to put it into the unselected state), you can click objects in the display without affecting the selections in the list windows.

25.1.7. Viewing Task Status

The Information dialog box for a simulation object lists its current task on the summary section. It also lists the state of the reactive tasks that are enabled for the simulation object. If a simulation object is executing a task as part of its plan, you can see which task it is executing by opening its Plan window (Figure 25-3). You can also see the current task on the Last Selected Object Panel and the Tasks page of the Information dialog box.

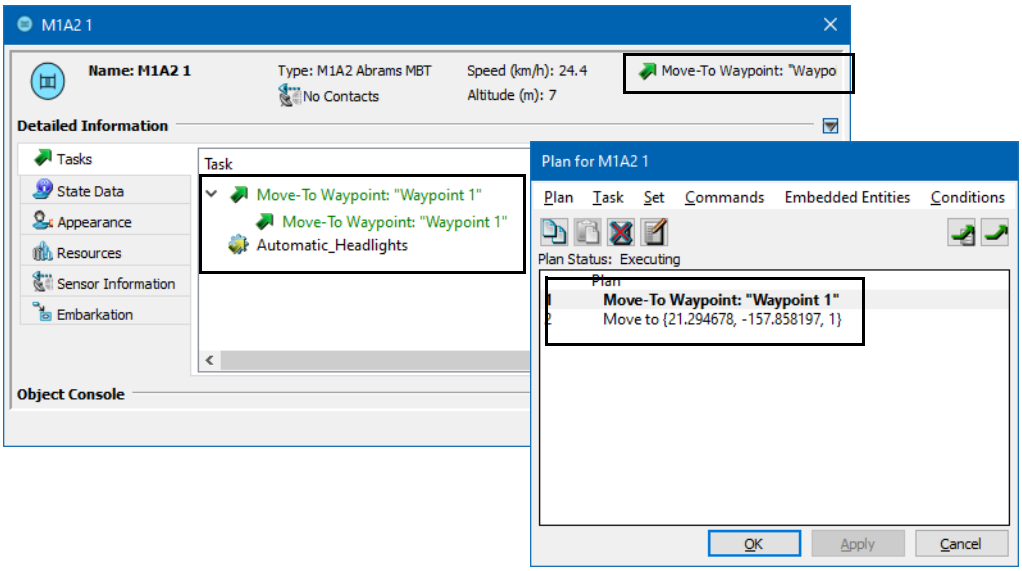


Figure 25-3. Display of simulation object task status

25.1.8. Filtering the Object Selection Lists

If a task requires you to select objects, you can filter the object selection list to make it easier to find the particular objects you need to select (Figure 25-4). If you are selecting simulation objects, for example in a Follow Entity task, you can display all simulation objects, individual simulation objects only, or units only. Within the latter two groups, you can display friendly or opposing simulation objects. In a Move To task, you can filter for waypoints, friendly simulation objects, or opposing simulation objects.

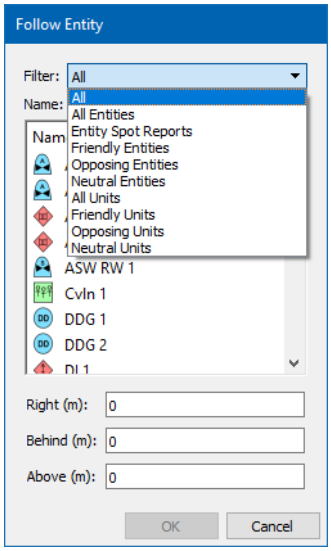


Figure 25-4. Simulation object filter list

➤ To filter a simulation object list, in a task or set data request dialog box, choose the filter type from the list.

25.1.9. Skipping (Stopping) a Task

You can instruct a simulation object to stop executing its current task. If the simulation object is executing a plan, it moves to the next task in its plan. If there are no further tasks, the simulation object waits for further orders.

➤ To order a simulation object to skip a task, choose **Objects** → **Manage Plan** → **Skip Task**.

- Skip Task does not apply to members of a unit that are executing tasks assigned to them by the unit leader.
- You can also stop tasks from the Task Status page of the Information dialog box. For details, please see "Canceling a Reactive Task," on page 25-17.

25.2. Assigning Tasks to Units

➤ To assign a task to a unit, select the unit, then assign it a task as you would any individual entity.

25.2.1. Convoy Tasks

VR-Forces has two tasks for moving in convoys – Convoy Along and Convoy To. These tasks can only be assigned to units made up of ground vehicles. When a unit receives a convoy task, it organizes itself into a column based on the echelon IDs. Then it executes the task.

Once the task begins, the members of the unit try to maintain a consistent distance apart from each other. You can configure this distance by editing the Ground Convoy Movement system. The distance is specified by the **separation-distance** parameter. You can also specify an amount by which the separation distance can vary using the **separation-tolerance** parameter.



If VR-Forces can determine that a convoy task is not appropriate for a given unit, for example a rotary wing unit, the convoy tasks are not available on the Task menu. However, in other cases, such as mixed units or if you aggregate disparate entities as Ground Unit, it is up to you to know whether or not a unit consists only of ground vehicles or contains platforms that do not support convoy tasks.

25.2.2. Independently Tasking Unit Members

Members of a disaggregated unit can have individual plans or be assigned tasks independently of the unit. If a member of a unit is independently tasked, when it completes its task it automatically becomes tasked by its superior. You do not need to give it a Tasked by Superior set data request. You can override this behavior by setting an object's tasked-by-superior-upon-task-complete parameter in the object parameter database to False.

If a Skip Task command is given to an independently tasked subordinate, it remains in the independently tasked state until it either completes a task or it is given a Set Tasked by Superior set data request.

When a simulation object becomes tasked by its superior, it does not execute the task that the unit is currently executing. It receives the next task that is sent out by the unit.



When a disaggregated unit changes to the aggregated state, the individual subordinates cease to exist. Therefore, if a subordinate is executing an independent task, when the unit changes state, the task activity ends immediately.

25.3. Reactive Tasks

Reactive tasks are a special category of scripted tasks. Unlike other tasks (whether scripted or C++), reactive tasks do not begin to execute as soon as they are assigned. They monitor events in a simulation and execute in response to conditions specified in the script. In this way they are similar to collision avoidance, which takes over entity movement in response to an imminent collision, or trigger statements in plans, which get triggered when a condition becomes true.

Because reactive tasks are scripted tasks, you are not restricted to the few conditions available in plans. You have the full VR-Forces Lua API available to script the conditions and the resulting behaviors. Like other scripted tasks, reactive tasks are either scenario-specific or system reactive tasks.

Reactive tasks have the following behaviors and characteristics:

- Reactive tasks can be automatically enabled for a category of simulation object. They can also be individually enabled and disabled.
- When a reactive task is activated, the current task is suspended. When the reactive task concludes, the simulation object resumes the previous task, if possible. There may be cases where the previous task is no longer meaningful due to changes in the scenario.
- Reactive tasks support task concurrency.
- Reactive tasks can be interrupted by other reactive tasks. Precedence among reactive tasks is based on a task's priority. If a reactive task is executing and a reactive task with a higher priority gets triggered, it suspends the current reactive task. When the higher priority task completes, the lower priority task resumes. When it completes, the original task resumes. Multiple nesting of tasks in this way is permitted. Reactive tasks with the same or lower priority do not interrupt the active task.
- Reactive tasks can have parameters, just like any other task. If a reactive task is enabled and the parameters are not set, it may not work as expected. A well-written reactive task should have default values for parameters, but VR-Forces does not enforce this.
- Users can change the priority of a reactive task while a scenario is running. This affects the next activation of the task.
- If a reactive task is interrupted by a non-concurrent independent task assignment from the Task menu or a global plan, the reactive task and all suspended tasks are canceled and the new task is executed. However, it is still possible that the new task could be interrupted by the same or a different reactive task.
- Active reactive tasks can be canceled by a VR-Forces user.
- After a reactive task is complete, it enters the enabled state and can become triggered again.
- The status of a reactive task is displayed on the Task Status page of the Information dialog box, along with the status of the current or suspended task.

The following sections describe how to manage reactive tasks in a scenario. For information about how to write and configure reactive tasks, please see Chapter 32, *Writing Scripts*.

25.3.1. Enabling Reactive Tasks

Like other scripted tasks, when a reactive task is written, it is specified as being valid for one or more simulation object types. It can also be configured to be enabled by default. If a reactive task is not enabled by default, you can enable it for specific simulation objects.



This procedure applies to the currently selected simulation object. If you have multiple Information windows open, selecting a reactive task on the Task Status page of a simulation object does not select the simulation object and does not make it the task affected by this procedure.

To enable a reactive task:

1. Select the simulation object for which you want to enable the task.
2. Choose **Set** → **Reactive Tasks** → **Enable**. The Enable Reactive Task dialog box opens (Figure 25-5). If there are no reactive tasks available, the dialog box states this. This condition occurs if all appropriate reactive tasks are already enabled or there are no reactive tasks for this entity type.

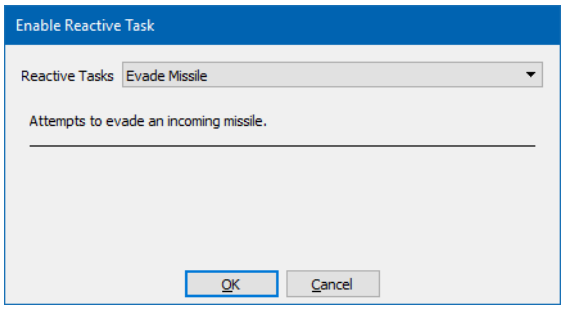


Figure 25-5. Enable Reactive Task dialog box

3. In the Reactive Tasks list, select the task you want to enable. If the task requires parameters, the dialog box redisplays and lists the parameters.
4. If necessary, specify the parameters.
5. Click OK.



You can also enable reactive tasks in the Manage Reactive Tasks dialog box. For details, please see "Managing Reactive Tasks," on page 25-16.

25.3.2. Disabling Reactive Tasks

You can disable reactive tasks on a per-simulation object basis. When you disable a reactive task, any active tasks continue to execute. However, no new instances of the disabled task will be activated.



This procedure applies to the currently selected simulation object. If you have multiple Information windows open, selecting a reactive task on the Task Status page of a simulation object does not select the simulation object and does not make it the task affected by this procedure.

To disable a reactive task for a simulation object:

1. Select the simulation object for which you want to disable a reactive task.
2. Choose **Set** → **Reactive Tasks** → **Disable**. The Disable Reactive Task dialog box opens. If there are no reactive tasks available, the dialog box states this. This condition occurs if all appropriate reactive tasks are already disabled or there are no reactive tasks for this simulation object type.
3. In the Reactive Tasks list, select the task you want to disable.
4. Click OK.



You can also disable reactive tasks in the Manage Reactive Tasks dialog box. For details, please see "Managing Reactive Tasks," on page 25-16.

25.3.3. Setting the Priority of a Reactive Task

When a reactive task is created, it is assigned a priority. The priority determines which reactive tasks it can override or be overridden by. Priorities are expressed as integers, with 1 being the highest priority. You can change the priority of a reactive task on a per-simulation object basis.



This procedure applies to the currently selected simulation object. If you have multiple Information windows open, selecting a reactive task on the Task Status page of a simulation object does not select the simulation object and does not make it the task affected by this procedure.

To change the priority of a reactive task:

1. Select the simulation object for which you want to change a reactive task's priority.
2. Choose **Set** → **Reactive Tasks** → **Priority**. The Reactive Task Priority dialog box opens. If there are no reactive tasks available, the dialog box states this.
3. In the Reactive Tasks list, select the task whose priority you want to change.
4. Select a priority in the box. The lower the number, the higher its priority.
5. Click OK.



You can also change priority in the Manage Reactive Tasks dialog box. For details, please see "Managing Reactive Tasks," on page 25-16.

25.3.4. Managing Reactive Tasks

You can enable, disable, and change priority for all the reactive tasks that apply to a simulation object from the Manage Reactive Task dialog box.

To manage multiple reactive tasks for a simulation object:

1. Select the simulation object for which you want to manage reactive tasks.
2. Choose **Set** → **Reactive Tasks** → **Manage** or choose **Objects** → **Manage Reactive Tasks**. The Manage Reactive Task dialog box opens (Figure 25-6). It lists each reactive task configured for this simulation object type, its status, and its priority. If a task requires parameters, you can set them.

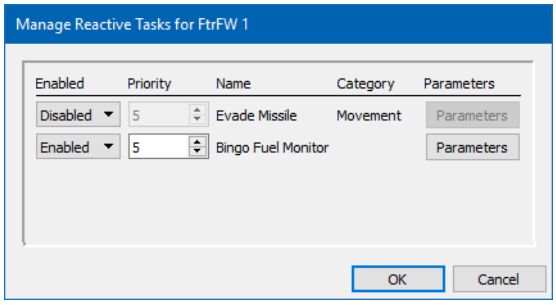


Figure 25-6. Manage Reactive Task dialog box

3. For each task, to enable or disable, select an option from the list in the Enabled column.
4. For each task, change the priority by selecting an option in the box in the Priority column.
5. For each task that has parameters, click Parameters and set its parameters. (A task must be enabled to set its parameters.)
6. Click OK.

25.3.5. Canceling a Reactive Task

You can cancel a reactive task while it is executing. When you cancel a reactive task, the simulation object resumes the suspended task. However, if the condition that caused the reactive task to activate is still true, when you cancel a reactive task, it will usually immediately activate again.



You can also use this procedure to cancel (skip) a regular task.

To cancel a reactive task:

1. Select the simulation object whose task you want to cancel.
2. Choose **Objects** → **Information**, or press **i**. The Information dialog box opens.
3. Select the Task Status page. If a reactive task is active, it has a button in the Cancel column (Figure 25-7).

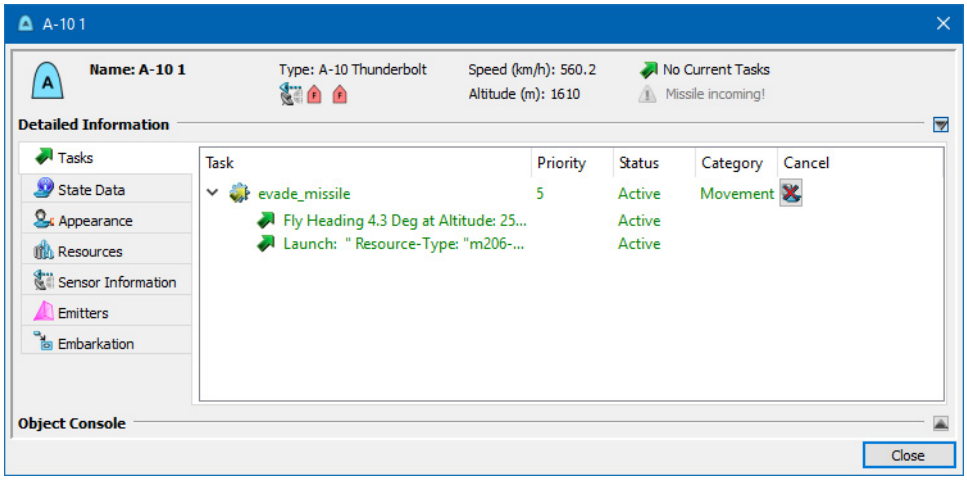


Figure 25-7. Active reactive task

4. Right-click the active task and choose **Cancel Reactive Task**, or click the button in the Cancel column.

25.4. Using Behavior Sets to Manage Scripts

Behavior sets let you organize scripts so that a set of related tasks and set data requests can be enabled or disabled at the force level.

If a script is not assigned to a behavior set, its availability to a simulation object is determined by the following settings:

- The Valid for Entity Types list for the script.
- For tasks specific to systems, configuration of that system on a simulation object.
- For reactive tasks, the Enable/Disable setting in the Manage Reactive Tasks dialog box for the simulation object.

If a script is assigned to a behavior set, then in addition to the requirements in the previous list, it is available to simulation objects only if the behavior set is assigned to a force and only to simulation objects of that force. (You can assign a behavior set to more than one force.)

As an example of how you might use behavior sets, suppose that dismounted infantry react differently to an attack depending on whether they are operating in rural terrain or in urban terrain. You have a set of scripted tasks that you have written for reacting to attack in each of these two situations. If you are not using behavior sets, then to manage which tasks would get used when the simulation objects are under attack, you might use one of the following strategies:

- Tell users not to use the inappropriate tasks.
- Enable and disable the correct set of tasks in the Scripts dialog box before the start of the scenario.
- Enable and disable the tasks on a per simulation object basis in the Manage Reactive Tasks dialog box.

Shifting back and forth between simulations in different terrains might become a management problem.

Using behavior sets, you could:

1. Create a behavior set for rural terrain operations and one for urban operations.
2. Assign the relevant scripts to each behavior set.
3. Before you run a scenario, you would assign the behavior set you want to use to the force.

The tasks in the behavior set would be enabled and those in the unused behavior set would be disabled. There would be no further configuration required.

25.4.1. Creating Behavior Sets

To create a behavior set:

1. Choose **Simulation** → **Behavior Sets**. The Edit Behavior Sets dialog box opens (Figure 25-8).

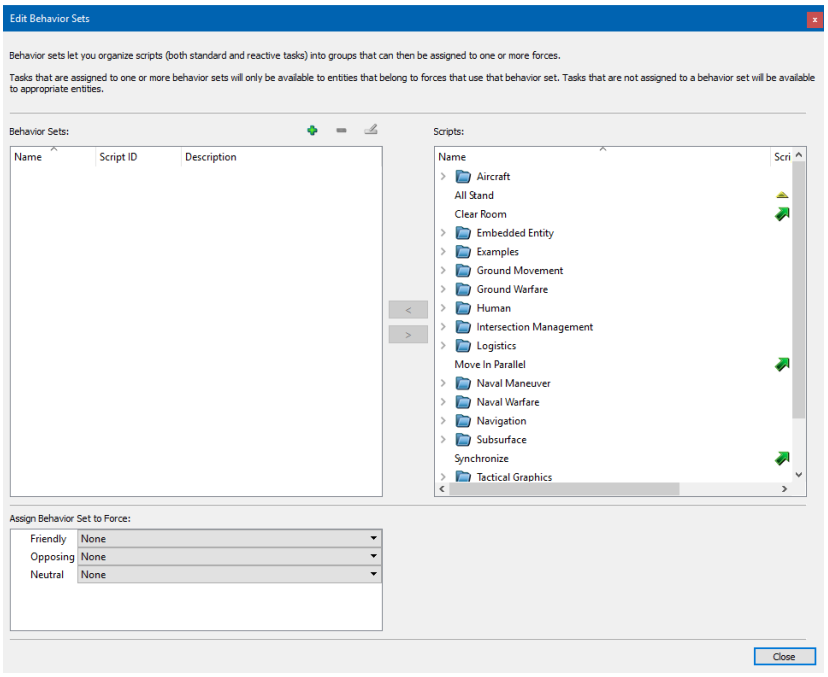


Figure 25-8. Edit Behavior Sets dialog box

2. Click the Add button (✚). The New Behavior Set Name dialog box opens.
3. Type a name for the behavior set.
4. Click OK.
5. In the Scripts list, select the scripts that you want to assign to this behavior set.
6. Click the left-facing arrow. The scripts are added to the behavior set. If you select a folder, all tasks in the folder get added to the behavior set. (You can also assign tasks to behavior sets in the Edit Script dialog box. For details, please see "Creating a New Script," on page 31-6.)
7. Optionally, assign the behavior set to a force.

25.4.2. Editing Behavior Sets

You can edit behavior sets, rename them, and delete them.

To edit behavior sets:

1. Choose **Simulation** → **Behavior Sets**. The Edit Behavior Sets dialog box opens (Figure 25-8).
2. Expand the behavior set that you want to edit.
3. Add and remove scripts by selecting them and clicking the appropriate arrow button.

➤ To rename a behavior set, select it in the Behavior Sets list, click the Rename button (), and type a new name.

➤ To delete a behavior set, select it in the Behavior Sets list and click the Delete button ().

25.4.3. Assigning a Behavior Set to a Force

You can only assign one behavior set to a force at any given time.

To assign a behavior set to a force:

1. Choose **Simulation** → **Behavior Sets**. The Edit Behavior Sets dialog box opens (Figure 25-8).
2. In the Assign Behavior Set to Force window, click the list for the force to which you want to assign a behavior set.
3. Select the behavior set that you want to assign. If you do not want any behavior sets assigned, select the blank line in the list.

25.5. Entity Movement On Roads and Off Roads

Some terrains have vector road networks. Ground vehicles can use these networks to move precisely through the terrain. The tasks for moving on roads are distinct from the tasks for moving without respect to road networks. (For a description of how entities move when they use road driving, please see "Road Driving Behavior," on page 25-22.) If you want entities to move both on and off roads, you need to understand how these tasks interact.

The road driving tasks are Move to Waypoint (Plan Along Roads) and Move to Location (Plan Along Roads). The generic movement tasks are Move to Location and Move to Waypoint.

! Road driving tasks are valid only for ground vehicles. If you give a road driving task (including Move To (Plan Along Roads), Treat Route as Road, and Use Roads in pattern of life default plans) to any other type of entity, it will fail.

When you give a simulation object a generic Move To task, it moves directly towards its destination. If navigation data is available, it plans a path using that data. If no navigation data is available, the entity avoids obstacles, but otherwise does not pay attention to the terrain except to the extent that it cannot move on certain soil types or terrains that are too steep.

When you give an entity a Plan Along Roads task, it looks for a road network. If it finds one:

- It moves to the nearest point on the road.
- It moves along the road network to the nearest point to the final destination.
- It moves to the final destination.

---

i The road driving feature in VR-Forces requires that the road vectors making the network be connected at their edges. If your terrain's data does not have accurate positions for road ends, which causes them to be disconnected, VR-Forces may not be able to plan a path along those roads.

25.5.1. Road Driving Behavior

The Move To (Plan Along Roads) tasks use road driving behaviors intended to provide realistic behavior of multiple cars on a road network. When an entity is given one of these tasks:

- It clamps to the road and does not deviate from it.
- It goes around corners accurately.
- It ignores any obstacles that are near to the road and which would ordinarily trigger obstacle avoidance. The vehicle only responds to obstacles that block the road.
- If a vehicle is blocking its progress and there is an adjacent lane in the road that is clear, the entity passes the blocking vehicle.

In the Move Along Route task, you can specify that VR-Forces treat the route as a road. The entity would then use the road driving system to follow the route.

25.5.2. Specifying the Side of the Road to Drive On

By default, entities that use road driving behavior drive on the right side of the road. You can specify that entities drive on the left side of the road. For this setting to take effect, you must make the change before you load a scenario or before you rewind it. If you change the setting after you rewind a scenario or while the scenario is running, the behavior does not change until the next time you rewind the scenario.

To specify the side of the road to drive on:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Entity Behavior Settings page (Figure 25-9).

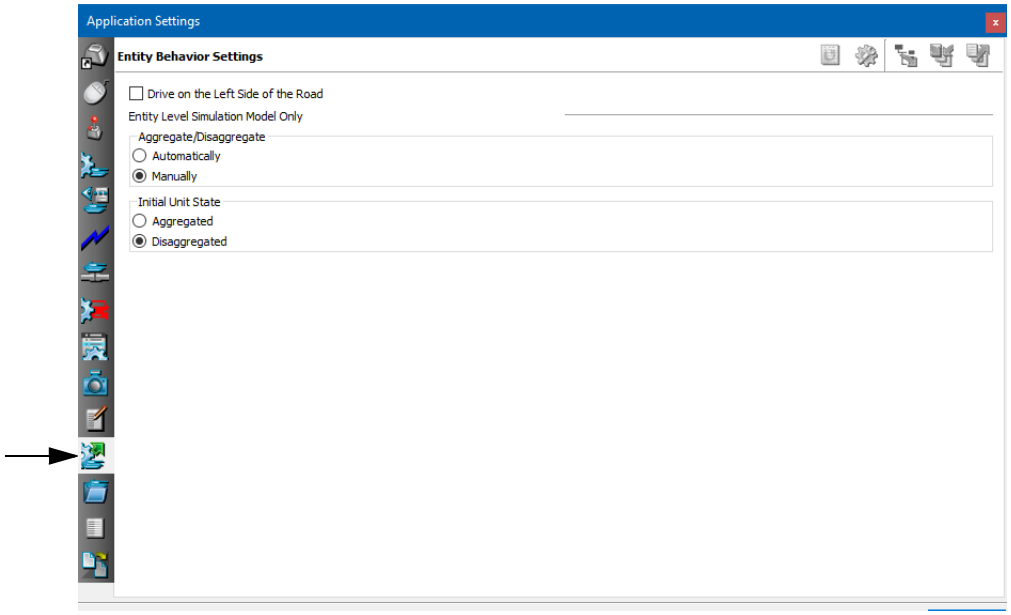


Figure 25-9. Entity Behavior Settings page

3. Select or clear the Drive on the Left Side of the Road check box.
4. Click Close.

25.6. Fixed-Wing Entity Tasks and Behaviors

This section describes the behaviors of fixed-wing entities supplied with VR-Forces and the issues you should keep in mind when assigning tasks and set data requests.

25.6.1. Placement of Newly Created Fixed-Wing Entities

When you create a fixed-wing entity, it is placed on the ground. If you want a fixed-wing entity to start a scenario in the air, you must set its altitude at the beginning of the scenario. It will loiter at the specified altitude.

25.6.2. Fly Task Behavior is Different from Move Task Behavior

Fixed-wing entities can move in response to Fly tasks (Fly Altitude, Fly Heading, and so on) and Move tasks (Move to Waypoint, Move Along Route, and so on). There are important differences between how fixed-wing entities respond to the different classes of movement tasks. Fixed-wing entities can move on the ground (taxi), in which case they move like ground vehicles.

For Move and Patrol tasks, if a fixed-wing entity is in the air, when it arrives at a point or the end point of a route, it loiters.

Fly tasks are only for entities that are in the air. You cannot use them to cause an entity to take off or land. For Fly tasks:

- Entities move to a new altitude at the specified heading, at the specified climb or descent rate.
- When they reach the new altitude, they continue flying at their current heading.

Given these differences, if you want to change heading, altitude, or both for an entity that is in the air and not executing a Move or Patrol task, use a Fly task. These tasks are not designed to have an entity move to a particular location or along a route. To have an entity move to a particular location or along a route, use a Move or Patrol task. In most cases, Fly Altitude will be preferable to Move to Altitude.

25.6.3. Specifying and Maintaining Altitude for Fixed-Wing Entities

You can explicitly change a fixed-wing entity's altitude in the following ways:

- The Altitude set data request moves an entity immediately to the specified altitude.
- The Move to Altitude task causes an entity to spiral to the specified altitude at the same coordinates that it had when it received the task.
- The Fly Altitude and Fly Heading and Altitude tasks cause the entity to move smoothly to the new altitude while continuing to fly at its current or newly specified heading.

An entity's altitude changes implicitly if it is given a Move or Patrol task in which the altitude of the target point or vertex is different from the entity's current altitude. When moving along a route, the entity changes altitude as it flies along a line between vertices.



When you assign a task that includes a location and you specify the location by clicking on the terrain, the default altitude is zero. If you do not specify a non-zero altitude in the task dialog box, VR-Forces uses the altitude of the entity at the time the task is assigned. This prevents you from inadvertently specifying a location that would cause the entity to crash.

25.6.4. Fixed-Wing Entity Movement on the Ground and in the Air

Fixed-wing aircraft support takeoff and landing. Since they can move on the ground and in the air, you need to understand their behavior in these different environments. Table 25-1 and Table 25-2 describe how fixed-wing entities respond to Move To, Move Along Route, and Patrol Route tasks depending on their current location. The rest of this section provides additional details about take-off and landing.

Table 25-1: Fixed-wing entity Move To behavior

If entity is on/in the:	and waypoint is on/in the:	the entity:
Ground	Ground	taxis to point.
Ground	Air	takes off towards the point.
Air	Air	flies to point.
Air	Ground	crashes.

Table 25-2: Fixed-wing entity route-following behavior

If entity is on/in the:	and next vertex is on/in the:	the entity:
Ground	Ground	taxis to vertex.
Ground	Air	takes off toward the vertex.
Air	Air	flies to the vertex.
Air	Ground	tries to land at the vertex.

The aircraft looks ahead a configured distance in its current flight direction. If any terrain blocks the path between it and its destination vertex on the route, it ascends to avoid the terrain. When no blocking terrain is found, it resumes flying toward the destination vertex directly.

25.6.5. How Fixed-Wing Entities Take Off

You can have fixed-wing aircraft take off using the Fixed Wing Takeoff task (for details, please see "Fixed Wing Takeoff," on page 26-7) or by using innate behavior. This section describes take-off using innate behavior.

Fixed-wing entities take off by moving to a waypoint that is above ground level or by following a route that has a vertex above ground level.

When you task a fixed-wing entity to move to a waypoint that is above ground level, the entity orients itself directly towards that point and accelerates in the direction of the point. When it reaches its minimum lift speed, it takes off and rises to the altitude of the point it is moving to. You can control the distance required to take off by setting the **max-thrust parameter**.

When a fixed-wing entity takes off, it does not take into account the soil type or terrain, except that if the slope of the terrain along which the entity must move to take off is greater than the **max-slope** parameter, the entity halts.

Fixed-wing entities do not know anything about runways that may be part of a terrain database. When you create a waypoint towards which an entity will take off, or create a route along which an entity will take off, you are responsible for ensuring that the entity is properly oriented towards the runway.

If an entity takes off as part of route following, you can define a runway with two points on the ground. If an entity detects that the point in the route after the one it is moving towards is in the air, it uses the current point as the take-off direction, but takes off as soon as possible and flies to the point in the air. Figure 25-10 illustrates this behavior.

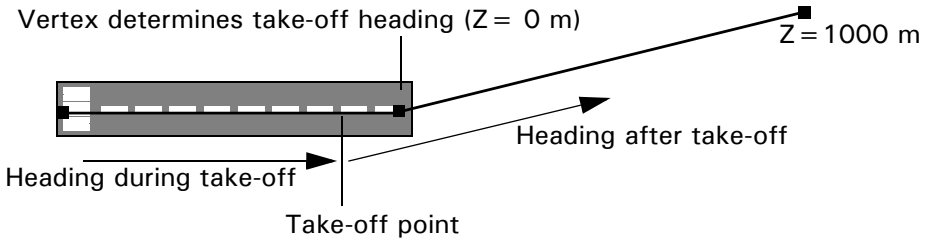


Figure 25-10. Defining a runway for take-off

25.6.6. How Fixed-Wing Entities Land

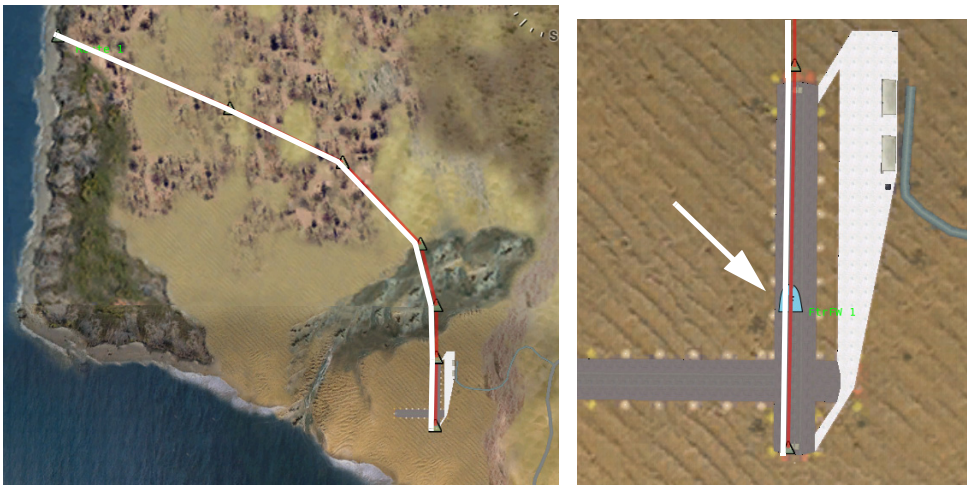
You can have aircraft land using the Fixed Wing Land task (please see "Fixed-Wing Land," on page 26-7), or as described in this section. The underlying behavior of the Fixed Wing Land task is the same as in this section. Using the Fixed Wing Land task automates the process of setting up the approach route.

A fixed-wing entity can land only in the context of following or patrolling a route. When a fixed-wing entity follows a route and determines that the next vertex in the route is on the ground, it decelerates as it reduces altitude towards that vertex. If it does not have enough distance from the previous vertex to decelerate sufficiently, it may crash. To ensure adequate deceleration, you may want to create a trigger that sets the entity's speed when it approaches the runway.



Any time an entity is within the on-ground-tolerance distance of the terrain and it is moving within 10 percent of its minimum lift speed, it will try to land. The on-ground-tolerance for most fixed-wing entities is 10 meters.

To orient an entity for landing, you may have to create several vertices prior to the runway to get the entity properly lined up. Figure 25-10 shows two different routes ending at an airport (the routes have been edited to improve visibility). The first route has vertices that line up the entity with the runway. The second route curves in towards the runway, so the entity does not approach the runway in a straight line. Therefore, as it reaches the landing point, it is not lined up with the runway and it lands on the taxiway.



Entity lands on runway



Entity lands to left of runway

Figure 25-11. Landing fixed-wing entities

25.7. Rotary-Wing Entity Tasks and Behaviors

This section describes the behaviors of rotary-wing entities supplied with VR-Forces and the issues you should keep in mind when assigning tasks and set data requests.

Rotary-wing entities get created on the ground unless you specify an altitude at creation. (For details about specifying an altitude when you create an air platform, please see "Specifying an Object's Altitude," on page 14-18.)

If a rotary-wing entity is airborne and it is not assigned a task, it hovers at the specified location and altitude. When it completes a task, it hovers in the location where it completed the task.

The terrain-check parameter determines how a rotary-wing entity interacts with the terrain. If terrain-check is False, a rotary-wing entity ignores the terrain. It could appear to fly below ground. However, when terrain-check is set to False, VR-Forces does not have to check ground points, so performance may improve.

When terrain-check is True, rotary-wing entities interact with the terrain. If a rotary-wing entity approaches a ground point at three meters per second or less, it lands. If it approaches the ground at a greater speed, it crashes.



When you assign a task that includes a location and you specify the location by clicking on the terrain, the default altitude is zero. If you do not specify a non-zero altitude in the task dialog box, VR-Forces uses the altitude of the entity at the time the task is assigned. This prevents you from inadvertently specifying a location that would cause the entity to crash.

Helicopters have a basic terrain avoidance algorithm that helps prevent crashing if they are sent to waypoints or follow routes that do not have an altitude. The terrain avoidance logic maintains a distance above the ground; it does not look ahead. So, a helicopter could crash if the terrain rises suddenly (Figure 25-12). You can view the terrain profile of the route to check for intersections with the terrain. (For details, please see "Displaying a Line's Terrain Profile," on page 56-3.)

Rotary-wing aircraft take wind resistance and the weight of the aircraft, including the weight of any embarked objects, into account when calculating fuel consumption.

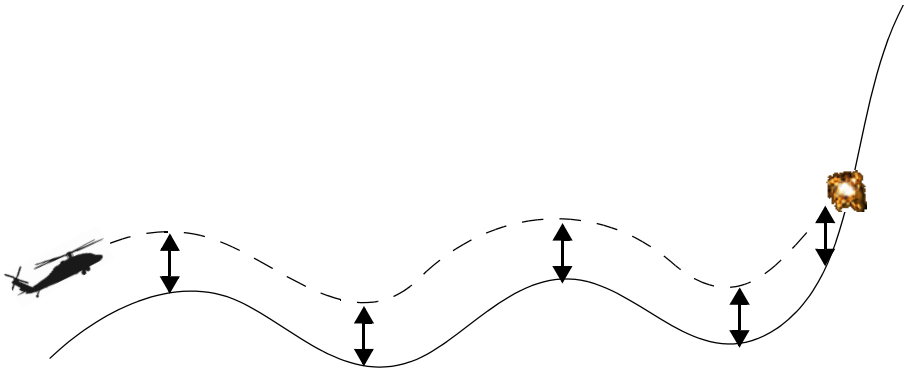


Figure 25-12. Helicopter terrain avoidance

25.8. Suppressive Fire Tasks

VR-Forces has two suppressive fire tasks.

- The Provide Suppressive Fire task tells an entity to fire at a point, along a line, or in an area, one meter above the terrain.
- The Provide Suppressive Fire on Location task tells an entity to fire at a location, one meter above the location.

Fire is rapid at first, then sustained for a specified period of time and maximum rounds to fire, or until the entity runs out of ammunition. The rapid and sustained rates of fire are configured in the weapon's system definition file. If you want to suppress a floor of a building, set the altitude at or slightly above the altitude of the floor of the building. Suppressive fire is executed even if line of sight is blocked near the target location, so you can place the target in a building and get fire on the wall in front of it.

Suppressive fire insults on an entity are accumulated as part of the internal state of the entity. This suppression value also continually decays over time. Therefore large insults, or many small insults in a short time, raises the suppression value, but the value naturally drops back down to its starting level.

An entity is considered suppressed when its suppression value rises above a certain threshold. The entity is no longer suppressed when its decaying suppression value falls below a different, lower threshold. This allows the model to produce a delay in suppression recovery and at the same time continue handling new insults.

The model represents three suppression levels (in addition to "not suppressed") so that behavior can be changed for more severe suppression. The number of suppression levels was chosen arbitrarily. Higher levels of suppression are reached when the suppression value passes higher thresholds. The State Data page in the Information dialog box displays the suppression level of the entity, a value between 0 and 3.

Suppressive fire causes humans that are near the detonation locations to temporarily stop shooting and lose some ability to sense entities. When an entity is suppressed:

- Its sensors may have their sensitivity reduced.
- Its guns may be prevented from firing.
- It may be forced into a crouching posture.
- It may be forced to stop moving and go prone.

The default configuration for humans is:

- Visual sensor sensitivity is reduced to 0.
- Guns are disabled.
- Level 1 suppression forces a crouch.
- Level 2 and 3 suppression stops movement and makes the entity go prone.

The Lifeform Suppression system enables two reactive behaviors. The first one makes the human adopt a crouching posture when it is suppressed. This does not interrupt other movement. The second reaction makes the entity go prone when it is suppressed to level 2 or 3. This interrupts any ongoing movement task. Both reactions restore the original posture when the reaction ends.

Several vehicles are configured so that they cannot fire their machine guns if they are suppressed.

An entity is susceptible to suppression if it has a suppression system. Lifeforms can have a Lifeform Suppression system. Vehicles can have a Vehicle Exposed Crew Suppression system. For details about configuring suppression, please see "Configuring Suppression Effects," on page 25-32.

25.8.1. Configuring Suppression Effects

Suppression effects are configured by parameters in several components.

The suppression system contains a suppression actuator with several parameters to control its behavior.

- suppression-insult-level. The level of insult necessary to suppress the entity, that is, reach the level 1 threshold. This value should be set appropriately for the amount of insult caused by munitions, as defined in the damage-model portion of this descriptor. You can set this parameter in the Simulation Object Editor.
- suppression-recovery-time. The time required to recover from the level 1 suppression threshold to a not-suppressed state. This same time is required to recover to level 2 from level 3, and to level 1 from level 2. You can set this parameter in the Simulation Object Editor.
- passing-rounds-suppress. Determines whether rounds passing by the entity, but not necessarily detonating near it, are considered when detecting insults. You can turn off this effect to save computation time. Default: true.
- max-detonation-distance-for-passing-rounds. The maximum distance that the actuator looks for detonations. A line of fire that passes near the entity may detonate some distance beyond the entity. To find such lines, the actuator must look at distant detonations. This may result in the actuator processing many detonations. (The detonation manager makes a distance test for each entity for each detonation, so not much computation is saved by reducing this parameter value.)
- hostile-fire-only. If true, the actuator does not consider detonations and passing rounds as insults if they are from a friendly force. Default: true. While the default may be unrealistic in the case of HE shells landing near the entity, it avoids unintended suppression when nearby friendly entities are firing at the enemy.

The actuator descriptor includes a damage-model section that is just like the damage-model in a damage system. For each munition there are two parameters:

- suppression-detonation-insult. The maximum insult that this munition can cause, if it detonates at the entity location or passed directly over the entity.
- suppression-detonation-range. The maximum range at which the munition can have effect. The insult from the munition decreases linearly until it is 0 at the maximum range.

In ballistic gun controllers, disabled-by-suppression indicates whether the gun is susceptible to suppression. If true, the gun is not allowed to fire while the entity is suppressed. This is true for the handheld weapon systems, but it can also be selectively set for guns on vehicles. For example, tank machine guns are often operated by an exposed crew member, and could be suppressed by small arms fire even if the tank is otherwise unaffected by such fire. The handheld systems have this parameter set True, while other machine guns use a variable and so the parameter can be set in the Simulation Object Editor.

In sensor components, suppression-degrades-sensor-performance-by indicates how they are affected by suppression. It is a value from 0 - 1.0. A value of 0 means that the sensor is not degraded at all by suppression, while a value of 1.0 means that sensitivity is reduced to 0 by suppression. The value can be set in the Simulation Object Editor.

25.9. Configuring Task Execution Rules

Each simulation object OPE file contains a parameter called task-execution-rules that specifies a task execution rule configuration file. (The default rules file is *./data/simulationModelSets/<model_set>/vrfSim/taskRules/default-task-rules.tsk*.) The task execution rule file specifies whether or not tasks can be executed concurrently, and if so, which tasks are mutually exclusive.

The task execution rules file has the following parameters:

- all-tasks-exclusive. This parameter enables (False) or disables (True) use of concurrent execution of tasks. When True, the Task Manager executes only one task at a time, and the mutually-exclusive-task-groups parameter is ignored.
- mutually-exclusive-task-groups. This parameter is a list of groups. Each group contains a list of tasks that cannot be executed at the same time. Any time a task from a particular group is started by the Task Manager, any currently running task that is part of the same group is interrupted. Tasks that are part of other groups, or not part of any group, are allowed to continue running.

The task names that are listed in the mutually-exclusive-task-groups parameter are the names that are registered with the task factory when a task is added to the system. Groups can contain any number of tasks, and there can be any number of groups.

Tasks are always considered to be mutually exclusive with themselves. For that reason, it is impossible to execute two tasks of the same type at the same time on a single simulation object.



default-task-rules.tsk configures the tasks that come with VR-Forces and which were developed in C++. It does not include scripted tasks. Concurrency for scripted tasks gets set when they are created.

25.9.1. Configuring Action Categories

In scripted tasks, action categories serve the same function as task rules for enforcing concurrency. Action categories are configured in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/actionCategories.tsk*. An action category is specified with the following entry:

```
<item>
  <first>Movement</first>
  <second>
    <first>Movement</first>
    <second>1</second>
  </second>
</item>
```

If the value in the <first> element matches a group in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/default-task-rules.tsk*, scripted tasks that use that action category are unioned with the tasks specified in *default-task-rules.tsk* to form the complete set of mutually exclusive tasks.



If you edit *actionCategories.tsk*, be sure to update the <count> element to reflect additions or deletions.



26. Movement Tasks for Entity-Level Scenarios

This chapter describes the movement tasks supplied with VR-Forces that apply to entity-level scenarios. Some of these tasks are also used in aggregate-level scenarios. Chapter 28, *Human and Crowd Tasks* describes some movement tasks that are specific to human characters.

Air Drop 26-3
Animated Movement 26-4
Come to Stop 26-4
Convoy Along 26-5
Convoy To 26-5
Fast Rope Insertion 26-6
Fixed-Wing Land 26-7
Fixed Wing Takeoff 26-7
Fly Altitude 26-8
Fly Heading 26-8
Fly Heading and Altitude 26-9
Follow Entity 26-10
    How Fixed-Wing Entities Follow Entities 26-10
Generate Air Traffic From Flight Plans 26-11
Land at Airport Near Location 26-11
Move Along Route 26-12
Move Into Formation 26-13
Move to Altitude 26-14
Move to Depth 26-14
Move to Location 26-15
    Adding Multiple Move to Location Tasks to a Plan 26-16
Move to Location (Plan Along Roads) 26-16
Move to Location (Plan Path) 26-17

Move to Waypoint 26-18
Move to Waypoint (Plan Along Roads) 26-19
Move to Waypoint (Plan Path) 26-20
Orbit 26-21
Orbit Object 26-21
Parallel Pattern 26-22
Patrol Route 26-23
Patrol Between 26-23
Pattern Hold (Location) 26-24
Pattern Hold (Waypoint) 26-24
Rotary Wing Airlift Cargo to Location 26-26
Rotary Wing Land 26-26
Rotary Wing Retrieve Cargo 26-27
Rotary Wing Unload Cargo 26-27
Sail Heading 26-28
Sweep Naval Mines 26-28
Turn to Heading 26-29
Zig Zag to Heading 26-29

26.1. Air Drop

The Air Drop task causes an aircraft to disembark selected paratrooper entities and have them parachute towards a target simulation object or location. The paratrooper tries to land as close to the target as it can given the altitude at which it disembarks and the distance to the target. If it cannot reach the target before it lands, the entity walks towards the target after it lands.



While an entity is parachuting, if you double-click it to try to move it, it snaps to the terrain.

To assign an Air Drop task:

1. Select an aircraft that has paratroopers embarked on it.
2. Choose **Task** → **Movement** → **Air Drop**. The Air Drop dialog box opens.
3. Specify the time to wait between each entity drop, in days, hours, minute, and seconds.
4. Select the entities to drop. You can select multiple entities.
5. Specify a target. It can be a simulation object or a location. If you specify both, the object is used. (After you select the entities to drop, the Target Object section automatically has focus. If you want to set a location, be sure to select the filter button first. (For details about controlling focus in dialog boxes, please see "Filtering the Object Selection Lists," on page 25-9.)
6. Specify the altitude at which to deploy the parachute. If the aircraft is below the deployment altitude, the parachute deploys as soon as the entity disembarks.
7. Click OK.

26.2. Animated Movement

The Animated Movement task lets you assign a scripted behavior to an entity. This task is a generalized version of firing ballistic missiles, which is described in "Ballistic Missiles," on page 10-8. To use this task, you must first create a file that describes the movements that you want an entity to make. Animated movement files can be in comma-separated values (CSV) format or 3DS ASCII Export (ASE) format. You configure animated movement tasks in the Simulation Object Editor. For details, please see "Configuring Animated Movement," on page 69-27.

To assign an animated movement task:

1. Select the entity to task.
2. Choose **Task** → **Movement** → **Animated Movement**. The Animated Movement dialog box opens (Figure 26-1).

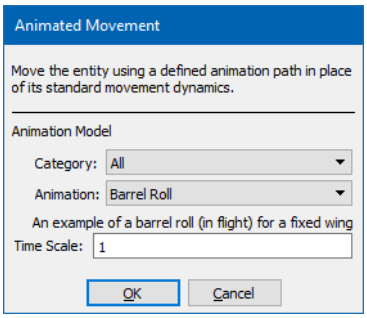


Figure 26-1. Animated Movement dialog box

3. Optionally, filter the animation list by selecting a category in the Category list.
4. Select an animation in the Animation list. You are responsible for knowing that the animation is appropriate to the entity you selected.
5. Optionally, specify a time scale. By changing the time scale you can make the animation run faster or slower than specified in the animation file.
6. Click OK.

26.3. Come to Stop

The Come to Stop task stops a surface entity using a normal stop, an emergency stop, or a slow drift.

To assign a Come to Stop task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Come to Stop**. The Come to Stop dialog box opens.
3. Select an option for how to stop.
4. Click OK.

26.4. Convoy Along

The Convoy Along task commands a unit to move along a route in a convoy. When the task starts, the lead entity moves far enough along the route for the rest of the unit to create a column that ends approximately at the beginning of the route. Then the unit begins moving along the route.



The Convoy Along task is valid only for ground-vehicle aggregates.

For more details about convoys, please see "Convoy Tasks," on page 25-10.

To assign a Convoy Along task:

1. Select the unit that you want to send in a convoy.
2. Choose **Task** → **Movement** → **Convoy Along**. The Convoy Along dialog box opens.
3. Specify the route to convoy along.
4. Click OK.

26.5. Convoy To

The Convoy To task commands a unit to move to a waypoint in a convoy. If you specify use of roads, VR-Forces uses the information in the road network to plan the convoy's movement. If you do not specify use of roads, the convoy moves directly across the terrain as an individual entity would move in a Move To task.



The Convoy To task is valid only for ground-vehicle aggregates.

For more details about convoys, please see "Convoy Tasks," on page 25-10.

To assign a Convoy To task:

1. Select the unit that you want to send in a convoy.
2. Choose **Task** → **Movement** → **Convoy To**. The Convoy To dialog box opens.
3. Specify the waypoint to convoy to.

When you specify use of roads, VR-Forces looks for roads over a fairly wide area and if one is found, will try to use them even if this results in an indirect approach to the waypoint. If there are no roads close to the route you would expect the unit to take to reach the waypoint, do not select this option.

4. Click OK.

26.6. Fast Rope Insertion

The Fast Rope Insertion task causes a helicopter to move to a specified waypoint and disembark a specified number of embarked humans using a fast rope animation. The disembarked humans slide down the rope, then move to a specified muster point and assume a kneeling posture.

This task is available only to helicopters that have the Fast Roping system, such as the MH-60 Black Hawk.

To assign a Fast Rope Insertion task:

1. Embark human entities on a helicopter that supports the Fast Rope Insertion task.
2. Select the helicopter.
3. Choose **Task** → **Movement** → **Fast Rope Insertion**. The Fast Rope Insertion dialog box opens.
4. Specify the waypoint at which the insertion should take place.
5. Specify the altitude at which the helicopter should hover.
6. Specify the number of humans to be disembarked and inserted.
7. Specify the waypoint the humans should go to after disembarking.
8. Click OK.

26.7. Fixed-Wing Land

"How Fixed-Wing Entities Land," on page 25-27, describes how fixed-wing entities land and the steps to design a landing approach. The Fixed Wing Land task automates the process of landing a fixed-wing aircraft. However, you must still specify a route to serve as the runway.

When the task is invoked, VR-Forces defines an approach route for the plane to move along. The length of this route is either the plane's altitude or the length needed to slow the plane down to landing speed, whichever is greater. The route is created and the plane moves along the route, coming to landing speed. At the end of the approach route, the plane is tasked to move along the runway.

! Although it is possible to land an entity on another entity, such as an aircraft carrier, if you want to embark an entity, you must use the Embark task, not a landing task.

To assign a Fixed Wing Land task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Fixed Wing Land**. The Fixed Wing Land dialog box opens.
3. In the Name box, type the name of the route you want it to land on, or select the route in the list or on the terrain.
4. Click OK.

26.8. Fixed Wing Takeoff

You can explicitly task a fixed-wing entity to take off. (For details about implicit take-off, please see "How Fixed-Wing Entities Take Off," on page 25-26.) In a Fixed Wing Takeoff task, the entity moves along the specified route, reaches takeoff speed (30% of minimum lift), and then moves to a point defined by the controller some distance up and away from the end of the route.

To assign a Fixed Wing Takeoff task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Fixed Wing Takeoff**. The Fixed Wing Takeoff dialog box opens.
3. Select the route to take off on or type its name in the Name box.
4. Click OK.

26.9. Fly Altitude

The Fly Altitude task commands a fixed-wing or rotary-wing entity to fly to the specified altitude at a specified ascent or descent rate, using its current heading and ordered speed. When it reaches the altitude it continues to fly at that altitude at the current heading until it receives other commands. The entity will not exceed the limits for altitude or ascent/descent rate set in the OPD for the entity type.

You cannot use a Fly Altitude task to cause a fixed-wing entity to take off. A fixed-wing entity that is on the ground ignores this task.

If a rotary-wing entity is on the ground and is given this task, it rises to the specified altitude as it moves forward at its current heading. If a rotary-wing entity is in the air and is given this task with an altitude of zero, it descends to approximately 33 M and continues flying at that altitude at its current heading.

To assign a Fly Altitude task:

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task** → **Movement** → **Fly Altitude**, or click the Fly Altitude button () on the Last Selected Object Panel or the Tasks Toolbar. The Fly Altitude dialog box opens.
3. In the Altitude box, specify the altitude above sea level to fly to.
4. In the Climb/Descent Rate box, specify the rate.
5. Click OK.

26.10. Fly Heading

The Fly Heading task commands a fixed-wing or rotary-wing entity to turn to the specified heading at the specified turn rate. It maintains its current altitude and speed. When it is finished turning to the new heading, it continues to fly at this heading until it receives other commands. The entity will not exceed the turn rate specified in the OPD.

A fixed-wing entity that is on the ground ignores this task.

If a rotary-wing entity is on the ground and is given this task, it rises to approximately 60 M and then drops down to approximately 33 M while simultaneously moving forward and turning to the specified heading. After turning to the new heading, it continues moving forward. Its altitude varies somewhat as it follows the terrain.

To assign a Fly Heading task:

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task** → **Movement** → **Fly Heading**, or click the Fly Heading button () on the Last Selected Object Panel or the Tasks Toolbar. The Fly Heading dialog box opens.
3. In the Heading box, specify the heading to fly to.
4. In the Turn Rate box, specify the rate.

5. Click OK.

26.11. Fly Heading and Altitude

The Fly Heading and Altitude task commands a fixed-wing or rotary-wing entity to fly to a specified altitude, at a specified ascent or descent rate, while turning to a specified heading at a specified turn rate. If you do not specify a heading or altitude, the entity continues to use the current value for that parameter.

While executing this task, the entity maintains its current speed. When it completes the command, it continues to fly at the new altitude and heading until given other commands. The entity will not exceed the limits on altitude, turn rate, and ascent/descent rate set in the OPD.

You cannot use a Fly Heading and Altitude task to cause a fixed-wing entity to take off. A fixed-wing entity that is on the ground ignores this task.

If a rotary-wing entity is on the ground and is given this task, it takes off and moves forward while it rises to the specified altitude and turns to the specified heading. If a rotary-wing entity is in the air and is given this task with an altitude of zero, it moves forward and turns to the new heading while it descends to approximately 33 M and continues flying at that altitude on the new heading.

To assign a Fly Heading and Altitude task:

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task** → **Movement** → **Fly Heading and Altitude**, or click the Fly Heading and Altitude button () on the Last Selected Object Panel or the Tasks Toolbar. The Fly Heading and Altitude dialog box opens.
3. In the Heading box, specify the heading to fly to.
4. In the Altitude box, specify the altitude above sea level to fly to.
5. In the Turn Rate box, specify the rate.
6. In the Climb/Descent Rate box, specify the rate.
7. Click OK.

26.12. Follow Entity

You can order an entity to follow a simulation object. You can specify a distance and offset between the simulation objects. If you tell two or more entities to follow the same simulation object, make sure that you do not give them the same offset (please see "Following Simulation Objects," on page 36-17).

An entity continues to follow its target entity until:

- The task is overridden.
- The followed entity is destroyed.

If entity A is assigned to follow entity B, but it cannot find entity B, entity A continues to look for entity B until entity B becomes available, or until entity A is assigned another task.

The following entity tries to match the speed of the followed entity unless it has to hold back or catch up to accommodate changes in direction or terrain effects.

To assign a Follow task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Follow Entity**. The Follow Entity dialog box opens.
3. Optionally, filter the entity list.
4. In the Name box, type the name of the entity to follow, or select it from the list or on the terrain.
5. Optionally, type an offset in the Behind, Right, and Above text boxes. If you want an entity to follow in front, to the left, or below, put a minus sign (-) in front of the value.
6. Click OK.

26.12.1. How Fixed-Wing Entities Follow Entities

A fixed-wing entity can follow an entity that is on the ground, or might be on the ground, for example if it is following a fixed-wing entity that lands. In this case, to ensure that the follower does not crash, make sure that it is following above the followed entity and that the distance above is high enough to account for changes in the terrain.

26.13. Generate Air Traffic From Flight Plans

VR-Forces can generate air traffic along routes created using X-Plane format flight plans. It includes several flight plans for flights in the U.S. and between the U.S. and Europe. You can add flight plans by placing them in *./data/flightPlans*.

When you execute this task, VR-Forces generates routes for flight plans that overlap a specified area of interest. If there are no flight plans in the area of interest, an error message is sent to the Information dialog box for the plan.

To assign a Generate Air Traffic From Flight Plans task:

1. Create a global plan.
2. Choose **Task** → **Generate Air Traffic From Flight Plans**. The Generate Air Traffic from Flight Plans dialog box opens.
3. In the Maximum Aircraft box, type the maximum number of aircraft you want to have flying at one time. This number only applies to the current task. If you run more than one instance of the task, each task handles the number of aircraft independently.
4. To create air traffic along the routes rather than have all the planes start at the beginning of the routes, select the Create Enroute check box.
5. Select an area of interest. Flight plans that overlap this area get generated.
6. Click OK.

26.14. Land at Airport Near Location

The Land at Airport Near Location task causes VR-Forces to find an airport near the specified location and land the specified aircraft on an appropriate runway based on the wind direction. This task requires use of a terrain database that has airport and runway data. The *ProceduralEarth (online).mtf* terrain has airport data.

To assign a Land at Airport Near Location task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Land at Airport Near Location**. The Land At Airport Near Location dialog box opens.
3. Specify the location in the dialog box or on the terrain.
4. Click OK.

26.15. Move Along Route

By default, a simulation object moves along a route from its nearest vertex to its end point.

- If you edit a route while a simulation object is moving to the starting point of that route as part of a Move Along Route task, the simulation object moves to the nearest vertex, whether it is the starting point, or not.
- If a human is moving along a route outside of a navigation area and you move it away from the route or change the route, the entity plans a path to the route vertex it is moving to rather than moving to the nearest point on the route and then proceeding to the target vertex.
- If you want a simulation object to repeatedly move along the same route from beginning to end (for example, to follow a circular route), be sure to clear the Start at Closest Vertex check box. If you do not, after the first time the simulation object moves along the route, it will keep going to the end vertex, because that will be the closest vertex to where it is starting the new Move Along Route task.
- When you task an air platform entity to follow a route, be sure to take into account the altitude of the route's vertices and the terrain the entity has to pass over to complete the task. (For more information, please see "Fixed-Wing Entity Movement on the Ground and in the Air," on page 25-25 and "Rotary-Wing Entity Tasks and Behaviors," on page 25-28.)
- Units do not generate subordinate routes until they reach the beginning of the route.
- A unit starts moving along a route when the leading edge of its formation is at the first point of the route. It is considered finished when the leading edge reaches the last point of the route.
- Pedestrian paths are valid for use as routes in this task.

To assign a Move Along Route task:

1. Select the simulation object.
2. Choose **Task** → **Movement** → **Move Along Route**, or click the Move Along Route button () on the Tasks Toolbar. The Move Along Route dialog box opens.
3. In the Name box, type the name of the route you want the simulation object to follow, or select a route in the list or on the terrain.
4. To have the simulation object move from the end of the route to the beginning, select Reverse Direction.
5. To have a ground vehicle use the road driving movement strategy, select the Treat Route as Road check box. The route is treated as a two-lane bidirectional road.

! If you select Treat Route as Road, the entity must be within 10 meters of the route or the task will fail.

---

6. To have the simulation object start the route at the beginning, clear the Start at Closest Vertex check box.
7. Click OK.

26.16. Move Into Formation

You can order a disaggregated ground unit to move to a formation at a specified location. Moving to a formation is not the same as setting formation. Setting formation changes the formation instantly. Moving to a formation causes the members of the unit to travel through the terrain into the new formation.

For aggregated units, Move Into Formation has the same effect as a Formation set data request.

To assign a Move Into Formation task:

1. Select the unit.
2. Choose **Task** → **Movement** → **Move Into Formation**. The Move Into Formation dialog box opens.
3. Select the desired formation from the list.
4. Specify the location at which the unit is to move into formation by entering coordinates, or by clicking on the terrain.
5. Specify the heading the unit is to assume at the target location.
6. Click OK.

26.17. Move to Altitude

You can order an aircraft entity to move to an altitude. When you assign this task, the entity moves to the new altitude at the same coordinates. Fixed-wing aircraft spiral to the new coordinates. Rotary-wing entities rise or descend. Moving to an altitude is not the same as setting altitude. Setting altitude moves an entity instantaneously. It is also not the same thing as Fly Altitude. At the conclusion of Move to Altitude, an aircraft circles or hovers at the new altitude. At the conclusion of Fly Altitude, an aircraft continues moving at its current heading.



Fixed-wing entities that are on the ground cannot respond to a Move to Altitude task. If they are given this task, a message is printed to the object console and the task is ignored.

To assign a Move to Altitude task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Move to Altitude**, or click the Move to Altitude button () on the Tasks Toolbar. The Move to Altitude dialog box opens.
3. Type the altitude that you want the entity to move to.
4. Click OK.

26.18. Move to Depth

The Move to Depth task lets you specify the depth for subsurface entities. When an entity moves to a depth, it continues any horizontal movement that was in effect prior to starting the task.

To assign a Move to Depth task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Move to Depth**. The Move to Depth dialog box opens.
3. Select an option for the depth. If you choose a specific depth, type a depth in the Target Depth box.
4. Specify the ascent/descent rate.
5. Click OK.

26.19. Move to Location

You can order a simulation object to move to a location. If navigation data is present, simulation objects take it into account in planning their path to the location.

If you add Move to Location tasks to a plan, you have more options than when you give an independent task.

Moving to a location is not the same as setting location. Setting location moves a simulation object instantaneously. Moving to a location causes a simulation object to travel through the terrain.

If you want a simulation object to use the road network to move to a location, use Move to Location (Plan Along Roads). For details about road movement, please see "Entity Movement On Roads and Off Roads," on page 25-21.

To assign a Move to Location task:

1. Select the simulation object.
2. Choose **Task** → **Movement** → **Move to Location**, or click the Move to Location button () on the Tasks Toolbar. The Move to Location dialog box opens.
3. Click on the terrain where you want the simulation object to move, or enter the coordinates of the location in the Move to Location dialog box.
4. Optionally, set the altitude, as described in "Specifying an Object's Altitude," on page 14-18.



If you specify the location by clicking and the task is for an air platform that is in the air, be sure to change the altitude to a height that will prevent the entity from crashing.

5. Click OK.

26.19.1. Adding Multiple Move to Location Tasks to a Plan

When you add a Move to Location task to a plan, you can add multiple locations without clicking OK and selecting the task again from the menu. Also, since in this method you cannot specify the altitude for each location, you can specify a height above the terrain for the altitude of the succeeding points.

To add multiple Move to Location tasks to a plan:

1. In the Plan window for a simulation object, choose **Task** → **Movement** → **Move to Location**. The Move to Location dialog box opens.
2. Select Each Click Generates A Task.
3. If you want to specify an altitude for the locations, specify an altitude, as described in "Specifying an Object's Altitude," on page 14-18.
4. Click a point on the terrain for each location that you want the simulation object to move to. A Move to Location task is added to the plan for each location.
5. Click OK.

26.20. Move to Location (Plan Along Roads)

The Move to Location (Plan Along Roads) task directs a ground vehicle to move to a location using roads, if there is a road network.

When an entity receives this task, it determines if there is a road network near by. If there is, it moves directly to the nearest point on the road network. Then it moves along the roads using road driving behavior until it gets to the point on the road that is closest to the destination. Then the entity moves off the road directly to the destination.

For more information about road driving and how the path planning tasks work, please see "Entity Movement On Roads and Off Roads," on page 25-21.

To assign a Move to Location (Plan Along Roads) task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Move to Location (Plan Along Roads)**. The Move to Location (Plan Along Roads) dialog box opens.
3. Click on the terrain where you want the entity to move, or enter the coordinates of the location in the dialog box.
4. Click OK.

26.21. Move to Location (Plan Path)

The Move to Location (Plan Path) task is for surface and subsurface entities. It plots a route to the destination location based on the depth of water between the starting point and ending point. You specify the minimum depth required for the ship to maneuver. You can specify that the depth be based on the ship's draft or you can enter a specific depth value. The ship's draft is determined by calculating the distance its bounding box extends below its origin, as set in the Simulation Object Editor.



The formula for calculating the draft is:

draft = height/2 - offset (Up)

where height and offset are set in the Size group box in the Simulation Object Editor.

If you give this task to a submarine, VR-Forces assumes that it is moving on the surface.



For this task to work, your terrain must include S-57 data or depth information provided in a similar way.

To assign a Move to Location (Plan Path) task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Move to Location (Plan Path)**. The Move to Location (Plan Path) dialog box opens.
3. Click on the terrain where you want the entity to move, or enter the coordinates of the location in the dialog box.
4. Specify how you want VR-Forces to determine the minimum depth of water for the route.
5. If you want to see the path displayed, select Display Route.
6. Click OK.

26.22. Move to Waypoint

You can order a simulation object to move to a point or a simulation object. If navigation data is present, simulation objects take it into account in planning their path to the waypoint.

Moving to a point is not the same as setting location. Setting location moves a simulation object instantaneously. Moving to a point causes a simulation object to travel through the terrain.

If you want a ground entity to use the road network to move to a waypoint, use the Move to Waypoint (Plan Along Roads) task. For details about road movement, please see "Entity Movement On Roads and Off Roads," on page 25-21.



Rotary-wing entities are not able to reach moving waypoints or land on moving ships. A rotary-wing entity will approach a given waypoint or landing point, but will never completely reach the goal.

To assign a Move to Waypoint task:

1. Select the simulation object.
2. Choose **Task** → **Movement** → **Move to Waypoint**, or click the Move to Waypoint button () on the Tasks Toolbar. The Move to Waypoint dialog box opens.
3. Optionally, filter the selection list.
4. In the Name box, type the name of the waypoint or entity to move to, or select it in the list or on the terrain.
5. Click OK.

26.23. Move to Waypoint (Plan Along Roads)

The Move to Waypoint (Plan Along Roads) task directs an entity to move to a waypoint and use the road network if it is available.

When an entity receives this task, it determines if there is a road network near by. If there is, it moves directly to the nearest point on the road network. Then it moves along the roads using road driving behavior until it gets to the point on the road that is closest to the destination. Then the entity moves off the road directly to the destination.

For more information about road driving and how the path planning tasks work, please see "Entity Movement On Roads and Off Roads," on page 25-21.

To assign a Move to Waypoint (Plan Along Roads) task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Move to Waypoint (Plan Along Roads)**. The Move to Waypoint (Plan Along Roads) dialog box opens.
3. Select the waypoint.
4. Click OK.

26.24. Move to Waypoint (Plan Path)

The Move to Waypoint (Plan Path) task applies to surface and subsurface entities. It plots a route to the waypoint based on the depth of water between the starting point and ending point. You specify the minimum depth required for the ship to maneuver. You can specify that the depth be based on the ship's draft or you can enter a specific depth value. The ship's draft is determined by calculating the distance its bounding box extends below its origin, as set in the Simulation Object Editor.



The formula for calculating the draft is:

draft = height/2 - offset (Up)

where height and offset are set in the Size group box in the Simulation Object Editor.

If you give this task to a submarine, VR-Forces assumes that it is moving on the surface.



For this task to work, your terrain must include S-57 data or depth information provided in a similar way.

To assign a Move to Waypoint (Plan Path) task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Move to Waypoint (Plan Path)**. The Move to Waypoint (Plan Path) dialog box opens.
3. Select the waypoint to move to.
4. Specify how you want VR-Forces to determine the minimum depth of water for the route.
5. If you want to see the path displayed, select Display Route.
6. Click OK.

26.25. Orbit

The Orbit task allows a fixed-wing aircraft to circle around a point without needing to create a route for it to follow.

To assign an Orbit task:

1. Select the entity that you want to orbit.
2. Choose **Task** → **Movement** → **Orbit**. The Orbit dialog box opens.
3. Specify the coordinates of a point on the terrain around which the entity should orbit. You can click on the terrain to specify the point.
4. Specify the altitude at which to fly.
5. Specify the radius of the circle.
6. Select or clear the Clockwise option to specify the direction of travel.
7. Click OK.

26.26. Orbit Object

The Orbit Object task causes an aircraft to fly a circular pattern around a simulation object or point tactical graphic. If the object moves, the aircraft moves to maintain the orbital relationship.

To assign an Orbit Object Task:

1. Select the entity that you want to orbit.
2. Choose **Task** → **Movement** → **Orbit Object**. The Orbit Object dialog box opens.
3. Select the simulation object or tactical graphic around which you want the entity to orbit.
4. In the Orbit Radius box, specify the radius of the orbit.
5. To orbit clockwise, select the Clockwise check box.
6. Click OK.

26.27. Parallel Pattern

The Parallel Pattern task causes an aircraft to fly a search and rescue trackline search using a parallel pattern. When you issue this task, VR-Forces generates routes for each leg of the pattern. As the aircraft completes each leg of the pattern, the route is removed. Figure 26-2 illustrates a three leg pattern.

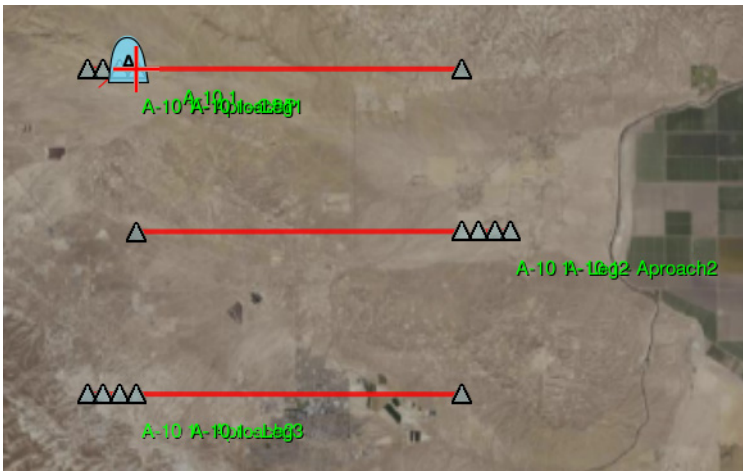


Figure 26-2. Parallel pattern

To assign a parallel pattern task:

1. Select an aircraft entity.
2. Choose **Task** → **Movement** → **Search and Rescue** → **Parallel Pattern**. The Parallel Pattern dialog box opens.
3. Specify the parameters, as follows:
   - Commence Search Point. The coordinates of the point at which to start the trackline.
   - Major Axis. The angle of the long leg of the search area.
   - Minor Axis. The angle of the short leg of the search area.
   - Leg. The length of the line to fly.
   - Space. The distance between each leg.
   - Altitude. The altitude above sea level at which to fly.
   - Air Speed. The speed at which to fly.
   - Legs. The number of passes to make.
4. Click OK. VR-Forces generates routes for the pattern.

26.28. Patrol Route

When a simulation object patrols along a route, it moves to the nearest vertex of the route, to the end point, back to the beginning of the route, and then continues going back and forth along the route until given another command.

If a human is moving along a route outside of a navigation area and you move it away from the route or change the route, the entity plans a path to the route vertex it is moving to rather than moving to the nearest point on the route and then proceeding to the target vertex.



For more information about moving along routes, see the issues discussed in "Move Along Route," on page 26-12.

To assign a Patrol Route task:

1. Select the simulation object.
2. Choose **Task** → **Movement** → **Patrol Route**, or click the Patrol Route button () on the Tasks Toolbar. The Patrol Along Route dialog box opens.
3. In the Name box, type the name of the route to patrol, or select the route in the list or on the terrain.
4. Click OK.

26.29. Patrol Between

A simulation object can patrol between two point objects, simulation objects, or both. When a simulation object patrols between two objects, it goes to the starting object, then moves back and forth between the two objects.

To assign a Patrol Between task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Patrol Between**, or click the Patrol Between button () on the Tasks Toolbar. The Patrol Between dialog box opens.
3. In the Name box for Point 1, type the name of the first object to patrol between, or select an object in the list or on the terrain.
4. In the Name box for Point 2, type the name of the second object to patrol between, or select an object in the list or on the terrain.
5. Click OK.

26.30. Pattern Hold (Location)

The Pattern Hold (Location) task causes a fixed-wing or rotary-wing entity to execute a racetrack hold pattern at the specified location.

In a standard racetrack, the aircraft tries to spend one minute per leg. Therefore, the length of the leg varies with the speed. The default turn radius is three degrees/second. In a custom racetrack, you specify the length and width of the legs.

The aircraft enters the pattern using direct, parallel, or teardrop entry. It determines the approach based on its location relative to the desired pattern. If you set the plane's object console to Info, the chosen entry method is displayed.

It may take an aircraft one or two loops before it settles into the desired pattern.

To assign a Pattern Hold (Location) task:

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task** → **Movement** → **Pattern Hold (Location)**. The Pattern Hold (Location) dialog box opens.
3. Enter the Holding Fix (location) or select it on the terrain.
4. Specify the heading, speed, and altitude for the hold pattern.
5. Specify the turn direction for the hold pattern.
6. For the Pattern Type, select Time or Distance. If you select Time, specify the amount of time per leg of the pattern. If you select Distance, specify a length and width.
7. Click OK.

26.31. Pattern Hold (Waypoint)

The Pattern Hold (Waypoint) task causes a fixed-wing or rotary-wing entity to execute a racetrack hold pattern at the specified waypoint.

In a standard racetrack, the aircraft tries to spend one minute per leg. Therefore, the length of the leg varies with the speed. The default turn radius is three degrees/second. In a custom racetrack, you specify the length and width of the legs.

The aircraft enters the pattern using direct, parallel, or teardrop entry. It determines the approach based on its location relative to the desired pattern. If you set the aircraft's object console to Info, the chosen entry method is displayed.

It may take an aircraft one or two loops before it settles into the desired pattern.

To assign a Pattern Hold (Waypoint) task:

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task** → **Movement** → **Pattern Hold (Waypoint)**. The Pattern Hold (Waypoint) dialog box opens.
3. In the Holding Fix section, select the waypoint.

4. Specify the heading, speed, and altitude for the hold pattern.
5. Specify the turn direction for the hold pattern.
6. For the Pattern Type, select Time or Distance. If you select Time, specify the amount of time per leg of the pattern. If you select Distance, specify a length and width.
7. Click OK.

26.32. Rotary Wing Airlift Cargo to Location

The Rotary Wing Airlift Cargo to Location task causes an unmanned rotary-wing aircraft to move to a list of objects, embark them onto itself, move to a location, and disembark the objects. The objects must be configured as embarkable.

If the total weight of the objects plus the weight of the aircraft is greater than the aircraft can carry, the aircraft will embark the objects, but be unable to fly.

The unmanned aircraft needs to use the Quadcopter or Quadcopter Heavy Lift movement dynamics system.

To assign a Rotary Wing Airlift Cargo to Location task:

1. Select an unmanned aircraft.
2. Choose **Task** → **Movement** → **Rotary Wing Airlift Cargo to Location**. The Rotary Wing Airlift Cargo To Location dialog box opens.
3. Optionally, filter the list of object types.
4. Type the name of an object in the Name box, or in the Cargo Objects list select one or more objects.
5. In the Cruise Altitude box, specify the altitude at which you want the aircraft to fly.
6. Specify the destination location by typing in the coordinates, or click the Destination Location filter button () to give the Destination Location focus, and click on the terrain to specify the destination location.
7. Click OK.

26.33. Rotary Wing Land

The Rotary Wing Land task automates the process of landing a rotary-wing aircraft. Rotary-wing aircraft can land on points.

! Although it is possible to land an entity on another entity, such as an aircraft carrier, if you want to embark an entity, you must use the Embark task, not a landing task.

To assign a Rotary Wing Land task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Rotary Wing Land**. The Rotary Wing Land dialog box opens.
3. In the Name box, type the name of the waypoint on which you want the entity to land, or select the waypoint in the list or on the terrain.
4. Click OK.

26.34. Rotary Wing Retrieve Cargo

The Rotary Wing Retrieve Cargo task causes an unmanned aircraft to move to one or more cargo items and embark them onto itself. It hovers at the location of the last item retrieved until given further tasks.

The unmanned aircraft needs to use the Quadcopter or Quadcopter Heavy Lift movement dynamics system.

To assign a Rotary Wing Retrieve Cargo task:

1. Select an unmanned aircraft.
2. Choose **Task** → **Movement** → **Rotary Wing Retrieve Cargo**. The Rotary Wing Airlift Cargo To Location dialog box opens.
3. Optionally, filter the list of object types.
4. Type the name of an object in the Name box or in the Cargo Objects list, select one or more objects.
5. In the Cruise Altitude box, specify the altitude at which you want the aircraft to fly.
6. Click OK.

26.35. Rotary Wing Unload Cargo

The Rotary Wing Unload Cargo task causes a rotary-wing aircraft to unload (disembark) its cargo at the current location of the aircraft.

To assign a Rotary Wing Unload Cargo task:

1. Select an unmanned aircraft.
2. Choose **Task** → **Movement** → **Rotary Wing Unload Cargo**. The aircraft unloads its cargo.

26.36. Sail Heading

The Sail Heading task causes a surface entity to sail towards the specified heading. After turning to the heading, the entity moves at its ordered speed indefinitely.

To assign a Sail Heading task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Sail Heading**. The Sail Heading dialog box opens.
3. Specify a heading in the indicated units (radians or degrees).
4. Specify the diameter of the turn.
5. Specify the direction of the turn, as follows:
   - Toward Heading. Chooses the most direct turn.
   - Port. Turns to port until it reaches the heading.
   - Starboard. Turns to starboard until it reaches the heading.
6. Click OK.

26.37. Sweep Naval Mines

The Sweep Naval Mines task causes an entity to look for and disable naval mines along a route. It looks within a specified distance from the route. An error rate specifies a percentage chance that the entity will detect a mine, but not disable it. The Sweep Naval Mines task is available to surface entities that have a Naval Mine Sweep system.

To assign a Sweep Naval Mines task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Sweep Naval Mines**. The Sweep Naval Mines dialog box opens.
3. Select the route to follow.
4. Specify the sensing range. The default is 300 meters.
5. Optionally, specify the disarm failure probability. This is the percentage chance that a mine will be found, but not disabled.
6. Click OK.

26.38. Turn to Heading

The Turn to Heading task provides a realistic way for ground vehicles to turn to a new heading. Rather than turning instantly, as happens with the Heading request, a vehicle pivots or makes a K-turn to the new heading. The turning behavior for an entity is controlled by its **can-pivot** parameter in the object parameter database.

To assign a Turn to Heading task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Turn to Heading**. The Turn to Heading dialog box opens.
3. Type a heading.
4. In the Relative To list, select one of the following options:
   - North. Turn to the specified absolute heading.
   - Current Heading. Turn the specified number of degrees relative to the current heading.
   - Host (if embarked). If embarked, turn to the specified heading relative to the host simulation object.
5. Click OK.

26.39. Zig Zag to Heading

The Zig Zag to Heading task causes a surface vessel to move towards a heading using a zig zag pattern.

To assign a Zig Zag to Heading task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Zig Zag to Heading**. The Zig Zag to Heading dialog box opens.
3. Specify the heading.
4. Specify the amount of time to spend on each leg of the zig zag pattern.
5. Specify the angle of direction for each leg.



27. Engagement Tasks for Entity-Level Scenarios

This chapter describes tasks in the engagement category that apply to entity-level scenarios.

Arm Mine at Depth 27-3
Attack Aircraft with Guns 27-3
Drop Naval Depth Charge 27-4
Drop Naval Depth Charge at Location 27-4
Drop Naval Mine 27-5
Drop Naval Mines Along Route 27-6
Execute Close Air Support 27-7
Explode Charge at Depth 27-9
Fire at Target 27-9
Fire Cruise Missile 27-10
Fire for Effect Tasks 27-11
  Firing Naval Guns 27-12
Lase Location 27-12
Lase Target 27-13
Launch Anti-Submarine Missile (Vertical) 27-13
Launch Ballistic Missile at Location 27-14
Launch Ballistic Missile at Target Object 27-14
Launch Counter Measures 27-15
Launch Smoke 27-15
Launch Torpedo 27-16
  Anti-ship (Fixed Depth) 27-16
  Anti-Submarine 27-17
Place IED 27-18
Provide Suppressive Fire 27-19
Provide Suppressive Fire at Location 27-20

Release Bomb Tasks 27-20
Strafe Ground Target 27-22
Throw Grenade Tasks 27-24
Throw Grenade at Location 27-24
Throw Grenade at Target 27-24

27.1. Arm Mine at Depth

If a naval mine has been deployed, Arm Mine at Depth lets you change the mine's parameters.

Arm Mine at Depth is a system scripted task. It is disabled by default.

To assign an Arm Mine at Depth task:

1. Select the mine.
2. Choose **Task** → **Arm Mine at Depth**. The Arm Mine at Depth dialog box opens.
3. Specify the depth at which to arm the mine.
4. Specify the radius within which target entities will detonate the mine.
5. Optionally, specify a time when the mine will be armed, in days and hours:minutes:seconds.
6. In the Targets list, add the types of entities that can trigger the mine, as follows:
   a. Click the Add button (). The Entity Type dialog box opens.
   b. Build an entity type by selecting options from the lists for each enumeration component or type an enumeration.
   c. Click OK.
7. To specify that only hostile entities will set off the mine, select Hostile Only. If this setting is cleared, any entity will set off the mine.
8. Click OK.

27.2. Attack Aircraft with Guns

The Attack Aircraft with Guns task commands aircraft that have mounted guns, such as the A10, to attack other aircraft with their guns rather than air-to-air missiles.

To assign an Attack Aircraft with Guns task:

1. Select an aircraft entity that has a gun.
2. Choose **Task** → **Engagement** → **Attack Aircraft with Guns**. The Attack Aircraft with Guns dialog box opens.
3. Select the target entity.
4. Click OK.

27.3. Drop Naval Depth Charge

The Drop Naval Depth Charge task causes a fixed-wing entity to drop a depth charge at its current location. To execute this task, an entity must have a Naval Depth Charge Deployment weapon system.

To assign a Drop Naval Depth Charge task:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Drop Naval Depth Charge**. The Drop Naval Depth Charge dialog box opens.
3. In the Naval Depth Charge list, select the type of depth charge to drop.
4. Specify the depth at which you want the depth charge to explode.
5. Click OK.

27.4. Drop Naval Depth Charge at Location

The Drop Naval Depth Charge at Location task causes a fixed-wing entity to move to the specified location and drop a depth charge. To execute this task, an entity must have a Naval Depth Charge Deployment weapon system.



In this task, the location is the location of the entity when it drops the depth charge. Since the entity is flying, the location must include a reasonable altitude.

To assign a Drop Naval Depth Charge at Location task:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Drop Naval Depth Charge at Location**. The Drop Naval Depth Charge At Location dialog box opens.
3. In the Naval Depth Charge list, select the type of depth charge to drop.
4. Specify the depth at which you want the depth charge to explode.
5. Specify the location where you want to drop the depth charge.

   By default, this task uses the altitude of the entity at the time the task is assigned as the altitude for the drop location. (Use Current Altitude check box.) If you specify an altitude and the check box is selected, VR-Forces ignores the altitude you entered and uses the current entity altitude. If you want to specify the altitude, clear the check box and make sure that you specify the altitude in the Altitude group box. If you click on the terrain to specify the X, Y location, the altitude gets set to 0, so be sure to enter the altitude you want after clicking on the terrain.
6. Click OK.

27.5. Drop Naval Mine

The Drop Naval Mine task allows entities with a Naval Mine Deployment system (fixed-wing, surface, and subsurface) to drop naval (underwater) mines at their current location.

To assign a Drop Naval Mine task:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Drop Naval Mine**. The Drop Naval Mine dialog box opens (Figure 27-1).

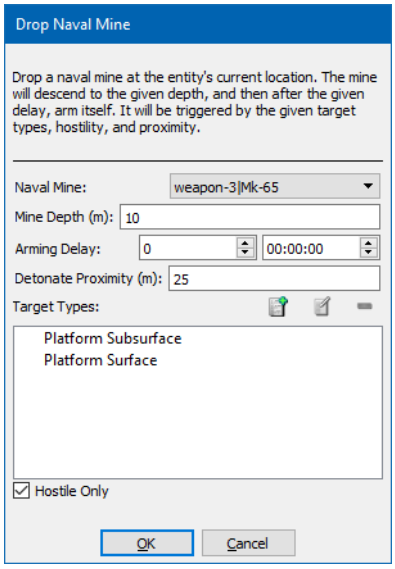


Figure 27-1. Drop Naval Mine dialog box

3. In the Naval Mine list, select the type of mine to drop.
4. In the Mine Depth field, set the depth to drop the mine. Positive numbers are below sea level.
5. In the Arming Delay fields, set the time to elapse before the mine becomes armed. The first field is days. The second field is Hours:Minutes:Seconds.
6. In the Detonate Proximity field, set the distance from the mine within which a target entity will detonate the mine.
7. In the Trigger Entity Types list, add the types of entities that can trigger the mine, as follows:
   a. Click the Add button ( ). The Entity Type dialog box opens.
   b. Build an entity type by selecting options from the lists for each enumeration component or type an enumeration.
   c. Click OK.
8. To specify that only hostile entities will set off the mine, select Hostile Only. If this setting is cleared, any entity will set off the mine.

9. Click OK.

27.6. Drop Naval Mines Along Route

The Drop Naval Mines Along Route task allows entities with a Naval Mine Deployment system (fixed-wing, surface, and subsurface) to drop naval (underwater) mines while they travel along a route.

To assign a Drop Naval Mines Along Route task:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Drop Naval Mines Along Route**. The Drop Naval Mines Along Route dialog box opens.
3. Select the route to follow.
4. In the Mine Spacing field, specify the distance between each mine.
5. In the Naval Mine list, select the type of mine to drop.
6. In the Mine Depth field, set the depth to drop the mine. Positive numbers are below sea level.
7. In the Arming Delay fields, set the time to elapse before the mine becomes armed. The first field is days. The second field is Hours:Minutes:Seconds.
8. In the Detonate Proximity field, set the distance from the mine within which a target entity will detonate the mine.
9. In the Trigger Entity Types list, add the types of entities that can trigger the mine, as follows:
   a. Click the Add button (). The Entity Type dialog box opens.
   b. Build an entity type by selecting options from the lists for each enumeration component or type an enumeration.
   c. Click OK.
10. To specify that only hostile entities will set off the mines, select Hostile Only. If this setting is cleared, any entity will set off the mine.
11. Click OK.

27.7. Execute Close Air Support

The Execute Close Air Support task assigns a fixed-wing fighter or bomber a close air support (CAS) task based on a 9-Line Briefing. You can call for a bombing run or a strafing run. The entity being tasked must be in the air to execute this task.



A 9-Line Briefing is a format used by U.S. armed forces to call in CAS. The Execute Close Air Support dialog box explicitly or implicitly requests all of the information in a 9-Line Briefing. However, the scripted task does not use all of this information to simulate CAS. (The procedure notes which fields are not used.) The task has parameters for all 9 lines to provide simulation authenticity. Since this is a scripted task, you could modify it to use all of the parameters.

To assign an Execute Close Air Support task:

1. Select the entity you want to execute close air support.
2. Choose **Task** → **Engagement** → **Execute Close Air Support**. The Execute Close Air Support dialog box opens (Figure 27-2).
3. In the Initial Point section, specify the location from which the aircraft should line up with the target by entering location values or clicking on the terrain.
4. In the Heading box, specify the approach heading. Optionally, specify an offset if you want the entity to favor a particular side of the line of approach.
5. In the Distance box, specify the distance from the initial point to the target. This information is not used by the task.
6. Select the target. Target selection satisfies lines 4, 5, and 6 of a 9-Line briefing, because it provides the location, altitude, and name of the target.
7. In the Mark Type box, specify how the target is marked. This information is not used by the task.

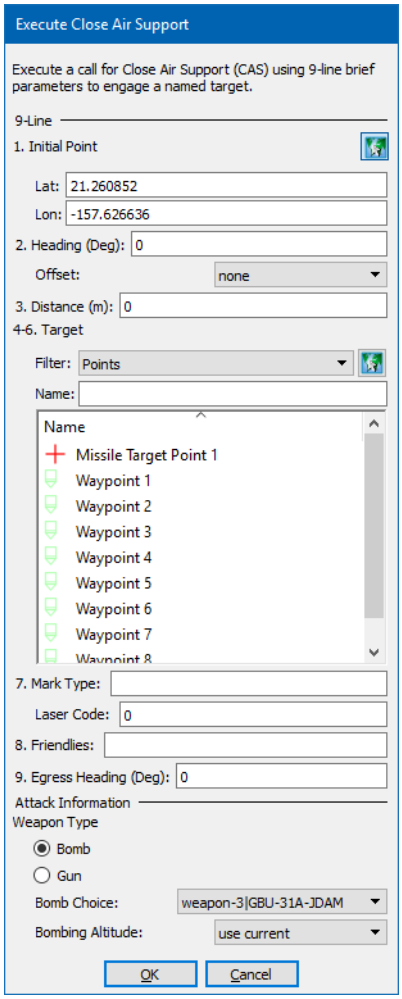


Figure 27-2. Execute Close Air Support dialog box

8. If you are using a laser designator to mark the target, enter the laser code. (If you specify a laser code, an entity in the simulation must be lasing the target for the CAS to work.)
9. In the Friendlies box, specify the distance and direction of friendly forces from the target. This information is not used by the task.
10. In the Egress Heading box, specify the heading by which the attacking entity should leave after firing at the target. It will continue on this heading until it is given another task.
11. In the Attack Information section, select bomb or gun. If you select gun, the entity executes a strafing attack.
12. If you selected bomb, select a Bomb Choice and Bombing Altitude.
13. Click OK.

27.8. Explode Charge at Depth

The Explode Charge at Depth task explodes a depth charge at the specified depth. You can use this task to explode a depth charge that has been deployed, but has not yet exploded.

Explode Charge at Depth is a system scripted task. It is disabled by default.

To assign an Explode Charge at Depth task:

1. Select the depth charge.
2. Choose **Task** → **Explode Charge at Depth**. The Explode Charge at Depth dialog box opens.
3. Specify the depth.
4. Click OK.

27.9. Fire at Target

The Fire at Target task commands an entity to fire at a specific target. You can specify which weapon system to use and how many rounds to fire.

When you issue this task, the entity ignores its rules of engagement and force types. Entities will fire at their own force if told to do so. The shooter must have line-of-sight to the target and have an appropriate weapon system and ammunition for the target.

To assign a Fire at Target task:

1. Select the entity you want to fire.
2. Choose **Task** → **Engagement** → **Fire at Target**, or click the Fire at Target button () on the Last Selected Object Panel or the Tasks Toolbar. The Fire at Target dialog box opens.
3. Select the target entity.
4. In the Weapon Selection group box, select an option as follows:
   - To let VR-Forces choose the weapon, select Automatic.
   - To choose the weapon yourself, select Manual and select a weapon from the list.
5. In the Maximum Rounds to Fire box, enter the number of rounds you want the entity to fire. (The entity stops firing when the target is destroyed or when the maximum number of rounds has been fired, whichever comes first.)

27.10. Fire Cruise Missile

A cruise missile can fly directly to a target or follow a route.

To assign a Fire Cruise Missile task:

1. Select an entity that has cruise missiles.
2. Choose **Task** → **Engagement** → **Fire Cruise Missile**. The Fire Cruise Missile dialog box opens (Figure 27-3).

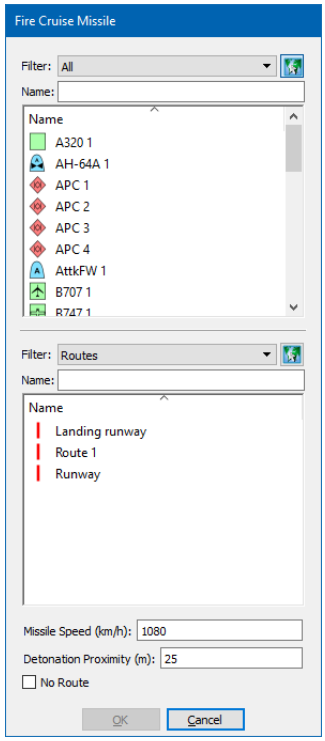


Figure 27-3. Fire Cruise Missile dialog box

3. Select the target from the list or on the terrain, or type its name in the Name box. Targets can be entities or points.
4. Select a route for the missile to follow or select the No Route check box. If no route is selected, the missile flies directly to the target.
5. Specify the ordered speed for the missile.
6. Specify the detonation proximity for the warhead. It will detonate within this range of the selected target.
7. Click OK.

27.11. Fire for Effect Tasks

You can task artillery entities and surface vessels that have naval guns to fire for effect. The three options are:

- Fire-for-Effect on Entity.
- Fire-for-Effect on Location, where location is a point on the terrain.
- Fire-for-Effect on Target, where a target is a waypoint.

If an entity has more than one weapon, you can select multiple weapons to fire. However, they will all fire on the same target. You cannot target them independently.

To assign a Fire-for-Effect task:

1. Select the entity to which you want to assign the task.
2. Choose **Task** → **Engagement** → **Fire-for-Effect on Entity** (or Location or Target). The appropriate dialog box opens.
3. Identify the target as follows:
   - For FFE Entity, select the target entity in the list, select it on the terrain, or type its name in the Name box.
   - For FFE Location, click a location on the terrain, or enter the coordinates.
   - For FFE Target, select a point from the list, or select it on the terrain.



For FFE Entity and FFE Target, you can filter the target list.

5. In the Select Weapons list, select the weapons that you want to fire.
6. Specify the number of rounds.
7. Optionally, specify the height above the terrain at which rounds should detonate.
8. Click OK.

- The Fire-for-Effect tasks are only meaningful when assigned to artillery entities or surface vessels that have naval guns.
- When you choose Fire-for-Effect on Entity, if the target entity is moving, VR-Forces dead reckons its position and the artillery fires at its expected location, not the location it was at the start of the task.
- Artillery entities have maximum and minimum ranges. To see these ranges, enable range rings for the entity. (For details, please see "Displaying Threat and Sensor Range Rings," on page 43-16.) The back-end console window prints messages when you select targets that are outside of the entity's range.
- The default preferred trajectory height for artillery and naval guns is 500 meters. If this height is not attainable given other parameters involved in calculating the trajectory, VR-Forces calculates a more appropriate height. You can specify a different preferred trajectory height with the Preferred Trajectory Height set data request. For details, please see "Preferred Trajectory Height," on page 34-32.

27.11.1. Firing Naval Guns

Ordinarily, naval guns fire their munitions in a parabola, similar to artillery. However, they can also fire directly at a target if the following conditions are true:

- The target's range is greater than the weapon's minimum range, but less than the first value in its range list.
- The capable-of-direct-alignment parameter for the weapon system's naval-gun-controller is set to True.
- The firing entity has line-of-sight to the target.

27.12. Lase Location

The Lase Location task simulates aiming a laser beam at a location to guide a laser guided weapon. By default, entities with laser capability lase autonomously. Use this task if you want to lase a specific location. An entity can lase a location while it is executing a movement task.

To lase a location, an entity must have a Laser Designator system. For more information about lasing targets, please see:

- "Lasing Targets," on page 9-18.
- "Laser Code," on page 34-24.
- "Synchronize Laser Code," on page 34-43.

To assign a Lase Target task:

1. Select the entity that will aim the laser.
2. Choose **Task** → **Engagement** → **Lase Location**. The Lase Location dialog box opens.
3. Click on the map or enter the coordinates of the location.
4. Click OK.

27.13. Lase Target

The Lase Target task simulates aiming a laser beam at a target to guide a laser guided weapon. By default, entities with laser capability lase autonomously. Use this task if you want to lase a specific target. An entity can lase a target while it is executing a movement task.

To lase a target, an entity must have a Laser Designator system. For more information about lasing targets, please see:

- "Lasing Targets," on page 9-18.
- "Laser Code," on page 34-24.
- "Synchronize Laser Code," on page 34-43.

To assign a Lase Target task:

1. Select the entity that will aim the laser.
2. Choose **Task** → **Engagement** → **Lase Target**. The Lase Target dialog box opens.
3. Select the entity that you want to lase.
4. Click OK.

27.14. Launch Anti-Submarine Missile (Vertical)

The Launch Anti-Submarine Missile (Vertical) task lets a surface entity launch a missile that flies to the vicinity of the target and then drops a homing torpedo. To execute this task, an entity needs an Anti-Submarine Missile (Vertically Launched) weapon system.

To assign a Launch Anti-Submarine Missile (Vertical) task:

1. Select the surface entity.
2. Choose **Task** → **Engagement** → **Launch Anti-Submarine Missile (Vertical)**. The Launch Anti-Submarine Missile (Vertical) dialog box opens.
3. Select the target entity.
4. Specify the location at which to drop the homing torpedo.
5. Click OK.

27.15. Launch Ballistic Missile at Location

The Launch Ballistic Missile at Location task lets an entity that has a strategic missile weapon system launch a missile at a location.

> The Ballistic Missile tasks do not use physics-based modeling. They calculate a simple parabola for the trajectory and move the missile along the trajectory ignoring atmospherics, thrust stages, cargo weight, gravity, and the rotation of the earth.

To launch a ballistic missile at a location:

1. Select an entity with a strategic missile weapon system.
2. Choose **Task** → **Engagement** → **Launch Ballistic Missile at Location**. The Launch Ballistic Missile at Location dialog box opens.
3. Specify a location.
4. In the Missile Type list, select the missile to fire.
5. Click OK.

27.16. Launch Ballistic Missile at Target Object

The Launch Ballistic Missile at Target Object task lets an entity that has a strategic missile weapon system launch a missile at a point object, entity, or unit. The destination of the missile is the target's location when the missile is fired. The missile does not track moving targets.

> The Ballistic Missile tasks do not use physics-based modeling. They calculate a simple parabola for the trajectory and move the missile along the trajectory ignoring atmospherics, thrust stages, cargo weight, gravity, and the rotation of the earth.

To launch a ballistic missile at a target object:

1. Select an entity with a strategic missile weapon system.
2. Choose **Task** → **Engagement** → **Launch Ballistic Missile at Target Object**. The Launch Ballistic Missile at Target Object dialog box opens.
3. Optionally, filter the list of objects.
4. Select the object to fire at.
5. In the Missile Type list, select the missile to fire.
6. Click OK.

27.17. Launch Counter Measures

Fixed-wing and rotary-wing entities can launch counter measures, such as chaff and flares. Launch Counter Measures is not an exclusive task. It does not interrupt an entity's current task. (For more information, please see "Launching Counter Measures (Chaff and Flare)," on page 9-26.)

The Evade Missile reactive task automatically launches chaff and flare when a fixed-wing entity evades a missile.

To assign a Launch Counter Measures task:

1. Select the entity that you want to launch counter measures.
2. Choose **Task** → **Engagement** → **Launch Counter Measures**. The Launch Counter Measures dialog box opens.
3. In the Counter Measure Types list, select the type of counter measures you want to launch.
4. In the Number To Fire box, specify the number of counter measures to launch.
5. In the Time Between Fires box, specify the amount of time, in seconds, between each counter measure launch.
6. Click OK.

27.18. Launch Smoke

The Launch Smoke task causes an entity to create a smoke cloud. The cloud is created as an environmental object. It grows over time.

An entity can launch a smoke cloud towards a threat or in a specific direction. Smoke clouds affect sensors. They do not affect entity movement.



Tactical smoke is delivered by the M250 smoke grenade launcher. An entity must have this system to execute this task.

To assign a Launch Smoke task:

1. Select the entity that you want to launch smoke.
2. Choose **Task** → **Engagement** → **Launch Smoke**. The Launch Smoke dialog box opens.
3. Select At Nearest Threat or Toward Compass Heading.
4. If you selected Toward Compass Heading, specify a compass heading, in degrees.
5. Click OK.

27.19. Launch Torpedo

Subsurface entities can launch two types of anti-ship torpedoes. One moves at a fixed depth; one homes in on the center of the target. In both cases, the torpedo travels at an initial bearing for a fixed distance before it homes in on the target.



Torpedoes rise to target depth at a fixed rate. If a submarine is quite close to a target when it fires, the torpedo might not be able to rise to target depth in the distance from the submarine to the target. If this happens, it circles until it reaches target depth.

27.19.1. Anti-ship (Fixed Depth)

To assign a Launch Torpedo – Anti-ship (Fixed Depth) task:

1. Select a subsurface entity.
2. Choose **Task** → **Engagement** → **Launch Torpedo** → **Anti-ship (Fixed Depth)**. The Anti-ship (Fixed Depth) dialog box opens.
3. Optionally, filter the entity list.
4. Select the target entity.
5. In the Homing Start Delay box, enter the amount of time that the torpedo should travel on its initial bearing.
6. In the Cruise Depth box, specify the depth the torpedo should travel at. It will remain at this depth.
7. In the Initial Bearing box, specify the bearing the torpedo should travel at until it begins homing in on the target.
8. In the Proximity Trigger Distance box, set the distance from a target at which the torpedo will detonate.
9. Click OK.

27.19.2. Anti-Submarine

The anti-submarine torpedo homes in on the center of the target.

To assign a Launch Torpedo – Anti-submarine task:

1. Select a subsurface entity.
2. Choose **Task** → **Engagement** → **Launch Torpedo** → **Anti-submarine**. The Anti-Submarine dialog box opens.
3. Optionally, filter the entity list.
4. Select the target entity.
5. In the Homing Start Delay box, enter the amount of time that the torpedo should travel on its initial bearing.
6. In the Initial Bearing box, specify the bearing the torpedo should travel at until it begins homing in on the target.
7. In the Proximity Trigger Distance box, set the distance from the target at which the torpedo will detonate.
8. Click OK.

27.20. Place IED

The Place IED task lets a human entity place an improvised explosive device (IED) at a particular location and move to a post-placement location. This task also sets the parameters for how the IED will be armed.

To assign a Place IED task:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Place IED**. The Place IED dialog box opens (Figure 27-4).

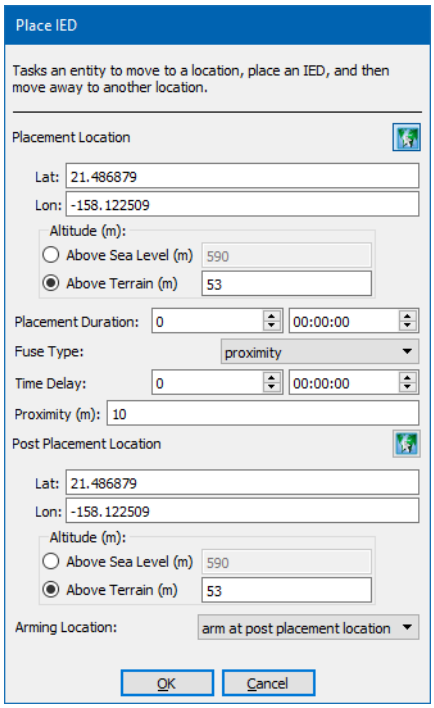


Figure 27-4. Place IED dialog box

3. Enter the location where the IED should be placed, or click on the terrain to specify the location.
4. In the Placement Duration box, enter the amount of time it takes to place the IED.
5. Select a fuse type from the Fuse Type list.
6. For a timed delay fuse, specify the time to delay in the Time Delay box.
7. For a proximity fuse, specify the distance from the IED within which another entity will trigger the IED.
8. For the Post Placement Location, specify the location where the entity should move to after it places the IED, or click on the terrain to specify the location.

9. In the Arming Location list select Arm at Placement Location to arm the IED as soon as it is placed, or Arm at Post Placement Location to arm it when the entity reaches the post-placement location.
10. Click OK.

27.21. Provide Suppressive Fire

The Provide Suppressive Fire task tells an entity to fire at a point, along a line, or in an area, one meter above the terrain. For more information, please see "Suppressive Fire Tasks," on page 25-30.



The target of suppressive fire must be at least 25 meters away.

To assign a Provide Suppressive Fire task:

1. Select the entity that will deliver suppressive fire.
2. Choose **Task** → **Engagement** → **Provide Suppressive Fire**. The Provide Suppressive Fire dialog box opens.
3. Select the tactical graphic to fire at.
4. In the Duration of Rapid Fire box, type the number of seconds for the rapid fire.
5. In the Total Fire Time box, type the number of minutes the entity should provide suppressive fire.
6. In the Maximum Round to Fire box, type the maximum number of rounds to fire.
7. In the Gun list, select the gun to use.
8. Click OK.

27.22. Provide Suppressive Fire at Location

The Provide Suppressive Fire on Location task tells an entity to fire at a location, one meter above the location. For more information, please see "Suppressive Fire Tasks," on page 25-30.



The target of suppressive fire must be at least 25 meters away.

To assign a Provide Suppressive Fire at Location task:

1. Select the entity that will deliver suppressive fire.
2. Choose **Task** → **Engagement** → **Provide Suppressive Fire at Location**. The Provide Suppressive Fire at Location dialog box opens.
3. Specify the location.
4. In the Duration of Rapid Fire box, type the number of seconds for the initial burst of rapid fire.
5. In the Total Fire Time box, type the number of minutes the entity should provide suppressive fire.
6. In the Maximum Round to Fire box, type the maximum number of rounds to fire.
7. In the Gun list, select the gun to use.
8. Click OK.

27.23. Release Bomb Tasks

Fixed-wing entities can release bombs targeted to a laser spot, a location, or an entity. The Release Bomb on Laser Spot task requires that another entity lase the target. You must synchronize the laser code for the bomber with the laser code for the lasing entity. Use of lasers is described in:

- "Lasing Targets," on page 9-18.
- "Lase Target," on page 27-13.
- "Laser Code," on page 34-24.
- "Synchronize Laser Code," on page 34-43.

The Release Bomb on Target and Release Bomb on Location tasks cause a bomb to be immediately dropped. The task assumes that the target is in front of the aircraft. The targetable area is a circle whose radius is defined by the altitude of the airplane and its maximum range.

The bomb navigates toward its target based on max-range and max-range-altitude parameters (configurable in the Simulation Object Editor). A bomb can navigate max-range meters laterally if it is at max-range-altitude meters above the target. As the altitude changes, the distance it can travel changes accordingly, so at half of max-range-altitude it can travel half of max-range.

The bomb follows a straight line path from its launch to the point it is navigating to, assuming it can reach it. If its target point is out of range, it gets as close to the target as it can. If the target point moves, the bomb changes course and heads for the new target point.



Use of a Release Bomb task requires that you determine if the airplane is in range of the target. The Execute Close Air Support task ensures that the airplane is within range of the target before it drops its bomb. Therefore, we recommend using that task rather than one of the Release Bomb tasks.

The Release Bomb on Laser Spot task steers the bomb towards the laser point.

To assign a release bomb task:

1. Select the entity that will release the bomb.
2. Choose **Task** → **Engagement** → *task*, where *task* is Release Bomb on Laser Spot, Release Bomb on Location, or Release Bomb on Target. The appropriate dialog box opens.
3. In the Bomb Types list, select the type of bomb you want to drop.
4. In the Detonation Proximity box, enter the distance from the target at which the bomb should detonate.
5. For Release Bomb on Location, specify the location.

   For Release Bomb on Target, select the target.
6. Click OK.

27.24. Strafe Ground Target

Fixed-wing entities that have the appropriate armament can strafe ground targets. The Strafe Ground Target task takes the following parameters:

- Target Point. The target point to strafe.
- Initial Point. The point at which the aircraft reaches strafing altitude and lines up to strafe the target point. VR-Forces enforces a minimum distance from the initial point to the target. It varies based on the plane's dive angle, strafe range, entry speed, and other parameters. Two kilometers is a good minimum for most entities. If the initial point is too close to the target, VR-Forces sends an error message to the entity console.
- Target Types. The type of entity or object to be targeted at the target point. If you specify none, the entity strafes the target location. If you specify an object type, the entity looks for that object type. If it finds the target type, it strafes it. If it does not find the target type, it either strafes the target point (default behavior) or aborts the strafing run, depending on the state of the Abort Without Target check box.
- Egress Maneuver and Egress Heading. The way the aircraft should exit the strafing run and the heading at which it exits.

You can also initiate a strafing run from the Execute Close Air Support task. Table 27-1 shows how arguments to the Execute Close Air Support task map to the Strafe Ground Target task.

Table 27-1: CAS to Strafe argument mapping

CAS Argument	Strafe Ground Target Argument
Initial Point	Initial Point.
Target Point	Target Point.
Target: a waypoint	Target Type: None (target location).
Target: an entity	Target Type: Target the indicated entity.
Offset: none	Egress Maneuver: Immediate Climb.
Offset: left	Egress Maneuver: Juke Left.
Offset: right	Egress Maneuver: Juke Right.
	Abort without target.
Egress Heading	Final Egress Heading.

The effectiveness of a strafing run at killing entities around the target point depends in part on the value of the **dispersal-angle** parameter specified for the munition. You can edit this parameter in *./data/simulationModelSets/base/detonationParams.mtl*. The parameter is described in the comments at the top of the file.

To assign a Strafe Ground Target task:

1. Select the entity that will make the strafing run.
2. Choose **Task** → **Engagement** → **Strafe Ground Target**. The Strafe Ground Target dialog box opens (Figure 27-5).

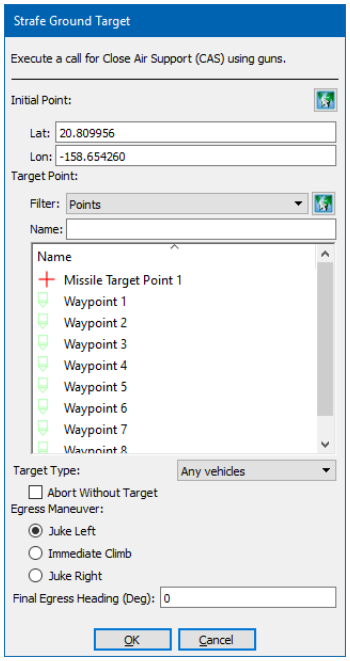


Figure 27-5. Strafe Ground Target dialog box

3. Click the terrain to specify the initial point or type the coordinates.
4. Select the target point.
5. Optionally, select a target type.
6. If you select a target type other than none, select the Abort Without Target check box to abort the strafing run if the target type is not found. Clear the check box if you want the entity to strafe the target point if the target type is not found.
7. Select an egress maneuver.
8. Optionally, specify an egress heading.
9. Click OK.

27.25. Throw Grenade Tasks

Infantry characters can throw hand grenades at a target or at a location. An entity can throw the following types of grenades:

- M67 - fragmentation.
- M84 - stun, or flashbang.
- M18 - colored smoke.
- M83 - white smoke.

27.25.1. Throw Grenade at Location

To assign a Throw Grenade at Location task:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Throw Grenade at Location**. The Throw Grenade at Location dialog box opens.
3. In the Grenade Type list, select the type of grenade to throw.
4. Specify the location.
5. Click OK.

27.25.2. Throw Grenade at Target

To assign a Throw Grenade at Target task:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Throw Grenade at Target**. The Throw Grenade at Target dialog box opens.
3. Specify the target.
4. In the Grenade Type list, select the type of grenade to throw.
5. Click OK.



28. Human and Crowd Tasks

This chapter describes the tasks for humans and crowds that apply to entity-level scenarios. Generic movement and engagement tasks that apply to humans are described in Chapter 26, *Movement Tasks for Entity-Level Scenarios* and Chapter 27, *Engagement Tasks for Entity-Level Scenarios*.

Circle Around Location 28-3
Civilian Idle 28-3
Close Door 28-3
Close Window 28-4
Create Pedestrians 28-4
Crowd Along Line 28-4
Crowd Around Location 28-5
Crowd Around Object 28-5
Crowd in Front of Entity 28-6
DI-Guy Animation 28-7
DI-Guy Gesture 28-7
Disperse Crowd 28-8
Find Cover 28-8
Find Firing Position 28-9
Flee from Entities 28-11
Move Between Cover to Location 28-11
Open Door 28-12
Open Window 28-12
Parachute to Target 28-13
Protest Around Location 28-14
Protest Along Line 28-14

Wait in Queue 28-15
Wander 28-15
Wander In Area 28-16

28.1. Circle Around Location

The Circle Around Location task causes the members of a unit or crowd to form a circle around a location. They can face in or out. This task only apples to the individually simulated members of a unit (that is, leaf members). If a unit has subordinate units, the task is not passed down to them.

If a unit or crowd has many members, using a small radius may result in overlapping entities.

To assign a Circle Around Location task:

1. Select the unit.
2. Choose **Task** → **Movement** → **Circle Around Location**. The Circle Around Location dialog box opens.
3. Specify the location.
4. Specify the direction to face.
5. Specify the radius of the circle to create.

28.2. Civilian Idle

The Civilian Idle task causes a human character to stand and cycle through DI-Guy animations typical of a person standing in a group.

To assign a Civilian Idle task:

1. Select the human characters.
2. Choose **Task** → **Movement** → **Civilian Idle**.

28.3. Close Door

If a human is within approximately one meter of an open door, this task causes the door to close. This task applies to doors controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing doors, please see Chapter 62, *Dynamic Terrain*.

To assign a Close Door task:

1. Move a human to within one meter of an open door.
2. Choose **Task** → **Terrain** → **Close Door**.

28.4. Close Window

If a human is within approximately one meter of an open window, this task causes the window to close. This task applies to windows controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing windows, please see Chapter 62, *Dynamic Terrain*.

To assign a Close Window task:

1. Move a human to within one meter of an open window.
2. Choose **Task** → **Terrain** → **Close Window**.

28.5. Create Pedestrians

The Create Pedestrians task adds pedestrians to a crowd. If the crowd is part of a pedestrian area, the new pedestrians wander within the area. If the crowd is not part of a pedestrian area, their wandering is unrestricted.



When the Create Pedestrians task runs, it gives each new pedestrian a Move To Location task, followed by a Wander task. The Create Pedestrians task is not complete until all the Wander tasks have been sent. Therefore, the task does not complete as quickly as you might expect simply based on when the pedestrians get created.

To assign a Create Pedestrians task:

1. Select a crowd or select a pedestrian area.
2. Choose **Task** → **Create Pedestrians**. The Create Pedestrians dialog box opens.
3. Specify the number of pedestrians to create.
4. Select a placement option.
5. Click OK.

28.6. Crowd Along Line

The Crowd Along Line task causes a crowd to gather to the left or right of a line at its center point.

To assign a Crowd Along Line task:

1. Select a crowd.
2. Choose **Task** → **Movement** → **Crowd Along Line**. The Crowd Along Line dialog box opens.
3. Select the line the crowd should crowd along.
4. Specify left or right of the line.
5. Click OK.

28.7. Crowd Around Location

The Crowd Around Location task causes a crowd unit to cluster around the specified location.

To assign a Crowd Around Location task:

1. Select a crowd.
2. Choose **Task** → **Movement** → **Crowd Around Location**. The Crowd Around Location dialog box opens.
3. Click on the terrain to specify the location or enter the coordinates in the dialog box.
4. Click OK.

28.8. Crowd Around Object

The Crowd Around Object task causes a crowd unit to cluster around the specified simulation object or point.



The Crowd Around Object task calculates target locations for the members of the crowd to move to that are greater than or equal to the minimum distance specified from the object. However, due to interactions between the crowd members as they move, it is possible that some will end up closer to the target object than the minimum distance.

To assign a Crowd Around Object task:

1. Select a crowd.
2. Choose **Task** → **Movement** → **Crowd Around Object**. The Crowd Around Object dialog box opens.
3. In the Filter list, select All Entities or Points.
4. Select the simulation object or point you want the crowd to gather around.
5. In the Minimum Distance box, specify the minimum distance between a member of the crowd and the object.
6. Click OK.

28.9. Crowd in Front of Entity

The Crowd in Front of Entity task causes a crowd to gather in front of an entity, where the front of the entity is determined based on its heading.

To assign a Crowd in Front of Entity task:

1. Select a crowd.
2. Choose **Task** → **Movement** → **Crowd in Front of Entity**. The Crowd in Front of Entity dialog box opens.
3. Select the entity you want the crowd to gather in front of.
4. Click OK.

28.10. DI-Guy Animation

VR-Forces can specify the movements and appearance of DI-Guy models used in the Stealth observer mode and in 3D visualization programs such as VR-Vantage Stealth. The DI-Guy Animation task directs the 3D model to engage in a one-time or repeated series of motions. VR-Forces does not simulate the actions that the 3D model performs; it just adjusts the location and orientation of the VR-Forces entity to synchronize it with the 3D model's animated behavior. To change a 3D model's appearance, use the DI-Guy Appearance set data request. For details, please see "DI-Guy Characteristics (Appearance, Head, Body Weight)," on page 34-15.



The DI-Guy Animation task applies only to lifeform entities.

To assign a DI-Guy Animation task:

1. Select the lifeform entity whose behavior you want to animate.
2. Choose **Task** → **Movement** → **DI-Guy Animation**. The DI-Guy Animation dialog box opens.
3. In the Animation list, select an animation.
4. In the Animate group box, specify the duration of the animation.
5. Click OK.

28.11. DI-Guy Gesture

The DI-Guy Gesture task lets you assign a gesture to a human character. Gestures are motions that a character makes using part of its body. By contrast, animations may involved moving from one place to another.

To assign a DI-Guy Gesture task:

1. Select the lifeform entity whose behavior you want to animate.
2. Choose **Task** → **Movement** → **DI-Guy Gesture**. The DI-Guy Gesture dialog box opens.
3. In the Gestures list, select a gesture.
4. Select the Wait Until Complete check box if you do not want the entity to start another gesture until this one is complete.
5. Select the Preview check box to see what the gesture looks like.
6. Click OK.

28.12. Disperse Crowd

The Disperse Crowd task causes a crowd to wander freely. If the crowd was created as part of a pedestrian area, this task allows it to wander outside the area. (A Wander task would restrict it to the area.) The primary purpose of this task is to disperse a crowd that has previously crowded around an entity or location.

To assign a Disperse Crowd task:

1. Select the crowd you want to disperse.
2. Choose **Task** → **Movement** → **Disperse Crowd**.

28.13. Find Cover

The Find Cover task causes a human entity to find a location from which a specified entity (the threat) cannot see it. The task takes a range, a distance from threat, and threat radius.

The range is the distance from its starting location within which the tasked entity will seek cover. The distance from threat is the minimum distance to maintain between the threat and the tasked entity. For example, you wouldn't expect the entity to run right next to the threat to reach some point on the other side of it. You would expect it to maintain some distance from the threat entity.

The threat radius defines an area around the threat from which line-of-sight calculations are made. Figure 28-1 illustrates this concept. Line-of-sight between the threat and the covered entity is blocked. However, the covered entity is still visible from a point within the threat radius.

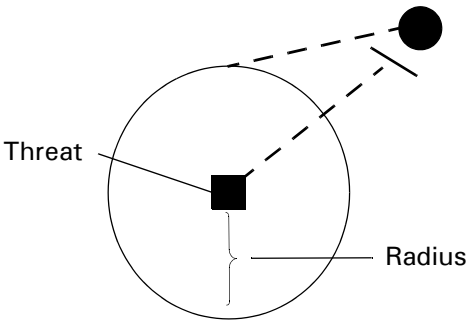


Figure 28-1. Threat radius

To assign a Find Cover task:

1. Select an entity.
2. Choose **Task** → **Movement** → **Find Cover**. The Find Cover dialog box opens.
3. Select the entity to find cover from (the threat).
4. Specify the Threat Radius.
5. Specify the Range.
6. Specify the Distance From Threat.
7. Click OK.

28.14. Find Firing Position

The Find Firing Position task causes a human entity to find a location from which a specified entity (the threat) cannot see it, but from which it can easily fire on the threat entity. Figure 28-2 illustrates this concept. The entity under cover can easily move to a location that has line-of-sight to the threat. This could be by moving to the side or looking over the top of the terrain feature it is hiding behind.

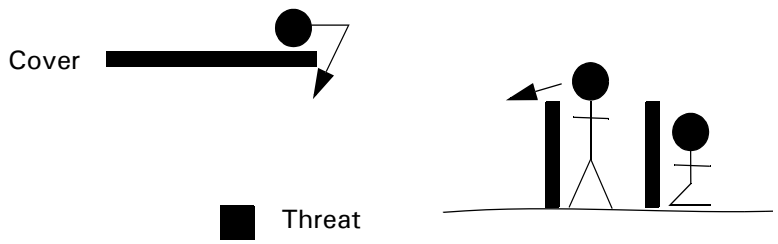


Figure 28-2. Firing position



The entity seeking cover cannot hide behind other entities. It only hides behind terrain features, such as walls or doors.

The task takes a range, a distance from threat, and a threat radius value. The range is the distance from its starting position within which the tasked entity will seek cover. The distance from threat is the minimum distance to maintain between the threat and the tasked entity. For example, you wouldn't expect the entity to run right next to the threat to reach some point on the other side of it. You would expect it to maintain some distance from the threat entity.

The threat radius defines an area around the threat from which line-of-sight calculations are made. Figure 28-1 illustrates this concept.

To assign a Find Firing Position task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Find Firing Position**. The Find Firing Position dialog box opens.
3. Select the entity to find cover from (the threat).
4. Specify the Threat Radius.
5. Specify the Range.
6. Specify the Distance From Threat.
7. Click OK.

28.15. Flee from Entities

The Flee from Entities task causes a human entity to flee from a specified entity. It flees until it is a specified distance away. This task can be assigned to individual humans and to crowds. The entities must be within a navigation area.

To assign a Flee from Entities task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Flee From Entities**. The Flee From Entities dialog box opens.
3. Select the entity that you want this entity to flee from.
4. Optionally, specify the radius of the flight zone.
5. Click OK.

28.16. Move Between Cover to Location

The Move Between Cover to Location task causes an entity to move to a location by moving between cover points (locations where it is not visible to a specified threat entity). The task requires you to specify a distance from the threat entity and a threat radius.



The entity seeking cover cannot hide behind other entities. It only hides behind terrain features, such as walls or doors.

The distance from threat is the minimum distance to maintain between the threat and the tasked entity. For example, you wouldn't expect the entity to run right next to the threat to reach some point on the other side of it. You would expect it to maintain some distance from the threat entity.

VR-Forces calculates cover points based on line-of-sight from a threat radius around the threat entity. Figure 28-1 illustrates this concept.



The default values for threat radius and distance from threat are fairly small. Unless you are playing in a congested area, you will probably want to change them.

To assign a Move Between Cover to Location task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Move Between Cover to Location**. The Move Between Cover to Location dialog box opens.
3. Click on the terrain to specify the location or enter coordinates in the dialog box.
4. Select the threat entity in the Find Cover From list.
5. In the Threat Radius box, specify the threat radius.
6. In the Distance From Threat box, specify the minimum distance to maintain between the threat entity and the moving entity.
7. Click OK.

28.17. Open Door

If a human is within approximately one meter of a closed door, this task causes the door to open. This task applies to doors controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing doors, please see Chapter 62, *Dynamic Terrain*.

To assign an Open Door task:

1. Move a human to within one meter of a closed door.
2. Choose **Task** → **Terrain** → **Open Door**.

28.18. Open Window

If a human is within approximately one meter of a closed window, this task causes the window to open. This task applies to windows controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing windows, please see Chapter 62, *Dynamic Terrain*.

To assign an Open Window task:

1. Move a human to within one meter of a closed window.
2. Choose **Task** → **Terrain** → **Open Window**.

28.19. Parachute to Target

The Parachute to Target task causes a paratrooper entity to disembark from an aircraft, deploy its parachute and try to land at the target object or location. The paratrooper tries to land as close to the target as it can given the altitude at which it disembarks and the distance to the target. If it cannot reach the target before it lands, the entity walks towards the target after it lands.



While an entity is parachuting, if you double-click it to try to move it, it snaps to the terrain.

The Air Drop task can also cause paratroopers to parachute. For details, please see "Air Drop," on page 26-3.

To assign a Parachute to Target task:

1. Select the paratrooper.
2. Choose **Task** →**Movement** → **Parachute to Target**. The Parachute to Target dialog box opens.
3. Specify a target. It can be a simulation object or a location. If you specify both, the object is used. The Target Object section automatically has focus. If you want to set a location, be sure to select the filter button first. (For details about controlling focus in dialog boxes, please see "Filtering the Object Selection Lists," on page 25-9.)
4. Specify the altitude at which to deploy the parachute. If the aircraft is below the deployment altitude, the parachute deploys as soon as the entity disembarks.
5. Click OK.

28.20. Protest Around Location

The Protest Around Location task causes a crowd to gather around a specified location and continuously execute a series of protest animations.

To assign a Protest task:

1. Select the crowd.
2. Choose **Task** → **Movement** → **Protest**. The Protest Around Location dialog box opens.
3. Specify a location.
4. Click OK.

28.21. Protest Along Line

The Protest Along Line task causes a crowd to gather along a specified line and continuously execute a series of protest animations.

To assign a Protest Along Line task:

1. Select the crowd.
2. Choose **Task** → **Movement** → **Protest Along Line**. The Protest Along Line dialog box opens.
3. Specify a line.
4. Specify which side of the line to protest on.
5. Click OK.

28.22. Wait in Queue

The Wait in Queue task causes a human to move to the beginning of a queue tactical graphic and then move along the line until it stands behind the last human in the line. (The first human in line moves to the end point of the tactical graphic, which is the beginning of the queue.) As people leave the top of the queue, the entity moves up in line.

Since this is a Wait task, the entity does not move to the next task in its plan until it is released from the queue. Entities get released from a queue when the Release Next Entity From Queue task is sent to the queue. For more information, please see "Release Next Entity From Queue," on page 29-23.



The Wait in Queue task only works with a queue tactical graphic. It does not work with other types of lines.

To assign a Wait in Queue task:

1. Select the entity.
2. Choose **Task** → **Wait** → **Wait in Queue**. The Wait in Queue dialog box opens.
3. Select the queue that the entity should wait in.
4. Click OK.

28.23. Wander

The Wander task causes a lifeform entity to move to a series of randomly chosen locations for the specified amount of time. You can constrain wandering to a specified area. If the entity is using a preferred navigation mode and there is preferred road data for the terrain, the Wander task will only choose destinations that meet the preference criteria. (For information about setting navigaiton preferences, please see "Navigation Preferences," on page 34-28.) The entity must be in a navigation area.

To assign a Wander task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Wander**. The Wander dialog box opens.
3. Select a Duration option – Indefinitely or Timed. If you select Timed, specify the amount of time to wander in days, hours, minutes, and seconds.
4. Select a Movement options – Free or Restricted To Area. If you select Restricted To Area, type the name of an area or select it in the list or on the map.
5. Click OK.

28.24. Wander In Area

The Wander In Area task causes a crowd to wander in the specified area. The entity must be in a navigation area.

To assign a Wander In Area task:

1. Select the crowd.
2. Choose **Task** → **Movement** → **Wander In Area**. The Wander In Area dialog box opens.
3. Select the area to wander in.
4. Click OK.



29. Embarkation, Wait, Radio, and Other Tasks

This chapter describes the tasks supplied with VR-Forces that are categorized as Embarkation, Radio, Wait, Other, and uncategorized.

Embarkation Tasks 29-3
Disembark 29-3
Disembark All 29-3
Embark 29-4
Embarking Humans in Cars 29-10
Radio Tasks 29-11
Send Radio Set 29-11
Send Radio Task 29-13
Send Text Message 29-14
Wait Tasks 29-15
Wait 29-15
Wait Duration 29-15
Wait Elapsed 29-15
Tasks for Moving Articulated Parts 29-16
Close Car Door 29-16
Close Cargo Door 29-16
Deploy Refueling Boom 29-16
Lower Periscope 29-17
Open Car Door 29-17
Open Cargo Door 29-18
Raise Periscope 29-18
Stow Refueling Boom 29-18
Embedded Entity Tasks 29-19
Deploying Embedded Entities 29-19
Recovering Deployed Entities 29-19
Sector Search Operation 29-20
Deploying an Entity by Assigning a Task 29-21

Deploy Sonobuoy 29-22
Deploy Sonobuoys Along Route 29-22
Sonar Dip 29-23
Release Next Entity From Queue 29-23
User Task 29-24
Reactive Tasks 29-25
Background Processes 29-26

29.1. Embarkation Tasks

This section describes tasks for simulating embarking and disembarking. Simulation objects move to the parent simulation object to embark, and move away after disembarking. For immediate embarkation and disembarkation, use the Embarked and Disembarked set data requests or the embarkation commands on the Objects menu. For a conceptual discussion of embarkation, please see "Embarking and Disembarking Simulation Objects," on page 17-2.

29.1.1. Disembark

The Disembark task causes a simulation object to move to the parent simulation object's egress (exit) point and disembark. If the selected simulation object is a unit (disaggregated), the disembark task is given to all embarked subordinates. The task is considered complete when all subordinates are disembarked.

- When a simulation object disembarks, it goes to the disembarkation location. However, it does not move away from that location. If you plan to disembark several simulation objects from the same parent by giving them disembark tasks, you are responsible for making sure that each simulation object moves from the disembarkation point before the next simulation object disembarks.
- Units in the aggregated state cannot embark or disembark. If a disaggregated unit is embarked, you cannot aggregate it either manually, or as a result of automated aggregation conditions.
- The Fast Rope Insertion task causes infantry to disembark from a helicopter. For details, please see "Fast Rope Insertion," on page 26-6.

To assign a Disembark task:

1. Select the simulation object you want to disembark.
2. Choose **Task** → **Embarkation** → **Disembark**, or click the Disembark button () on the Last Selected Object Panel or the Tasks Toolbar.

29.1.2. Disembark All

The Disembark All task is given to a parent simulation object. It gives a Disembark task to all embarked simulation objects except aggregates. When you issue a Disembark All task, VR-Forces tries to disembark the simulation objects sequentially and have them move to sufficiently dispersed locations to avoid collision problems.

To assign a Disembark All task:

1. Select the parent simulation object.
2. Choose **Task** → **Embarkation** → **Disembark All**.

29.1.3. Embark

The Embark task causes a simulation object to move to the ingress (entry) point of the parent simulation object and embark on it. If the embarking simulation object is a unit, each available taskable subordinate is given an embark task. When all subordinates have embarked, the task is considered complete.



Units in the aggregated state cannot embark or disembark.

The Embark task is valid only if the specified parent simulation object is configured to accept the simulation object that wants to embark on it. If you task a simulation object to embark on a simulation object that is not configured to receive it, for example, you task a DI to embark on an M1A2, the simulation object will not respond to the task. If you want to embark a simulation object on a simulation object that is not configured to receive it, you can use the immediate embarkation process. For details, please see "Embarking and Disembarking Simulation Objects," on page 17-2.

Human characters have a set of animations that allow them to open and close car doors and get into and out of vehicles that have interior geometry and articulated doors. For details, please see "Embarking Humans in Cars," on page 29-10.

Table 29-1 lists the simulation entities that support embarkation that have slots configured and the number and type of embarked entities they can hold. (Other simulation objects may support embarkation, but not have slots configured. Slots for humans represent passengers only, not crew members.) High fidelity entities (HF column) have polygonal models of the entity and embarkation slots defined in such a way that the embarking entities can smoothly embark on the host entity when viewed in the 3D view or VR-Vantage.

Table 29-1: Embarkation support

Entity	Can Carry	HF
Aerostat	11 humans	
AH-6 Little Bird	1 human	
Aircraft Carrier	14 fighters or 9 fighters and 2 cargo planes. Three helicopters. 70 planes below deck.	x
AirMule	2 humans, 1 cultural feature	
Arleigh Burke-class Destroyer	1 aircraft	
ASTROS II MLRS	6 humans	
ATF Dingo	4 humans	
Auto Ferry	Unlimited humans or platform entities.	
Auverland A4 AVL	4 humans	
BAZ-69092 Tow Vehicle	14 humans	
Bell 206 Jet Ranger	4 humans	
Bell 406	6 humans	
Bell V 280-Valor	11 humans	
BMP-1 AFV	7 humans	
BMP-2 AFV	7 humans	
BTR-60 Armored\ Personnel Carrier	7 humans	

Table 29-1: Embarkation support

Entity	Can Carry	HF
BTR-80 Armored\ Personnel Carrier	7 humans	
BTR-90 Armored\ Personnel Carrier	7 humans	
C-5 Galaxy	73 humans or 3 ground vehicles	
C-130 Hercules	92 humans or 3 ground vehicles	
C-17 Globemaster	134 humans or 3 ground vehicles	
CH-149 Cormorant	5 humans and one unlimited human slot	
CH-46E Sea Knight	11 humans	
CH-47 Chinook	24 humans	
CH-53E Super Stallion	55 humans	
Cadillac Gage Commando V-150	4 humans	
cargo-warfare-fixed-wing-platform	92 humans or 3 ground vehicles	
Charles De Gaulle R91	14 fighters or 9 fighters and 2 cargo planes. Three helicopters. 70 planes below deck.	
Cruise Ship	Unlimited humans	
CUCV II	4 humans	
Dardo IFV	11 humans	
Dassault Atlantic	11 humans	
DJI S1000	6 boxes	
EC 135	9 humans	
EC 145	9 humans	
EC 635	9 humans	
ESK Mungo	4 humans	
Fenneck	4 humans	
Fire Engine	6 humans	
Fleet-class USV	7 humans	
Freccia IFV	11 humans	
FV101 Scorpion_CVR	6 humans	
GAZ-69 Utility Vehicle	4 humans	
generic-cargo-helicopter-platform	55 humans	
generic-observation-helicopter-platform	6 humans	
generic-utility-helicopter-platform	11 humans	

Table 29-1: Embarkation support

Entity	Can Carry	HF
GTK Boxer APC	7 humans	
Harbin SH-5 ASW	11 humans	
HMMWV (all types)	4 humans	x
IL-76 Candid	134 humans, 3 land vehicles	
Kuznetsov-class	14 fighters or 9 fighters and 2 cargo planes. Three helicopters. 70 planes below deck.	
LAPV Enok	4 humans	
LCAC, LCAC-Landing	9 HMMWVs or 4 trucks or 1 M1A2 or 1 generic ground vehicle	
Legend Class (WMSL)	2 helicopters	
LIV (SO) Serval	4 humans	
LSD-49 Harpers Ferry	2 LCACs and 2 helicopters	x
M-939A2 5-Ton Truck	14 humans	x
M1126 Stryker ICV	9 humans	
M113 Armored Utility Vehicle	11 humans	
M2A2 Bradley IFV	6 humans	
M35 Truck	14 humans	
M58 MICLIC	4 humans	
M939A2 5-Ton Truck	14 humans	
M993_MLRS	6 humans	
MD 500	4 humans	
MH-47 Chinook	55 humans	
MH-6 Little Bird	1 human	
MH-60 Blackhawk	11 humans	
MH-60L Blackhawk DAP	11 humans	
Mi-2 Hoplite	8 humans	
Mistral Class AAS	1 helicopter	
MOWAG DURO	4 humans	
MZKT-7930 Special Wheeled Chasis	14 humans	
Nimitz Class Carrier	48 aircraft, including fighter jets, helicopters, and reconnaissance.	
OH-58 Kiowa	1 human	
P-3 Orion	11 humans	

Table 29-1: Embarkation support

Entity	Can Carry	HF
Police Cruise	2 humans	
Pomornik (Zubr)-class LCAC	9 HMMWVs or 4 trucks or 1 M1A2 or 1 generic ground vehicle	
Rigid Hull Inflatable Boat	8 humans	
Rigid Hull Inflatable Boat - Civilian	6 humans	
SA-20 Missile Transport Reload Vehicle (22T6)	2 humans	
SH-60 Seahawk	11 humans	
Skiff	7 humans	
Slava-class Guided Missile Cruiser	1 helicopter	
Space Shuttle	7 humans	
Technical Truck	4 humans	
Tractor Trailer Cab	2 humans	
Tractor Trailer with Cab	2 humans	
Type 85 (YW 531H)_ APC	7 humans	
UH-60 Blackhawk	11 humans	
UH-72 Lakota	9 humans	
URO VAMTAC	4 humans	
USCG Bertholf	5 humans, 1 helicopter	
V-22 Osprey	24 humans, 1 light ground vehicle	
VAB APC	10 humans	
WPB 87329 Cochito	Unlimited humans	
WZ551 APC	7 humans	

The aircraft carrier can carry both fixed-wing and rotary-wing aircraft. After the deck positions are used, additional entities embarked are stored below deck, as in a low-fidelity model.

- If you are using HLA and want to use the embarkation feature, you must use RPR FOM 2, draft 17 or later.
- If the embarkation spaces on the parent simulation object are full, VR-Forces sends a message to the back-end console and the simulation object does not try to embark.
- If you give multiple simulation objects embark tasks for the same parent, you are responsible for spreading the tasks over time so that the simulation objects do not all converge on the parent at the same time, which would result in problems due to collision avoidance.
- A simulation object is considered embarked as soon as it reaches its ingress point. In plan view mode, its icon is hidden immediately. However, in a 3D mode, the simulation object may need to continue moving to reach its slot on the parent. If you task the parent simulation object to move before a simulation object has moved to its slot, the embarking 3D model may end up in an odd position relative to the parent simulation object. To avoid this visual problem, allow time for the embarking simulation object to stop moving before the parent entity moves.

To assign an Embark task:

1. Select the simulation object .
2. Choose **Task** → **Embarkation** → **Embark**, or click the Embark button () on the Last Selected Object Panel or the Tasks Toolbar. The Embark On dialog box opens.



Unlike the Embark On dialog box that opens when you choose **Objects** → **Embarkation** → **Embark**, the Embark On dialog box for the Embark task does not allow you to choose the embarkation point.

3. Select the parent simulation object (the simulation object to embark on) or type its name in the Name box.
4. If the parent simulation object has slots of a specific type and you want the simulation object to embark in a slot of a particular type, select the type in the Slot Type list. Slots of that type are listed in the Slot Name list. (For information about slot types and slot names, please see "Configuring Slots," on page 72-5.)
5. Select a slot from the Slot Name list.
6. If you want the simulation object to use any available slot, select the Choose Any Slot if Requested Slot Not Available check box.

7. Click OK.

29.1.4. Embarking Humans in Cars

Some human characters have animations that allow them to open and close car doors and get into and out of cars. Humans get into and out of cars using the Embark and Disembark tasks.

When you give an entity an embark task for a car, you can specify that it embark in the driver's seat or a passenger seat. If you choose a passenger seat, you can specify which one to use. You do this by specifying the slot to embark in, as described in "Embark," on page 29-4.

If you do not specify the slot to use, the first entity that embarks in a car gets into the driver's seat. Additional entities embark in passenger seats until the number of slots available is used up.

The embark in cars capability is primarily intended for use with aerial surveillance scenarios. Viewing embarkation at ground level will not provide a completely realistic experience, including the following issues:

- You cannot change the speed at which entities enter and exit the vehicle and open and close the door. For example, they cannot run and jump into a car or jump out of a car and immediately run away.
- Contact between an entity's hand and the car door will not be entirely lifelike.
- If Show Occupants is enabled, an entity that embarks in the driver's seat will get in on top of the automatically displayed driver. We recommend you disable occupants for all cars or at least the car to be embarked on. For information about showing and hiding occupants, please see "Hide Occupants," on page 34-20 and "Displaying DI-Guys in Vehicles," on page 44-8.
- The entry and exit animations are calibrated for civilian vehicles. Trying to embark on a large vehicle, such as a truck that has doors that are high off the ground, is not taken into account.
- This release does not support drivers in the right front seat.

29.2. Radio Tasks

This section describes the tasks that let you use the VR-Forces radio network to send messages, tasks, and set data requests.

29.2.1. Send Radio Set

The Send Radio Set task lets you send a set data request to a simulation object using a VR-Forces radio message. The receiving entity executes the set data request as if you had issued it directly to the entity from the Set menu or a set data request in a plan.



Since VR-Forces has no idea what type of entity you might want to give a set data request to, when you add this task the full list of possible sets is available. There is no context sensitivity like there is when you select a simulation object and view the Set menu. It is up to you to be sure that the set you are assigning to a simulation object is supported by that object type.

To assign a Send Radio Set task:

1. Select the simulation object that will send the message.
2. Choose **Task** → **Radio** → **Send Radio Set**. The Send Radio Set dialog box opens.
3. To send the message to all simulation objects configured to receive messages from this simulation object's radio, select the Broadcast check box. (For details about configuring the radio model, please see "Configuring Radios," on page 79-4.)

   To send the message to a specific simulation object, in the Send Set To group box, in the Name text box, type the name of the simulation object that will receive the set data request message, or select the simulation object in the list or on the terrain.

   A list of set data requests is displayed (Figure 29-1).

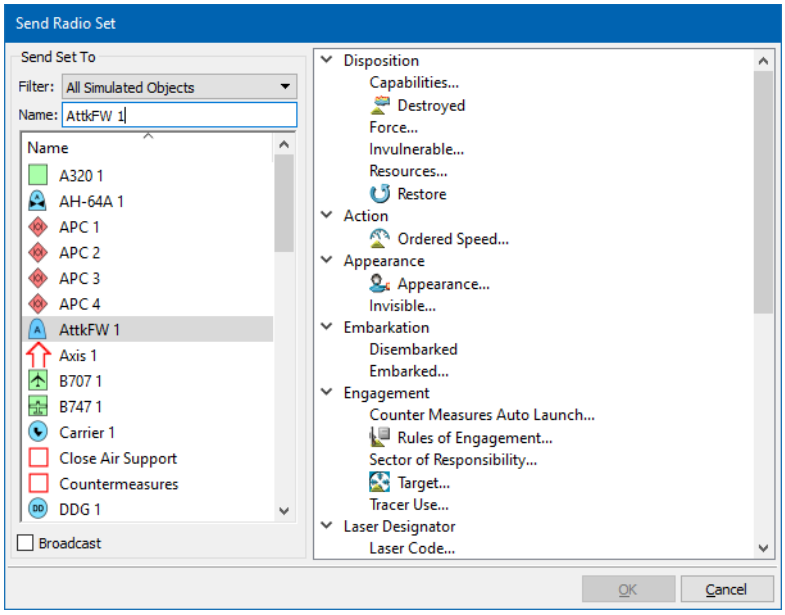


Figure 29-1. Send Radio Set dialog box

4. Select the set data request you want to send.
5. Click OK. If the set data request requires input, an appropriate dialog box opens. If no input is required, the task setup is finished.
6. If a dialog box opens, complete the dialog box as described in chapters 34 and 35.
7. Click OK.

29.2.2. Send Radio Task

The Send Radio Task task lets you send a task to a simulation object using a VR-Forces radio message. The receiving simulation object executes the task as an independent task.



Since VR-Forces has no idea what type of simulation object you might want to give a task to, when you add this task the full list of possible tasks is available. There is no context sensitivity like there is when you select a simulation object and view the Task menu. It is up to you to be sure that the task you are assigning to a simulation object is supported by that object type.

To assign a Send Radio Task:

1. Select the simulation object that will send the message.
2. Choose **Task** → **Radio** → **Send Radio Task**. The Send Radio Task dialog box opens.
3. To send the message to all simulation objects configured to receive messages from this simulation object's radio, select the Broadcast check box. (For details about configuring the radio model, please see "Configuring Radios," on page 79-4.)

   To send the message to a specific simulation object, in the Send Task To group box, in the Name text box, type the name of the simulation object that will receive the task message, or select the simulation object in the list or on the terrain.

   A list of tasks is displayed (similar to Figure 29-1).
4. Select the task you want to send.
5. Click OK. If the task requires input, an appropriate dialog box opens. If no input is required, the task setup is finished.
6. If a dialog box opens, complete the dialog box as described in the various task procedure chapters.
7. Click OK.

29.2.3. Send Text Message

The Send Text Message task lets you send a text message to a simulation object using a VR-Forces simulated radio message. The receiving simulation object must be on the same radio network as the sending simulation object. Usually this means the same unit or same force. The receiving simulation object can test for receipt of the message in a conditional statement in its plan. (For details, please see "Receive Text Message," on page 36-12.)



If you want to send a text message that can be received by simulation objects in a different radio network from the sender, use the Send Text Message command. (For details, please see "Sending Text Messages from the Commands Menu," on page 37-34.)

To assign a Send Text Message task:

1. Select the simulation object that will send the message.
2. Choose **Task** → **Radio** → **Send Text Message**. The Send Text dialog box opens.
3. To send the message to all simulation objects configured to receive messages from this simulation object's radio, select the Broadcast check box. (For details about configuring the radio model, please see "Configuring Radios," on page 79-4.)

   To send the message to a specific simulation object, in the Name text box, type the name of the simulation object that will receive the message, or select the simulation object in the list or on the terrain.
4. In the Message Text box, type the message.
5. Click OK.

29.3. Wait Tasks

You can order a simulation object to wait indefinitely, for a specified period of time, or until a specified elapsed simulation time. You can also order a human to wait in a queue. For details, please see "Wait in Queue," on page 28-15.



The Wait task is not equivalent to an immediate stop task. If a simulation object is moving and you give it a Wait task, it might not stop immediately, particularly if a controller such as the collision avoidance controller is controlling its movement.

29.3.1. Wait

To assign a Wait task:

1. Select the simulation object.
2. Choose **Task** → **Wait** → **Wait**.



If you put an indefinite wait in a plan, the simulation object does not progress through its plan until it receives another command.

29.3.2. Wait Duration

To assign a Wait Duration task:

1. Select the simulation object.
2. Choose **Task** → **Wait** → **Wait Duration**. The Wait Duration dialog box opens.
3. Specify the duration of the wait in hours, minutes, and seconds.
4. Click OK.

29.3.3. Wait Elapsed

To assign a Wait Elapsed task:

1. Select the simulation object.
2. Choose **Task** → **Wait** → **Wait Elapsed**. The Wait Elapsed dialog box opens.
3. Specify the elapsed time from the start of the simulation, in hours, minutes, and seconds.
4. Click OK.

For information about simulation time and simulated exercise time, please see "Simulation Time," on page 3-10.

29.4. Tasks for Moving Articulated Parts

Some entity models have articulated parts, such as doors and periscopes, that you can move.

29.4.1. Close Car Door

Vehicles that have a Car Door system and whose model includes articulated parts for the car doors can open and close their doors. Close Car Door closes a specific door on a vehicle.

To close a car door:

1. Select a vehicle that has an open door.
2. Choose **Task** → **Car Door** → **Close Car Door**. The Close Car Door dialog box opens.
3. In the Door list, select the door that you want to close.
4. Optionally, in the Swing Rate box, specify the speed at which you want the door to close.
5. Click OK.

29.4.2. Close Cargo Door

The Close Cargo Door task closes a cargo door on an entity that has a cargo door articulated part.

To assign a Close Cargo Door task:

1. Select the entity.
2. Choose **Task** → **Cargo Door** → **Close Cargo Door**.

29.4.3. Deploy Refueling Boom

The Deploy Refueling Boom task causes an aircraft that has a refueling boom articulated part to extend it (Figure 29-2). Use Stow Refueling Boom ("Stow Refueling Boom," on page 29-18) to retract the boom.



This task is strictly for visual effect. There is no refueling task in *EntityLevel.sms*.



Figure 29-2. KC-135 Stratotanker with refueling boom deployed

To assign a Deploy Refueling Boom task:

1. Select the entity.
2. Choose **Task** → **Refueling Boom** → **Deploy Refueling Boom**. The boom is extended.

29.4.4. Lower Periscope

The Lower Periscope task lowers the periscope of a subsurface entity. The periscope is an articulated part that is visible in the Stealth view.

To assign a Lower Periscope task:

1. Select the entity.
2. Choose **Task** → **Other** → **Lower Periscope**.

29.4.5. Open Car Door

Vehicles that have a Car Door system and whose model includes articulated parts for the car doors can open and close their doors. Open Car Door opens a specific door on a vehicle.

To open a car door:

1. Select a vehicle that has a Car Door system.
2. Choose **Task** → **Car Door** → **Open Car Door**. The Open Car Door dialog box opens.
3. In the Door list, select the door that you want to open.
4. Optionally, in the Swing Rate box, specify the speed at which you want the door to open.
5. Click OK.

29.4.6. Open Cargo Door

The Open Cargo Door task opens a cargo door on an entity that has a cargo door articulated part.



Figure 29-3. CH-53E Super Stallion with open cargo door

To assign an Open Cargo Door task:

1. Select the entity.
2. Choose **Task** → **Cargo Door** → **Open Cargo Door**.

29.4.7. Raise Periscope

The Raise Periscope task raises the periscope of a subsurface entity. The periscope is an articulated part that is visible in the Stealth observer mode.

To assign a Raise Periscope task:

1. Select the entity.
2. Choose **Task** → **Other** → **Raise Periscope**.

29.4.8. Stow Refueling Boom

The Stow Refueling Boom task causes an aircraft that has a refueling boom articulated part to stow it. Use Deploy Refueling Boom ("Deploy Refueling Boom," on page 29-16) to deploy a refueling boom.

To assign a Stow Refueling Boom task:

1. Select the entity.
2. Choose **Task** → **Refueling Boom** → **Stow Refueling Boom**. The boom contracts.

29.5. Embedded Entity Tasks



You can deploy and recover embedded entities. You can also task an embedded entity to conduct a sector search. For details about embedded entities, please see "Embedded Entities," on page 17-8.

29.5.1. Deploying Embedded Entities

To add an embedded entity to a simulation, you deploy it from the parent entity. You can deploy an embedded entity explicitly or implicitly (by assigning a task to it). An embedded entity's label displays its name and the name of its parent.



There is a delay of a few seconds from the time a deployment is ordered and the time it is executed.

To assign a Deploy task:

1. Select the parent entity.
2. Choose **Task** → **Embedded Entity** → **Deploy**. The Deploy dialog box opens.
3. Select the entity you want to deploy.
4. Click OK. The embedded entity gets deployed next to the parent entity in the relative position in its configuration.

29.5.2. Recovering Deployed Entities

If you want an embedded entity that has been deployed to return to its parent, you can have the parent recover it. If you do not recover it, the deployed entity continues to function in the scenario just like any other entity.

To assign a Recover task:

1. Select the parent entity.
2. Choose **Task** → **Embedded Entity** → **Recover**. The Recover Embedded Entity dialog box opens.
3. Select the entity you want to recover.
4. Click OK. The parent sends a message to the deployed entity to return. It returns and is embedded again.

29.5.3. Sector Search Operation

The Sector Search Operation task causes an embedded entity to fly a sector search and rescue pattern. In addition to the basic search and rescue parameters, you have the following options:

- If there are no undeployed embedded entities available, divert an already deployed entity to do the search.
- Force the parent entity to wait until the search is complete and the embedded entity is recovered until it continues with its plan or allow the parent entity to continue with its plan while the search is underway.

To assign a Sector Search Operation task:

1. Choose an entity that has embedded entities.
2. Choose **Task** → **Sector Search Operation**. The Sector Search Operation dialog box opens.
3. Select the Allow Diversion check box if you want a deployed entity to be diverted to this task if necessary.
4. Select the Recover When Finished check box if you want the parent entity to wait until it recovers the deployed entity before it continues with its plan.
5. Specify the starting point of the search.
6. In the Leg box, specify the length of each leg of the search.
7. In the Altitude box, specify the altitude the search aircraft should fly at.
8. In the Air Speed box, specify the speed of the search aircraft.
9. Click OK.

29.5.4. Deploying an Entity by Assigning a Task

You can assign a task to a deployable entity, that is, an embedded entity that has not been deployed yet. This implicitly deploys the entity so that it can execute the task. You can assign a task to an embedded entity in the parent's plan or you can directly assign a task to the embedded entity. The Sector Search example scripted task is an example of how to task a parent entity to deploy an embedded entity.



There is a delay of a few seconds from the time a deployment is ordered and the time it is executed.

To assign a task to a deployable entity in a parent's plan:

1. Select the parent entity.
2. Choose **Objects** → **Plan**. The Plan window opens.
3. Choose **Embedded Entities** → *entity* → **Task** (or **Set**) → *task* (or *set*), where *entity* is one of the entities that the parent can deploy and *task* (or *set*) is one of the tasks or sets on the Task or Set menu. An appropriate dialog box for the task or set opens.



If an entity does not have embedded entities, the Embedded Entities menu is not available.

4. Fill out the dialog box like you would for any task or set.
5. Click OK. The task is added to the plan. When you run the scenario, the embedded entity gets deployed and is assigned the task or set.

When you deploy an entity from the Objects menu, you can continue to set its state or assign tasks using this method or you can select the entity itself and issue tasks and sets from the Task and Set menus. The logic of assigning tasks from the Objects menu's Embedded Entities submenu rather than directly, is that the parent entity issues the tasks to the deployed entity.

To deploy an embedded entity by assigning it a task:

1. Select the parent entity.
2. Choose **Objects** → **Embedded Entities** → *entity* → **Task** (or **Set**) → *task* (or *set*), where *entity* is one of the entities that the parent can deploy and *task* (or *set*) is one of the tasks or sets on the Task or Set menu. An appropriate dialog box for the task or set opens.
3. Fill out the dialog box like you would for any task or set.
4. The entity is deployed. If you assigned it a task, it begins to execute its task. If you set its state, it is deployed with the updated state setting.

29.6. Deploy Sonobuoy

The Deploy Sonobuoy task lets an aircraft deploy a sonobuoy for taking sonar readings. A sonobuoy is deployed as an entity. This task is available to entities that have a Sonobuoy Deployer system configured.

> RPR FOM 1.0 does not support active sonar. You will not be able to create sonobuoys if they have an active sonar system.

To assign a Deploy Sonobuoy task:

1. Select the entity.
2. Choose **Task** → **Deploy Sonobuoy**. The Deploy Sonobuoy dialog box opens.
3. Select the type of sonar to use – Active or Passive.
4. Specify the depth at which to deploy the sonobuoy.
5. Click OK.

29.7. Deploy Sonobuoys Along Route

The Deploy Sonobuoys Along Route task lets an aircraft deploy sonobuoys as it moves along a route. A sonobuoy is deployed as an entity. This task is available to entities that have a Sonobuoy Deployer system configured.

> RPR FOM 1.0 does not support active sonar. You will not be able to create sonobuoys if they have an active sonar system.

To assign a Deploy Sonobuoys Along Route task:

1. Select the entity.
2. Choose **Task** → **Deploy Sonobuoys Along Route**. The Deploy Sonobuoys Along Route dialog box opens.
3. Select the route to use.
4. Specify the distance between each sonobuoy.
5. Select the type of sonar to use – Active or Passive.
6. Specify the depth at which to deploy the sonobuoys.
7. Click OK.

29.8. Sonar Dip

Rotary-wing entities that have sonar can dip the sonar into water at a specified depth. Dipped sonar is not visualized in the Stealth view. The sonar is just simulated at the specified depth. Entities cannot dip sonar if they are moving faster than a set speed.

To assign a Sonar Dip task:

1. Select the entity.
2. Choose **Task** → **Sonar Dip**. The Sonar Dip dialog box opens.
3. Specify the location at which to dip the sonar.
4. Specify how long the sonar should stay dipped.
5. Select the type of sonar to dip – Active or Passive.



RPR FOM 1.0 does not support active sonar.

6. Specify the depth for the sonar.
7. Click OK.

29.9. Release Next Entity From Queue

The Release Next Entity From Queue task gets sent to a queue tactical graphic and releases the entity at the head of the queue to move to the next task in its plan.

You can assign this task directly, as described in this procedure, or from the **Commands** → **Send Task** command in a plan. If you are trying to create a scenario that involves crowd queueing behaviors, then using a plan to control the queue is the more likely approach.



When an entity is released from the queue, the Wait in Queue task ends. If the entity does not have another task in its plan, it remains in its location. The next entity in the line moves to the head of the queue. This might result in some visual anomalies.

To assign a Release Next Entity From Queue task:

1. Select the queue tactical graphic.
2. Choose **Task** → **Release Next Entity From Queue**.

29.10. User Task

User tasks are tasks that have been added to VR-Forces using the VR-Forces toolkit. To assign a user task, you must know the name of the task and the required values for up to four parameters.



The User Task command is not available on the Task menu unless a developer adds a controller that accepts a user task.

To assign a user task:

1. Select the simulation object that you want to execute the task.
2. Choose **Task** → **Other** → **User Task**. The User Task dialog box opens.
3. Type the name of the task.
4. Type any required parameters.
5. Click OK.

29.11. Reactive Tasks

Reactive tasks are not listed individually because they run automatically; no user action is required to execute them. This section describes the reactive tasks included with VR-Forces.

Table 29-2 lists the reactive tasks provided with VR-Forces.

Table 29-2: Reactive tasks

Reactive Task	Description
Automatic Headlights	Turn headlights and taillights on and off based on local lighting conditions.
Automatic Jamming	Causes an entity to automatically respond to being tracked by radar.
Automatic Look	Causes humans to randomly look at other humans.
Auto-Enable Show Terrain	Turns on Show Terrain task for object upon creation.
Avoid Other Ships	Considers ship type: Aircraft carrier is higher than fishing boat (assumes nets are out), which is higher than sailboat, which is higher than other. Will avoid surfaced submarines.
Bingo Fuel Monitor	Monitors fuel on embedded entities and requests a recovery when bingo fuel level is reached.
Evade Missile	Causes fixed-wing entities to try to evade missiles.
Flee From Entity Types	Flees objects that match the configured entity types.
Flee From Explosion	Entity will run from explosions within the specified range.
IADS Launcher Control	Automates the launcher of an Integrated Air Defense system. Communicates with a control station running the IADS Command Station script.
IFF Code Changer	Automatically changes the IFF code every half hour based on the time of day.
Make Way for Emergency	If an emergency vehicle that has its lights on approaches a ground vehicle that is driving on a road, the vehicle pulls over to the side of the road.
Suppressed Crouch	Forces entity to crouch when suppressed. Returns to previous posture when no longer suppressed. Does not otherwise interrupt movement.
Suppressed Prone	Forces the entity to stop and fall prone when suppressed to level 2 or 3. Resumes movement at old posture when no longer suppressed.
Visit Interest Points	Causes civilians to visit nearby civilian visit points.

29.12. Background Processes

EntityLevel.sms has the following background processes:

- Delete Created Pedestrians. Deletes pedestrians when a pedestrian area is deleted.
- Flare Fade. Gradually fades out illumination flares.
- Queue Manager. Manages queueing for the Wait in Queue task.



30. Tasks for Aggregate-Level Scenarios

This chapter describes tasks that apply to aggregate-level scenarios.

Introduction to Aggregate-Level Tasks 30-4
Activate Jammer 30-4
Air-to-Air Attack 30-4
Air-to-Ground Air Base Attack 30-4
Air-to-Ground Attack 30-5
Attack by Fire 30-5
Attack with Anti-Air Missile 30-6
Attack with Anti-Ship Missile 30-6
Attack with Land-Attack Missile 30-6
Attack with Torpedo 30-7
Automatic Air Defense 30-7
Biological Attack 30-7
Bomb Location 30-8
Breach Obstacles 30-9
Change MOPP Level 30-10
Change Posture 30-10
Chemical Attack 30-11
Construct Abatis 30-12
Construct Anti-Tank Ditch 30-12
Construct Barbed Wire 30-13
Construct Barricade 30-13
Construct Berm 30-14
Construct Booby Trap 30-14
Construct Bridge 30-15
Construct Dry Gap 30-16

Construct Fortified Area 30-17
Construct Fortified Line 30-18
Construct Minefield 30-18
Construct Strong Point 30-19
Deactivate Jammer 30-19
Destroy Bridge 30-20
Destroy Ditch 30-20
Destroy Explosive 30-20
Destroy Fortification 30-21
Destroy Obstacle 30-21
Disembark 30-21
Disembark All 30-21
Drop Fuel Tanks 30-21
Embark 30-22
Escort Air Unit 30-22
FASCAM 30-22
Halt Movement 30-22
Improve Booby Trap 30-23
Improve Breach 30-23
Improve Bridge 30-23
Improve Ditch 30-24
Improve Fortification 30-24
Improve Obstacle 30-24
Indirect Fire 30-25
Land at Air Base 30-25
Launch Flight 30-26
Launch QRA Aircraft 30-27
Move Along Route 30-28
Move Along Route Retrograde 30-28
Move to Location 30-28
Move to Location Retrograde 30-28
Move To Waypoint 30-28
Move to Waypoint Retrograde 30-28
Nuclear Attack 30-29
Patrol Route 30-29
Patrol Between 30-29
Prepare QRA 30-30
Refuel from Tanker 30-30
Refueling Mission 30-31
Send NBC Report 30-31

Send Obstacle Report 30-32
Send Radio Set 30-32
Send Radio Task 30-32
Send Text Message 30-32
User Task 30-33
Wait for Escort 30-33
Wait Tasks 30-33
Background Process List 30-34

30.1. Introduction to Aggregate-Level Tasks

This chapter describes tasks that simulation objects in aggregate-level scenarios can run. A few tasks are identical to tasks supported by simulation objects in entity-level scenarios.

30.2. Activate Jammer

The Activate Jammer task turns on jamming for a simulation object equipped with a jamming emitter. This task has the same effect as Set Emitter - On.

➤ To activate a simulation object's jammer, choose **Task** → **Jamming** → **Activate Jammer**.

30.3. Air-to-Air Attack

The Air-to-Air Attack task instructs an air unit to attack another air unit.

To assign an air-to-air attack:

1. Select an air unit.
2. Choose **Task** → **Engagement** → **Air-to-Air Attack**. The Air-to-Air Attack dialog box opens.
3. Select the target unit.
4. Optionally, in the Maximum Engagement Distance box, specify the maximum distance from which the unit can attack the target. If it is set to 0 (zero), the maximum range is the maximum range of the weapon.
5. Click OK.

30.4. Air-to-Ground Air Base Attack

The Air-to-Ground Air Base Attack task causes a unit to move along an ingress route, move to a target point, make HE attack(s) against an air base, and then move along the egress route. The unit attacks with each different HE ammunition type in succession. For each ammunition type, the unit expends all of its ammunition. There is a 30-second delay between attacks with different ammunition types as the unit prepares to attack again.

To order and air-to-ground air base attack:

1. Select an aircraft unit.
2. Choose **Task** → **Engagement** → **Air-to-Ground Air Base Attack**. The Air-to-Ground Air Base Attack dialog box opens.
3. In the Target Air Base window, select the air base to attack.
4. Optionally, in the Part to Target list, select a specific type of target at the airbase (buildings, apron, runway, or no specific part).

5. In the Ingress Route window, select the route the unit should follow to approach the air base.
6. In the Egress Route window, select the route the unit should follow to leave the air base.
7. Click OK.

30.5. Air-to-Ground Attack

The Air-to-Ground Attack task requires an attack (ingress) route, an exit (egress) route, and a target point.

When the aircraft reaches the target point, it attacks with each type of HE ammunition it has. It expends all ammunition of each type. There is a 30 second delay between attacks with each ammunition type.

To assign an Air-to-Ground Attack task:

1. Select the unit that will attack.
2. Choose **Task** → **Engagement** → **Air-to-Ground Attack**. The Air-to-Ground Attack dialog box opens.
3. Select a target point.
4. Select an ingress route.
5. Select an egress route.



This is a multiple list dialog box. For details about how to control which list has focus, please see "Selecting Objects in Multiple List Dialog Boxes," on page 25-7.

6. Click OK.

30.6. Attack by Fire

The Attack By Fire task is for use by units created by aggregating multiple units. When assigned, the units move to the specified location. Armor and mechanized units are placed towards the front of the unit. Artillery is placed in the rear. Other units are placed between the forward and rear simulation objects. When the unit is in position, artillery begins indirect fire on the target and the other units engage any enemy simulation objects that are in range.

To assign an Attack By Fire task:

1. Select the unit.
2. Choose **Task** → **Engagement** → **Attack By Fire**. The Attack By Fire dialog box opens.
3. Specify the location of the target entity.
4. Specify a location for the attacking unit.

5. Click OK.

30.7. Attack with Anti-Air Missile

This task is available to any simulation object that has anti-air missiles.

To assign an Attack with Anti-Air Missile task:

1. Select the simulation object that will attack.
2. Choose **Task** → **Engagement** → **Attack with Anti-Air Missile**. The Attack with Anti-Air Missile dialog box opens.
3. Select a target.
4. Specify the number of missiles to fire.
5. Click OK.

30.8. Attack with Anti-Ship Missile

This task is available to any simulation object that has anti-ship missiles.

To assign an Attack with Anti-Ship Missile task:

1. Select the simulation object that will attack.
2. Choose **Task** → **Engagement** → **Attack with Anti-Ship Missile**. The Attack with Anti-ship Missile dialog box opens.
3. Select a target.
4. Specify the number of missiles to fire.
5. Click OK.

30.9. Attack with Land-Attack Missile

This task is available to entities that have land-attack missiles, such as the Arleigh Burke DDG.

To assign an Attack with Land-Attack Missile task:

1. Select the simulation object that will attack.
2. Choose **Task** → **Engagement** → **Attack with Land-Attack Missile**. The Attack with Land-Attack Missile dialog box opens.
3. Select a target.
4. Specify the number of missiles to fire.
5. Click OK.

30.10. Attack with Torpedo

This task is available to entities that have torpedoes, such as the Arleigh Burke DDG.

To assign an Attack with Torpedo task:

1. Select the entity that will attack.
2. Choose **Task** → **Engagement** → **Attack with Torpedo**. The Attack with Torpedo dialog box opens.
3. Select a target.
4. Specify the number of torpedoes to fire.
5. Click OK.

30.11. Automatic Air Defense

When a simulation object has the Automatic Air Defense task, it automatically fires at enemy aircraft that enter the specified area. A simulation object must have an anti-air weapon, such as air-to-air missile to execute this task.

To assign an Automatic Air Defense task:

1. Select a simulation object that has anti-air capability.
2. Choose **Task** → **Engagement** → **Automatic Air Defense**. The Automatic Air Defense dialog box opens.
3. Select the area in which the simulation object should automatically fire anti-air weapons.
4. Click OK.

30.12. Biological Attack

The Biological Attack task lets a simulation object create an area contaminated with biological agents.

When you assign a Biological Attack task, you specify the center of the area in which it will take place. The size of the area and the allowed range from the simulation object is specified in the biological weapon weapon system. You can edit these values in the Simulation Object Editor.

To assign a Biological Attack task:

1. Select the simulation object that will make the attack.
2. Choose **Task** → **Engagement** → **Biological Attack**. The Biological Attack dialog box opens.
3. Click a location for the center of the area of attack or specify the coordinates.
4. In the Agent Type list, select the type of biological agent.
5. Click OK.

30.13. Bomb Location

You can specify that a simulation object bomb a location. When you choose a weapon, the Bomb Location dialog box displays information about the chosen weapon (Figure 30-1).

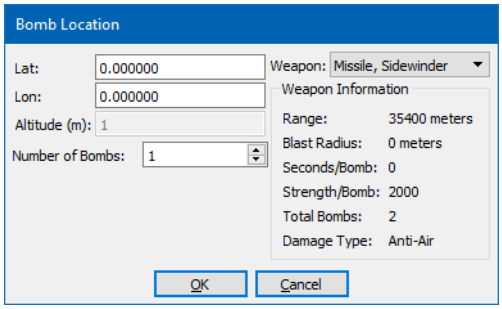


Figure 30-1. Bomb Location dialog box

To assign a Bomb Location task:

1. Select the simulation object that will attack.
2. Choose **Task** → **Engagement** → **Bomb Location**. The Bomb Location dialog box opens.
3. Click the location to bomb or enter the coordinates.
4. Specify the number of bombs to drop.
5. In the Weapon list, select the type of bomb to drop.
6. Click OK.

30.14. Breach Obstacles

The Breach Obstacles task lets an engineering unit breach an obstacle so that units can pass through it. Depending on the breaching system a unit has, it may be able to breach fortifications, minefields, or ditches.

A breach line is specified with two points.

To assign a Breach Obstacles task:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Breach Obstacles**. The Breach Obstacles dialog box opens (Figure 30-2).

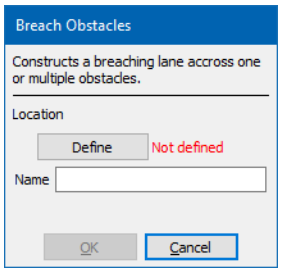


Figure 30-2. Breach Obstacles dialog box

3. Click Define. The cursor changes to draw mode and the Create Breach dialog box opens or becomes available (Figure 30-3).

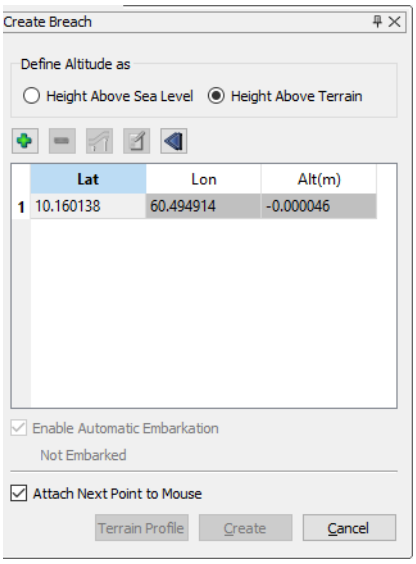


Figure 30-3. Create Obstacles dialog box

4. Draw a two-point line to specify where to create the breach. After the second click the line disappears. This is because it is not being created now. The Breach Obstacles dialog box button now says Edit and the text next to it says Defined.
5. Click OK.

30.15. Change MOPP Level

Units automatically raise their mission oriented protective posture (MOPP) level when they encounter a contamination area. However, you can manually change a unit's MOPP level.

Supported MOPP levels are as follows:

- MOPP Level 0 — Protective mask carried. Suit, gloves, and boots accessible/available.
- MOPP Level 1 — Suit worn. Mask, gloves and boots carried.
- MOPP Level 2 — Suit and boots worn. Gloves and mask carried.
- MOPP Level 3 — Suit, boots and mask worn. Gloves carried.
- MOPP Level 4 — All protection worn.

Changes to MOPP level do not take place immediately. (To change the MOPP level immediately, use Set MOPP Level.) Increasing the MOPP level degrades a simulation object's speed and combat effectiveness. Increasing the MOPP level does not remove existing contamination, but prevents most NBC contamination from harming the simulation object.

To assign a Change MOPP Level task:

1. Select the unit.
2. Choose **Task** → **Change MOPP Level**. The Change MOPP Level dialog box opens.
3. Specify a new MOPP level.
4. Click OK.

30.16. Change Posture

You can set a simulation object's posture to one of the following:

- Travel
- Reconnaissance
- Hasty attack
- Deliberate attack
- Hasty defense
- Deliberate defense.

Changes to posture take place over time. To change posture immediately, use Set Posture.

To assign a Change Posture task:

1. Select the simulation object.
2. Choose **Task** → **Change Posture**. The Change Posture dialog box opens.
3. Specify a new posture.
4. Click OK.

30.17. Chemical Attack

The Chemical Attack task lets a unit create an area contaminated with chemical agents.

When you assign a Chemical Attack task, you specify the center of the area in which it will take place. The size of the area and the allowed range from the unit is specified in the chemical weapon weapon system. You can edit this value in the Simulation Object Editor.

To assign a Chemical Attack task:

1. Select the unit that will make the attack.
2. Choose **Task** → **Engagement** → **Chemical Attack**. The Chemical Attack dialog box opens.
3. Click a location for the center of the area of attack or specify the coordinates.
4. In the Agent Type list, select the type of chemical agent.
5. Click OK.

30.18. Construct Abatis

Combat engineering units can construct abatises. An abatis is specified as a point.

> You can create abatises directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this obstacle.

To assign a Construct Abatis task:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Abatis**. The Construct Abatis dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Abatis dialog box opens (similar to Figure 30-3).
4. Click where you want to locate the abatis.
5. In the Create Abatis dialog box, click OK.
6. Optionally, specify a name for the abatis.
7. In the Construct Abatis dialog box, click OK.

30.19. Construct Anti-Tank Ditch

Combat engineering units can construct anti-tank ditches. An anti-tank ditch is a linear object. This task takes place over time.

> You can create anti-tank ditches directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this obstacle.

To assign a Construct Anti-Tank Ditch task:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Anti-Tank Ditch**. The Construct Anti-Tank Ditch dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Anti-Tank Ditch dialog box opens and the cursor changes to draw mode.
4. Click to specify the vertexes of the line as you would when creating any linear object, or add points in the dialog box and click Create.
5. Optionally, specify a name for the ditch.
6. Click OK.

30.20. Construct Barbed Wire

Combat engineering units can construct barbed wire obstacles. Barbed wire is specified as a line.



You can create barbed wire directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this obstacle.

To construct barbed wire:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Barbed Wire**. The Construct Barbed Wire dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Barbed Wire dialog box opens.
4. Click to specify the vertexes of the line as you would when creating any linear object.
5. Optionally, specify a name for the barbed wire.
6. In the Construct Barbed Wire dialog box, click OK.

30.21. Construct Barricade

Combat engineering units can construct barricades. A barricade is specified as a line.



You can create barricades directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this obstacle.

To construct a barricade:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Barricade**. The Construct Barricade dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Barricade dialog box opens.
4. Click to specify the vertexes of the line as you would when creating any linear object.
5. Optionally, specify a name for the barricade.
6. In the Construct Barricade dialog box, click OK.

30.22. Construct Berm

Combat engineering units can construct berms. A berm is specified as a line.

> You can create berms directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this obstacle.

To construct a berm:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Berm**. The Construct Berm dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Berm dialog box opens.
4. Click to specify the vertexes of the line as you would when creating any linear object.
5. Optionally, specify a name for the berm.
6. In the Construct Berm dialog box, click OK.

30.23. Construct Booby Trap

Combat engineering entities can construct booby traps. A booby trap is a location where explosives detonate if triggered by the presence of a simulation object. Booby traps cause some level of attrition on the affected simulation object. To create a booby trap, a unit must have explosive resources.

> You can create booby traps directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this object.

To construct a booby trap:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Booby Trap**. The Construct Booby Trap dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Booby Trap dialog box opens.
4. Click where you want to locate the booby trap or type the coordinates in the dialog box and click OK.
5. Optionally, specify a name for the booby trap.
6. Click OK.

30.24. Construct Bridge

Combat engineering units can construct bridges. A bridge is specified as a two-point line.



You can create bridges directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this object.

To construct a bridge:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Bridge**. The Construct Bridge dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Bridge dialog box opens and the cursor changes to draw mode.
4. Click on the terrain to specify the start and end vertices of the bridge.
5. Right-click to finish the bridge, or click Create in the Create Bridge dialog box.
6. Optionally, specify a name for the bridge.
7. Click OK.

30.25. Construct Dry Gap

Combat engineering entities can construct dry gaps. A dry gap is a type of ditch. (Use Improve Ditch and Destroy Ditch to continue building a dry gap or destroy it.)



You can create dry gaps directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this object.

To construct a dry gap:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Dry Gap**. The Construct Dry Gap dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Dry Gap dialog box opens and the cursor changes to draw mode.
4. Click to specify the vertices of the dry gap.
5. Right-click to finish the dry gap, or click Create in the Create Dry Gap dialog box.
6. Optionally, specify a name for the dry gap.
7. Click OK.

30.26. Construct Fortified Area

Combat engineering entities can construct fortified areas. A fortified area is defined as a box. You specify the southwest corner of the area. Construction of a fortified area takes place over time.



You can create fortifications directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this object.

To construct a fortified area:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Fortified Area**. The Construct Fortified Area dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Fortified Area dialog box opens and the cursor changes to show an area.
4. Click where you want to locate the southwest corner of the area or type the coordinates in the dialog box and click Create.
5. Optionally, specify a name for the area.
6. Click OK.

30.27. Construct Fortified Line

Combat engineering entities can construct fortified lines.

> You can create fortifications directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this object.

To construct a fortified line:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Fortified Line**. The Construct Fortified Line dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Fortified Line dialog box opens and the cursor changes to draw mode.
4. Click to specify the vertexes of the line as you would when creating any linear object, or add points in the dialog box and click Create.
5. Optionally, specify a name for the line.
6. Click OK.

30.28. Construct Minefield

Combat engineering entities can construct volcano minefields. Minefields are fixed size rectangles.

> You can create minefields directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this obstacle.

To construct a minefield:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Minefield**. The Construct Minefield dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Minefield (Volcano) dialog box opens and the cursor changes to show a minefield.
4. Click where you want to locate the southwest corner of the minefield or type the coordinates in the Create Minefield (Volcano) dialog box and click Create.
5. Optionally, specify a name for the minefield.
6. In the Construct Minefield dialog box, click OK.

30.29. Construct Strong Point

Combat engineering entities can construct strong points. A strong point is defined as a box. You specify the southwest corner.



You can create strong points directly from the Tactical Graphics Palette. This task is used when you want a unit to construct this object.

To construct a strong point:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Strong Point**. The Construct Strong Point dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Strong Point dialog box opens and the cursor changes to show a box.
4. Click where you want to locate the southwest corner of the box or type the coordinates in the dialog box and click Create.
5. Optionally, specify a name for the strong point.
6. Click OK.

30.30. Deactivate Jammer

The Deactivate Jammer task turns off jamming for a simulation object equipped with a jamming emitter. This task has the same effect as Set Emitter - Off.

➤ To deactivate a simulation object's jammer, choose **Task** → **Jamming** → **Deactivate Jammer**.

30.31. Destroy Bridge

When you task an engineering unit to destroy a bridge, it consumes its Explosives resource. As the unit attacks the bridge, the bridge's completeness percentage decreases. When the completion percentage reaches zero, the bridge is removed from the scenario.

To destroy a bridge:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Destroy Bridge**. The Destroy Bridge dialog box opens.
3. Select the bridge you want to destroy.
4. Click OK.

30.32. Destroy Ditch

When you task an engineering unit to destroy a ditch, it consumes its Explosives resource. As the unit attacks the ditch, the ditch's completeness percentage decreases. When the completion percentage reaches zero, the ditch is removed from the scenario.

To destroy a ditch:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Destroy Ditch**. The Destroy Ditch dialog box opens.
3. Select the ditch you want to destroy.
4. Click OK.

30.33. Destroy Explosive

The Destroy Explosive task lets a unit destroy a booby trap or unexploded ordnance.

To destroy explosives:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Destroy Explosive**. The Destroy Explosive dialog box opens.
3. Select the booby trap or unexploded ordnance you want to destroy.
4. Click OK.

30.34. Destroy Fortification

When you task an engineering unit to destroy a fortification, it consumes its Explosives resource. As the unit attacks the fortification, the fortification's completeness percentage decreases. When the completion percentage reaches zero, the fortification is removed from the scenario.

To destroy a fortification:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Destroy Fortification**. The Destroy Fortification dialog box opens.
3. Select the fortification you want to destroy.
4. Click OK.

30.35. Destroy Obstacle

The Destroy Obstacle task lets a unit destroy obstacles (such as berms, barricades, and barbed wire) that do not have a specific destroy task, such as Destroy Fortification or Destroy Explosive.

To destroy an obstacle:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Destroy Obstacle**. The Destroy Obstacle dialog box opens.
3. Select the obstacle you want to destroy.
4. Click OK.

30.36. Disembark

For details, please see "Disembark," on page 29-3.

30.37. Disembark All

For details, please see "Disembark All," on page 29-3.

30.38. Drop Fuel Tanks

The Drop Fuel Tanks task causes an air unit to drop its fuel tanks.

To drop an air unit's fuel tanks:

1. Select an air unit.
2. Choose **Task** → **Drop Fuel Tanks**.

30.39. Embark

For details, please see "Embark," on page 29-4.

30.40. Escort Air Unit

The Escort Air Unit task causes an aircraft unit to move to a location and escort the specified unit.

To assign an escort air unit task:

1. Select an air unit.
2. Choose **Task** → **Movement** → **Escort Air Unit**. The Escort Air Unit dialog box opens.
3. Select a point as the location to meet.
4. Select the unit to escort.
5. Click OK.

30.41. FASCAM

Entities with a FASCAM weapon system, such as artillery, can fire FASCAM (family of scatterable mines).

To fire FASCAM:

1. Select a simulation object that can execute this task.
2. Choose **Task** → **FASCAM**. The FASCAM dialog box opens (similar to Figure 30-2). The Location is specified as "Not defined".
3. Click Define. The Create Minefield (ADAM-RAAM) dialog box opens and the cursor changes to show a minefield.
4. Click where you want to locate the southwest corner of the minefield or type the coordinates in the Create Minefield (ADAM-RAAM) dialog box and click Create.
5. Optionally, specify a heading.
6. In the FASCAM dialog box, click OK.

30.42. Halt Movement

The Halt Movement task causes a simulation object to stop moving.

To halt a simulation object's movement:

1. Select the simulation object.
2. Choose **Task** → **Movement** → **Halt Movement**. Movement stops immediately.

30.43. Improve Booby Trap

The Improve Booby Trap task lets a combat engineering unit improve an incomplete booby trap. When you improve a booby trap it gets improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a booby trap.

To improve a booby trap:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Improve Booby Trap**. The Improve Booby Trap dialog box opens.
3. Select the booby trap that you want to improve.
4. Click OK.

30.44. Improve Breach

The Improve Breach task lets a combat engineering unit improve an incomplete breach. When you improve a breach it gets improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is breaching an obstacle.

To improve a breach:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Improve Breach**. The Improve Breach dialog box opens.
3. Select the breach you want to improve.
4. Click OK.

30.45. Improve Bridge

The Improve Bridge task lets a combat engineering unit improve an incomplete bridge. When you improve a bridge it gets improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a bridge.

To improve a bridge:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Improve Bridge**. The Improve Bridge dialog box opens.
3. Select the bridge you want to improve.
4. Click OK.

30.46. Improve Ditch

The Improve Ditch task lets a combat engineering unit improve an incomplete anti-tank ditch. When you improve a ditch it gets improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a ditch.

To improve a ditch:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Improve Ditch**. The Improve Ditch dialog box opens.
3. Select the ditch that you want to improve.
4. Click OK.

30.47. Improve Fortification

The Improve Fortification task lets a unit improve an incomplete fortification area, line, or strong point. When you improve a fortification it gets improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a fortification.

To improve a fortification:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Improve Fortification**. The Improve Fortification dialog box opens.
3. Select the fortification that you want to improve.
4. Click OK.

30.48. Improve Obstacle

The Improve Obstacle task lets a unit improve an incomplete obstacle (such as berms, barricades, and barbed wire) that does not have a dedicated improve task, such as Improve Breach or Improve Bridge. When you improve an obstacle it gets improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is constructing an obstacle.

To improve an obstacle:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Improve Obstacle**. The Improve Obstacle dialog box opens.
3. Select the obstacle that you want to improve.
4. Click OK.

30.49. Indirect Fire

The Indirect Fire task allows a simulation object to fire on a point. The indirect fire scatters bursts within a sheaf radius, which is a circle around the target point.

If multiple units are within the radius, they are all attacked. Attack strength is applied fully to each attacked simulation object. It is not divided among them. For example, if there are four units in the sheaf radius and the attack strength is 100, an attack strength of 100 is applied to each of the units.

If the blast effect radius of the munitions being fired is less than the sheaf radius, the attack strength is reduced by the ratio of the areas of the blast circle to the sheaf circle. For example, if the sheaf radius is 100 and the blast effect radius is 50, the ratio of the areas is 1:4, so the attack strength would be 25% of the full value.

To specify indirect fire for a unit:

1. Select the unit.
2. Choose **Task** → **Engagement** → **Indirect Fire**. The Indirect Fire dialog box opens.
3. Select the point on which you want to center the indirect fire.
4. In the Weapon list, select the weapon you want to use. Its characteristics are displayed.
5. In the Number of Attacks box, specify how many attacks to make.
6. In the Sheaf Radius box, specify the radius, in meters, around which to scatter indirect fire. This value must be equal to or less than the Max Sheaf Radius value shown in the Weapon Information group box for the selected weapon.
7. Click OK.

30.50. Land at Air Base

The Land at Air Base task causes an air unit to request an air base to let it land. The air unit tells the air base how many aircraft it has and how long a runway it requires.

If the request is accepted, the unit flies to the air base and requests final permission to land. The air base absorbs the resources of the unit and deletes it from the scenario. For complete details about how landing works, please see "Landing Air Units," on page 22-27.

To land an aircraft unit at an air base:

1. Select an aircraft unit.
2. Choose **Task** → **Movement** → **Land at Air Base**. The Land at Air Base dialog box opens.
3. Select the air base you want the unit to land at.
4. Click OK.

30.51. Launch Flight

An air base uses the Launch Flight task to configure and launch a flight of aircraft. A flight can be launched to execute a single unit mission or it can be one of several flights being launched as part of a multi-unit mission. If you want to set up a multi-unit mission, you must create a unit independently of the air base to act as the mission superior. For information about missions, please see "Air Missions," on page 22-22.

When you assign this task, aircraft of the indicated type are taken from the available aircraft at the base. The air base determines the munitions and fuel required by the flight given the loadout and provides them from its stores. Air crews are supplied from the available air crews. If the air base does not have enough of any of these resources, the task fails and the flight is not scheduled to launch.

When the flight is launched, the aircraft and crews are removed from the air base.

A Launch Flight task takes values for the following parameters:

- Call sign.
- Aircraft type.
- Number of aircraft in the flight.
- Loadout.
- Mission superior.
- Mission type. The mission can be one of the following:
  - BAI. Battlefield air interdiction.
  - CAP. Combat air patrol.
  - CAS. Close air support.
  - Escort.
  - Intercept.
  - OCA. Offensive counter air (bombing air bases).
  - Recon. Reconnaissance.
  - SEAD. Suppression of enemy air defense.
  - Sweep. Sweep enemy aircraft from the air space where you are going to attack.
  - Tanker.

  The mission type is informational. It does not influence the behavior of the flight.
- Launch time. This can be immediate or at a specified date and time.
- Wait for launch. Global plans only. If selected, the global plan does not move to the next task until the flight is launched.

To launch a flight:

1. Select an air base.
2. Choose **Task** → **Launch Flight**. The Launch Flight dialog box opens.
3. Specify a call sign for the flight. It can be any text.
4. Select the aircraft type. It must be a type of aircraft that is already assigned to the air base. The list of aircraft types may include aircraft that are not assigned.
5. Specify the number of aircraft in the flight.
6. The Unit Type is not editable. It is taken from the aircraft type.
7. Select a loadout from the Loadout list.
8. Optionally, specify a mission superior if this flight is subordinate to another one.
9. In the Mission Type list, select the type of mission this is.
10. Select Launch as Soon as Aircraft are Available or specify a date and time for the launch.
11. Optionally, for global plans only, specify that the global plan not proceed to the next task until the flight is launched (Wait for Launch check box).

30.52. Launch QRA Aircraft

The Launch QRA Aircraft task creates an air unit and launches the specified number of aircraft that are on Quick Reaction Alert (QRA) status.

To launch a QRA unit:

1. Select an air base.
2. Choose **Task** → **Launch QRA Aircraft**. The Launch QRA Aircraft dialog box opens.
3. Specify a call sign for the unit that is to be created.
4. Specify the number of aircraft to include in the unit.
5. Click OK.

30.53. Move Along Route

For details, please see "Move Along Route," on page 26-12.

30.54. Move Along Route Retrograde

Move Along Route Retrograde is the same as Move Along Route, except that the simulation object does not change its heading to the direction of movement. It maintains the heading at the start of movement or a heading necessary for defensive operations. See the Move Along Route task for the procedure for this task.

30.55. Move to Location

For details, please see "Move to Location," on page 26-15.

30.56. Move to Location Retrograde

Move to Location Retrograde is the same as Move to Location, except that the simulation object does not change its heading to the direction of movement. It maintains the heading at the start of movement or a heading necessary for defensive operations. See the Move to Location task for the procedure for this task.

30.57. Move To Waypoint

For details, please see "Move to Waypoint," on page 26-18.

30.58. Move to Waypoint Retrograde

Move to Waypoint Retrograde is the same as Move to Waypoint, except that the simulation object does not change its heading to the direction of movement. It maintains the heading at the start of movement or a heading necessary for defensive operations. See the Move to Waypoint task for the procedure for this task.

30.59. Nuclear Attack

The Nuclear Attack task lets a simulation object create an area contaminated with nuclear radiation.

When you assign a Nuclear Attack task, you specify the center of the area in which it will take place. The size of the area and the allowed range from the simulation object is specified in the nuclear weapon's weapon system. You can edit this value in the Simulation Object Editor.

To execute a nuclear attack:

1. Select the simulation object that will make the attack.
2. Choose **Task** → **Engagement** → **Nuclear Attack**. The Nuclear Attack dialog box opens.
3. Click a location for the center of the attack area or specify the coordinates.
4. In the Agent Type list, select the type of radiation.
5. Click OK.

30.60. Patrol Route

For details, please see "Patrol Route," on page 26-23.

30.61. Patrol Between

For details, please see "Patrol Between," on page 26-23.

30.62. Prepare QRA

The Prepare QRA task assigns aircraft to quick reaction alert (QRA) status. You specify the type of aircraft, loadout, when they should be prepared, and their response time. Once the aircraft are prepared they are available to be launched.

To prepare aircraft for QRA status:

1. Select an air base.
2. Choose **Task** → **Prepare QRA**. The Prepare Aircraft for QRA Status dialog box opens.
3. In the Aircraft Type list, select an aircraft type.
4. In the Number list, select the number of aircraft.
5. In the Loadout list, select a loadout.
6. If you want aircraft to be prepared as soon as they are available, select the Prepare As Soon As Aircraft Are Available check box. If you want them prepared at a specific time, clear the check box and in the Start Time box, specify the time when you want the preparation process to start.
7. Select a Response Time option.
8. Click OK.

30.63. Refuel from Tanker

The Refuel from Tanker task causes an aircraft unit to request refueling from a fuel tanker. A message is sent to the tanker indicating the amount of fuel required and the type of connection. The task then performs a series of steps as follows:

a. The aircraft unit flies to a standby position near tanker and follows the tanker.
b. The aircraft unit follows the tanker at range 0, and refuels.
c. The aircraft unit moves away from tanker and follows from a small distance.

Each of these steps is performed in response to a message from the tanker indicating that the tanker is ready. During this time the receiving aircraft unit also provides the tanker information about the number of aircraft in the unit and the maximum transfer rate for the aircraft.

If the receiving aircraft unit is an echelon containing subordinate units, then each subordinate will coordinate with the tanker independently.

To refuel an aircraft unit:

1. Select the unit.
2. Choose **Task** → **Combat Service Support** → **Refuel from Tanker**. The Refuel from Tanker dialog box opens.
3. Select the tanker you want to refuel from.
4. Click OK.

30.64. Refueling Mission

The Refueling Mission task causes a tanker aircraft to follow a route or move to a waypoint and fly a holding pattern. It awaits requests from other aircraft to refuel.

To issue a refueling mission:

1. Select an aircraft that has refueling capability.
2. Choose **Task** → **Combat Service Support** → **Refueling Mission**. The Refueling Mission dialog box opens.
3. Specify the altitude at which to conduct the refueling mission.
4. Specify how much fuel to reserve for the tanker's own use. If the fuel level drops to this amount, the tanker will stop refueling aircraft.
5. Specify a route to follow or specify the parameters for a race track holding pattern.
6. Click OK.

30.65. Send NBC Report

An NBC report is the equivalent of sending a spot report when an NBC area is found. NBC reports are sent automatically if a simulation object has an NBC sensor. However, this task allows you to send a report manually.

To send an NBC report:

1. Select a simulation object.
2. Choose **Task** → **Send NBC Report**. The Send NBC Report dialog box opens.
3. Select the contamination area for which to send a report.
4. Specify the Engagement Result. This is any text that you want to accompany the report.
5. Click OK.

30.66. Send Obstacle Report

An obstacle report is the equivalent of sending a spot report when an obstacle area is found. Obstacle reports are sent automatically if a simulation object has an engineering object sensor. However, this task allows you to send a report manually.

To send an obstacle report:

1. Select a simulation object.
2. Choose **Task** → **Send Obstacle Report**. The Send Obstacle Report dialog box opens.
3. Select the obstacle for which to send a report.
4. Specify the Engagement Result. This is any text that you want to accompany the report.
5. Click OK.

30.67. Send Radio Set

For details, please see "Send Radio Set," on page 29-11.

30.68. Send Radio Task

For details, please see "Send Radio Task," on page 29-13.

30.69. Send Text Message

For details, please see "Send Text Message," on page 29-14.

30.70. User Task

For details, please see "User Task," on page 29-24.

30.71. Wait for Escort

You can instruct an aircraft unit to move to a waypoint and wait for an escort.

To assign a Wait for Escort task:

1. Select an aircraft unit.
2. Choose **Task** → **Movment** → **Wait for Escort**. The Wait for Escort dialog box opens.
3. Select the waypoint that the unit should move to.
4. Click OK.

30.72. Wait Tasks

You can specify unconditional wait, wait for a duration, or wait until a simulation time. For details, please see "Wait Tasks," on page 29-15.

30.73. Background Process List

Background processes are not described in detail because they run automatically; no user action is required to execute them.

Table 30-1 lists the background tasks provided. These tasks are only available for units in *AggregateLevel.sms*.

Table 30-1: Background processes

Background Process	Description
Attack Enemies	Determines when to begin and stop attacking enemies.
Attrit Air Base	Handles air-to-ground attacks against an air base.
Calculate Maximum Speed Modifier	Calculates a modifier that is applied to the maximum possible speed of a unit to modify its movement rate.
Make EW Attack	If a jammer system is turned on, attack all non-friendly simulation objects within range of the jammer. The simulation object attacks continuously.
Manage Air Base	Launches scheduled flights and updates resources for an air base.
Perform Attrition	Determines when a unit is being attacked and modifies its health based on the current attrition rate.
Provide Supplies	Continually checks to see if conditions allow providing supplies (not in combat, not ordered to stop providing supplies). Checks to see what friendly units are in range, and sends each new unit an influence to let it know that this unit is available. When a unit goes out of range, or this unit is ordered to stop providing supplies, it sends a stop influence to that unit.
Receive Construction	Allows an engineering object to receive construction from other simulation objects.
Resize Footprints	Updates the size of a simulation object's physical and sensor footprints.
Update Combat Effectiveness	Updates the combat effectiveness state of a simulation object.
Update EW Degradation	Processes jamming interactions and computes the level of degradation in communications and radar from these attacks.
Use Supplies	Uses supplies based on the simulation object's current activities.



31. Creating Scripted Tasks and Sets

VR-Forces lets you create new tasks and set data requests by writing scripts. You do not need to know C++ or have a developers license to write these scripts. This chapter explains how to create and edit script metadata and how to organize and enable scripts. Chapter 32, *Writing Scripts* explains how to write script code. Chapter 33, *Using Qt Designer for Script Dialog Boxes* explains how to use Qt Designer to lay out dialog boxes.

Introduction to Scripts 31-3
- Scripted Tasks 31-4
- Scripted Sets 31-5
- Reactive Task Scripts 31-5
- Background Process Scripts 31-5
- Script Metadata 31-5
- The Lua Scripting Language 31-5

Creating a New Script 31-6
- Specifying the Script ID 31-11
- Specifying a Menu Icon 31-12
- Specifying an Extended Name 31-12
- Specifying a Short Description 31-12
- Specifying a Description 31-13
- Specifying Script Parameters 31-14
- Specifying a Menu Location 31-23
- Specifying Action Categories 31-24
- Configuring a Script's Availability to a Simulation Object Type . 31-25
- Specifying Equivalent Scripts for a Toolbar Button 31-29
- Specifying the Programming Language for a Script 31-30
- Configuring Scripted Tasks to Run Concurrently 31-31
- Creating Scripted Sets 31-31
- Creating Reactive Tasks 31-33
- Creating Background Processes 31-34

Saving Scripts 31-36

Editing a Script 31-37
Filtering the List of Scripts 31-37
Organizing Scripts into Folders 31-38
Adding a Folder 31-38
Renaming a Folder 31-38
Deleting a Folder 31-38
Adding Scripts to a Folder 31-38
Removing a Script from a Folder 31-38
Exporting and Importing Scripts 31-39
Importing a Script Package 31-40
Copying a Script 31-41
Deleting a Script 31-41
Creating a System Script 31-42
Including System Scripts on the Task and Set Menus 31-42
Specifying a Script Editor 31-44
Editing Lua Files 31-44

31.1. Introduction to Scripts

The script capability in VR-Forces allows you to create new tasks and set data requests for simulation objects without using the VR-Forces Toolkit and C++ API. Scripts use the Lua programming language. The actions that simulation objects can carry out are still limited to the basic C++ tasks provided by VR-Forces. However, since Lua is a full-featured scripting language, it is far more powerful than the language used to create plans and you can be much more creative in how you use the basic tasks. In addition to functions for tasks and sets, Lua scripts can make many types of calls to the simulation engine to get simulation object status or other simulation data. Scripts can also set some simulation data.

Scripts are classified as system scripts or as scenario scripts. System scripts are saved as part of a simulation model set (SMS) and are available to all scenarios that use that SMS. Scenario scripts are available only in the scenario for which the script was created. However, you can save a scenario script as a system script if you want to and, just as with other data used by VR-Forces, there are ways of copying scripts from one VR-Forces installation to another.

VR-Forces supports the following types of scripts:

- Task. Scripted tasks can be listed on the Task menu and used as independent tasks or in plans just like C++ tasks.
- Reactive task. A reactive task monitors the simulation and gets activated when the conditions set in the script become true. Reactive tasks are enabled or disabled for simulation objects through the GUI.
- Background process. A background process runs continuously from the time a simulation object is created. There is no user interface for assigning background processes to simulation objects. Their use is based entirely on the scripting process.
- Set data request. Set data request scripts change simulation object state without interrupting the current task.

From a Lua script you can:

- Create and delete simulation objects and tactical graphics.
- Set weather and environment state.
- Get simulation object state, such as:
  - Aggregation
  - Location
  - Embarkation
  - Type
  - Force
  - Damage state
  - Velocity.
- Monitor and set simulation object resources.
- Get simulation object sensor contacts.
- Task other simulation objects.
- Subtask self.
- Send Set Data Request messages.
- Test terrain:
  - Get terrain height.
  - Check line of sight.
  - Find feature data.

VR-Forces includes a text editor (SciTE) to use as the default script editor. You can configure VR-Forces to use a different editor if you want to. For details, please see "Specifying a Script Editor," on page 31-44.

31.1.1. Scripted Tasks

Scripted tasks can be added to the Task menu and are treated just like the tasks that are built into VR-Forces.

When you create a scripted task, you can specify input parameters. VR-Forces can automatically build a dialog box for users to specify the parameter values or you can use Qt Designer to lay out the dialog box. For more information, please see Chapter 33, *Using Qt Designer for Script Dialog Boxes*.

31.1.2. Scripted Sets

Scripted sets can be added to the Set menu and are treated just like the set data requests that are built into VR-Forces.

When you create a scripted set, you can specify input parameters. VR-Forces can automatically build a dialog box for users to specify the parameter values or you can use Qt Designer to lay out the dialog box. For more information, please see Chapter 33, *Using Qt Designer for Script Dialog Boxes*.

A scripted set cannot issue tasks to the entity that executes it or to any other entity.

31.1.3. Reactive Task Scripts

Reactive tasks monitor a simulation and get activated when the conditions set in the script become true. Reactive tasks are not added to the Task menu. In most respects the process for creating and writing reactive tasks is the same as creating and writing all scripts. Assume that all details about scripts apply to reactive tasks unless noted otherwise. For conceptual details about reactive tasks, please see "Reactive Tasks," on page 25-11.

31.1.4. Background Process Scripts

The background process is a script that runs in the background for the simulation object types for which it was written. Since a background process has no user interface, it does not have any input parameters. When you create a background process, you specify the simulation object types for which it is available and you write the script.

31.1.5. Script Metadata

A script has two major components, the script code, and the script metadata. The metadata provides all of the information that VR-Forces needs to store the script, add it to a menu, and generate a dialog box.

31.1.6. The Lua Scripting Language

Lua is a lightweight scripting language used by VR-Forces to implement the script feature. VR-Forces includes a set of Lua functions that map the built-in tasks and set data requests as well as the ability to obtain simulation object state information. You use these functions as the starting point for building new tasks. For details about the Lua language, please see the Lua web site (www.Lua.org).

31.2. Creating a New Script

To create a new script, you specify the metadata for the script and write the script code. A script's metadata specifies the script's name, where it will be placed on a menu, and its parameters, if any. Specifying metadata may be an iterative process, particularly the parameters, which may change as you write the code for the script. However, the minimum first step in creating a script is to specify a name for the script and add it to the list of scripts. Once you do this, you can edit the metadata and the code as needed until you have completed the script to your satisfaction.

All new scripts are created as scenario scripts. If you want to create a new system script, you save the scenario script as a system script. For details, please see "Creating a System Script," on page 31-42.



To create a script, you must have a scenario open. This is true even if you plan to create a system script rather than a scenario script.

To create a new script:

1. Open a scenario. (If you want to create a system script, you must create it in the context of a scenario. Then you will promote it to a system task.)
2. Choose **Simulation** → **Scripts**. The Scripts dialog box opens (Figure 31-1).

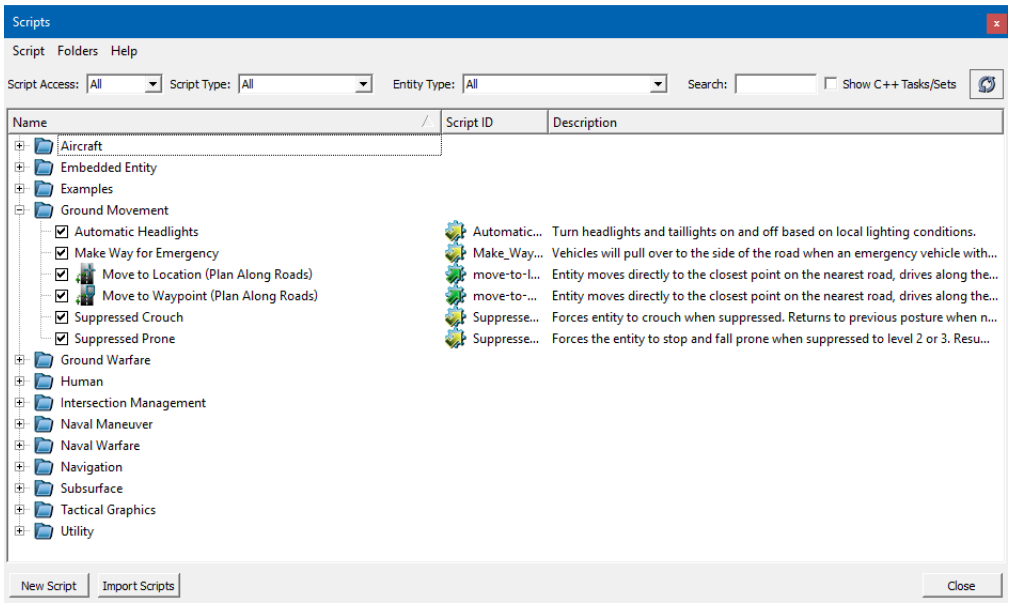


Figure 31-1. Scripts dialog box

3. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, please see "Organizing Scripts into Folders," on page 31-38.)
4. Choose **Script** → **New Script** or click the New Script button. The New Script dialog box opens (Figure 31-2).

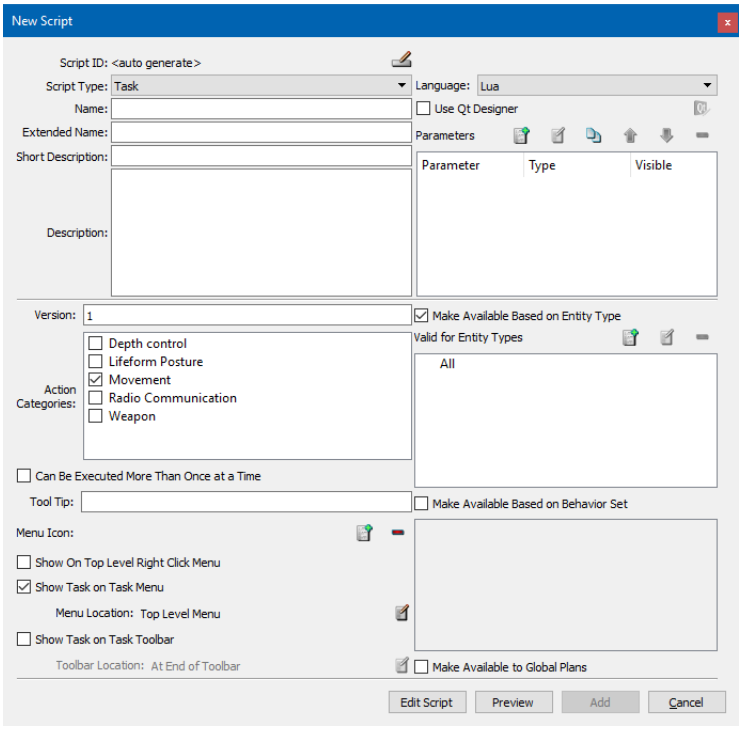


Figure 31-2. New Scripted Task dialog box

5. In the Script Type list, select the type of script you are creating (Task, Reactive Task, Set, or Background Process). The dialog box displays the correct set of metadata for the type of script you chose.

6. Complete the fields in the dialog box. They are described in Table 31-1 and in following sections.

Table 31-1: Script metadata

Input Field	Description
Script ID	A unique name for the script. VR-Forces generates a default name, but you will probably want to change it to something meaningful. It is your responsibility to make sure that it is unique. Please see "Specifying the Script ID," on page 31-11 for details.
Script Type	One of the following: ◆ Task. ◆ Reactive Task. ◆ Background Process. ◆ Set. The fields on the New Script dialog box vary based on which type of task is being created.
Language	Specify Lua if you are writing a Lua script. Specify C++ if you have written a new task controller and want to use the script interface to create the GUI portion of the task. (Does not apply to reactive tasks and background processes.) For more information, please see "Specifying the Programming Language for a Script," on page 31-30.
Name	The text that will be added to the Task or Set menu for this script.
Extended Name	An alternative name to use on context-insensitive task or set lists if more than one task or set has the same name. For details about extended names, please see "Specifying an Extended Name," on page 31-12.
Short Description	The text to use for this script in a Plan window.
Description	A description of what the script does. This text is displayed on the dialog box. Providing a description is optional. Also, optionally you can choose to include the description on the task or set dialog box.
Version	Script version. Available as a parameter to a Lua script. For use by script writers who want to have different versions of a script.
Action Categories	For tasks, specifies the categories of tasks that this scripted task will interrupt. (In other words, action categories manage task concurrency. For details about concurrent tasks, please see "Concurrent Task Execution," on page 25-4.) (Does not apply to background processes or sets.)

Table 31-1: Script metadata

Input Field	Description
Can Be Executed More Than Once at a Time	Allows an entity to manage more than one instance of this task at a time. For details, please see "Configuring Scripted Tasks to Run Concurrently," on page 31-31.
Tool Tip	Text for a tooltip when the script is selected on a menu. (Optional) (Does not apply to reactive tasks and background processes.)
Menu Icon	The icon to display next to the name on a menu. Please see "Specifying a Menu Icon," on page 31-12 for details. (Optional) (Does not apply to reactive tasks and background processes.)
Parameters	The input parameters that the task or set requires. Please see "Specifying Script Parameters," on page 31-14 for details. (Optional) (Does not apply to background processes.)
Make Available Based on Entity Type	If selected, the script is available to the simulation object types in the Valid for Entity Types list.
Valid for Entity Types	The simulation object types that can use this script. Editable only if Make Available Based on Entity Type is selected. This entry affects the context sensitivity of the Task or Set menu. Please see "Specifying the Valid Simulation Object Types for a Script," on page 31-26 for details. Default: All.
Make Available Based on Behavior Sets	If selected, the script is available to the simulation object only if the selected behavior set is assigned to its force.
Make Available to Global Plans	If selected, the script is available on the Task or Set menu in global plans.
Behavior Sets	Select each behavior set that you want this script to be part of. You can also assign scripts to behavior sets in the Edit Behavior Sets dialog box. For details about behavior sets, please see "Using Behavior Sets to Manage Scripts," on page 25-18.
Show on Top Level Right Click Menu	Select to show the task or set on the top level of the shortcut menu. This setting can be in addition to showing the script on the Task or Set menu.
Show Task on Task Menu	Select to show the scripted task on the Task menu. Clear to hide the task. (You might want to write a script that will be used by other scripts, but which you do not want a user to assign directly to a simulation object.) (For task scripts only.)

Table 31-1: Script metadata

Input Field	Description
Show Set on Set Menu	Select to show the scripted set data request on the Set menu. Clear to hide the set. (You might want to write a script that will be used by other scripts, but which you do not want a user to assign directly to a simulation object.) (For set scripts only.)
Menu Location	The location on the Task or Set menu where this script will be listed. Please see "Specifying a Menu Location," on page 31-23 for details. (Does not apply to reactive tasks and background processes.)
Show Task on Task Toolbar	Select to add this task to the Task Toolbar. The toolbar displays the name of the script. If you want to display an icon, select a Task menu icon for this task, as described in "Specifying a Menu Icon," on page 31-12. (Tasks only.)
Show Set on Set Toolbar	Select to add this set data request to the Set Toolbar. The toolbar displays the name of the script. If you want to display an icon, select a menu icon for this task, as described in "Specifying a Menu Icon," on page 31-12. (Sets only.)
Toolbar Location	The location on a toolbar for this script. You can assign more than one script to a toolbar button. For details, please see "Specifying Equivalent Scripts for a Toolbar Button," on page 31-29.
Enabled by Default	Reactive tasks only. If selected, the reactive task is automatically enabled for all supported simulation object types.
Default Priority	Reactive tasks only. The priority for this task. For information about the effect of reactive task priority, please see "Reactive Tasks," on page 25-11.
View Like Priorities	Reactive tasks only. Displays a list of the reactive tasks that are valid for this task's simulation object types and action categories. This helps you decide what priority you want to give to the reactive task you are creating or editing.

7. Optionally, click Preview to see the dialog box that the script will create.
8. Optionally, click Edit Script to begin writing the code for this script.
9. Click Add. The script is added to the list in the Scripts window.
10. To save the script task as part of the scenario, save the scenario.
11. Write the code for the script. For details, please see Chapter 32, *Writing Scripts*.

31.2.1. Specifying the Script ID

Each script must have a unique ID. VR-Forces generates a default ID, but it does not contain meaningful text. Therefore, you may want to change the ID.

The script ID is used internally by VR-Forces to manage how a script is executed. If a script is used in a plan, it is referenced by its script ID. If you change the ID, you must ensure that it is unique.

To specify the script ID:

1. In the New Script (or Edit Script) dialog box, click the button to the right of the Script ID. The New Script ID dialog box opens.
2. Type the ID you want to use. Internally, VR-Forces converts all script IDs to lower case. Therefore it is recommended that you use lower case when creating a script ID.
3. Click OK.

Editing a Script ID

Editing script IDs is not recommended because it can invalidate scripts that reference the changed script. However, you can change a script ID if you want to.

If you change the script ID for a scenario script, you are responsible for ensuring that scripts that reference it are changed accordingly.

If you change the script ID for a system script, VR-Forces generates an alias so that scripts that reference the old script ID will continue to work.

To edit a script ID:

1. Open the script whose ID you want to change.
2. Click the button to the right of the Script ID. The New Script ID dialog box opens.
3. Type the ID you want to use. Internally, VR-Forces converts all script IDs to lower case. Therefore it is recommended that you use lower case when creating a script ID.
4. Click OK.

31.2.2. Specifying a Menu Icon

If you want to put an icon on the Task or Set menu next to the menu text, you can specify one. (Does not apply to reactive tasks and background tasks.)

To specify a menu icon:

1. Click the Add button () next to the Menu Icon label. The Select Menu Icon dialog box opens.
2. Browse the directory structure and select the icon that you want to use.
3. Click Open.

➤ To remove a menu icon for a script, click the Remove button () next to the Menu Icon label.

31.2.3. Specifying an Extended Name

Sometimes a script needs to be implemented differently for different types of simulation objects. To accomplish this you can write different scripts for each type of simulation object, but give the task the same name. The scripts are distinguished by their script IDs and VR-Forces picks the correct script using context sensitivity. For example, you might have two scripts with the IDs Move_To_Waypoint_Entity and Move_To_Waypoint_Unit. Both would have the name Move to Waypoint. When you wanted to give a Move to Waypoint task, you would select the simulation object and choose Move to Waypoint from the Task menu. VR-Forces would pick the correct version of the task.

The one problem with this system is that in cases where VR-Forces does not enforce context-sensitivity of task or set lists, you would end up with two tasks called Move to Waypoint and no way to know which was the correct one to use for the simulation object that you wanted to assign the task to. The extended name solves this problem. If you specify an extended name for a script, it is used to distinguish between scripts that use the same task or set name.

31.2.4. Specifying a Short Description

When you add a scripted task or set data request to a plan, the entry in the Plan window can be lengthy and cryptic. It often includes information, such as location values that are not easily understood or put to use by a reader. The Short Description field for a script lets you specify a short, meaningful description to be used in the Plan window.

The description can include variable information. To include variable data, use the syntax *$variable_name*, where *variable_name* is one of the variables for the script. For example, the default listing for the Move to Waypoint (Plan Along Roads) task is:

```
move-to-waypoint-path-plan: destination=Waypoint1
```

If you specify a short description of Move to $destination (Plan Path), the listing in the Plan window would be:

```
Move to Waypoint1 (Plan Path)
```

In this case, the value of the destination parameter replaces the text $destination in the description.

31.2.5. Specifying a Description

You can include a long description of the script. You can do this solely for the benefit of someone examining the metadata in the Edit Script dialog box or you can include the text of the description in the dialog box that gets generated for the scripted task or set. This is a way of providing context-sensitive help to the person who must specify the parameter values for the task or set.

To specify a task or set description:

1. In the Description box, type a description of the task or set.
2. If you want the description to appear on the generated dialog box, select the Description check box.

31.2.6. Specifying Script Parameters

Scripted tasks and sets can take input parameters. When you specify parameters for a script, VR-Forces automatically builds a dialog box that lets you enter the parameter values. The parameter types that you can specify for a script are as follows:

- Active Sonar Mode. Lets you select the sonar mode to use.
- Aggregate Formation. Provides a list of formations for units.
- Altitude Label. Adds an altitude label independently of the altitude parameter.
- Altitude MSL. Altitude above mean sea level.
- Altitude Rate Change. Lets you specify the rate at which an aircraft changes altitude.
- Animated Movement. Lets you assign an animated movement to an entity.
- Bomb Resource. Creates a list of bomb resources.
- Boolean. Creates a check box.
- Choice (Drop-Down List). Specifies choices presented in a drop-down list.
- Choice (Image Drop-Down List). Lets you specify a set of images to choose instead of text.
- Choice (Multiple Option Buttons). Specifies choices presented as a group of option (radio) buttons.
- Choice (Single Option Button). Specifies a choice presented as a single option (radio) buttons. Single option buttons can be associated through a button group, but placed at different locations on the dialog box.
- Color. Lets you select a color.
- DI-Guy Animation. Lets you select a DI-Guy animation.
- DI-Guy Appearance. Lets you select a DI-Guy appearance.
- DI-Guy Gesture. Lets you select a DI-Guy gesture.
- Date/Time. Specifies the date and time.
- Depth MSL. Specifies the depth below mean sea level.
- Direct Fire Weapon. Lets you select a direct fire weapon.
- Distance. Specifies a distance.
- Embarkation Slot. Lets you select a slot an entity can embark on.
- Embedded Entity. Lets you select an embedded entity to deploy.
- Entity Type. Lets you specify one object type enumeration.
- Entity Types. Lets you specify a list of object type enumerations.
- Filename. Lets you specify a file name.
- Force. Specifies the force.
- Heading. Heading, in degrees.
- Heading Label. Adds a heading label independently of the heading parameter.
- Indirect Fire Weapon. Specifies an indirect fire weapon.
- Integer. A whole number.
- Integer Map. Maps a keyword to an integer.

- Jammer Selector. Lets you select a jammer for an entity. The jammer types list is built automatically based on the entity type.
- Jamming Mode. Lets you select a jamming mode. The jamming mode list is built automatically based on the entity type. If you add a Jamming Mode parameter, VR-Forces automatically adds an Emitter ID parameter to the dialog box.
- Loadout. Specify a loadout for an aircraft unit. (Aggregate-level only.)
- Location (with altitude). Vector description of a location, in geocentric, including altitude.
- Location (without altitude). Vector description of a location, in geocentric, altitude not included.
- Munition Resource. Creates a list of munitions.
- Naval Mine Resource. Specifies a naval mine resource.
- New Tactical Graphic. Lets you define a tactical graphic.
- Observer Mode. Specifies an observer mode, for example, Stealth, Plan View, or XR.
- Observer Saved View. Specifies a saved view.
- Offset Location. Specifies an offset to the side, behind, and above, as in the Follow Entity task.
- Overlay Parent. Lets you select an overlay.
- Radar Mode. Lets you specify a radar mode. The radar mode list is built automatically based on the entity type. If you add a Radar Mode parameter, VR-Forces automatically adds an Emitter ID parameter to the dialog box.
- Radar Selector. Lets you select a type of radar. The radar list is built automatically based on the entity type.
- Range Label. Adds a range label.
- Rate Label. Adds a rate label.
- Real. A floating point decimal number.
- Real Map. Maps a keyword to a real number.
- Resource. Creates a list of available resources.
- Separator. Lets you put a separator in the generated dialog box. This is just for formatting purposes.
- Simulation Object (Single). Lets you select one simulation object in a list window.
- Simulation Objects (Multiple). Lets you select multiple simulation objects in a list window.
- Speed. Any kind of rate, such as speed, climb rate, and so on.
- Sphere. Lets you create a sphere for defining an area on the terrain (Figure 31-3).

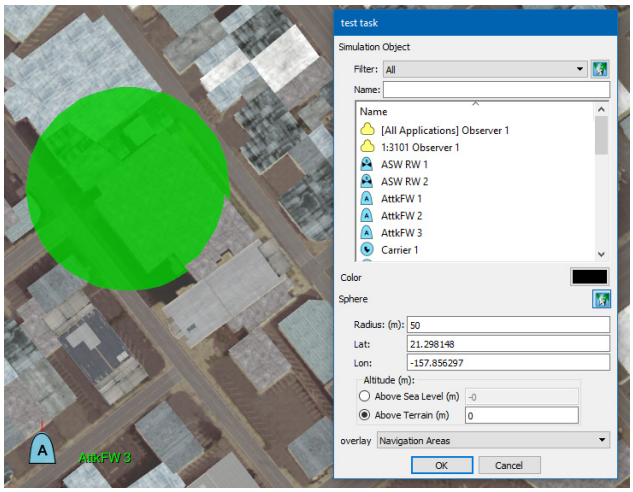


Figure 31-3. A sphere created as part of a scripted task

- String. An alphanumeric string.
- String Map. Maps a keyword to a string.
- Time. Time in days:hours:minutes:seconds.
- Topographic Orientation. Specifies orientation in topographic coordinates.
- Turn Rate. Turn rate in radians/second.
- Turn Rate Label. Adds a turn rate label independently of the turn rate parameter.
- Visual Meta Data. Allows you to select markings for objects that support settings marking, such as a flag or license plate.
- Weapon. Specifies a weapon.

If a script does not have any input parameters, it executes as soon as it is assigned to a simulation object, like the Wait command does.

This section explains how to add parameters using the VR-Forces script creation interface and automatic dialog box creation. You can also add parameters using Qt Designer. For details, please see "Adding Parameters When Using Qt Designer," on page 33-2.

To specify a parameter for a script:

1. Click the Add button () above the Parameters list. The Add Parameter dialog box opens (Figure 31-4).

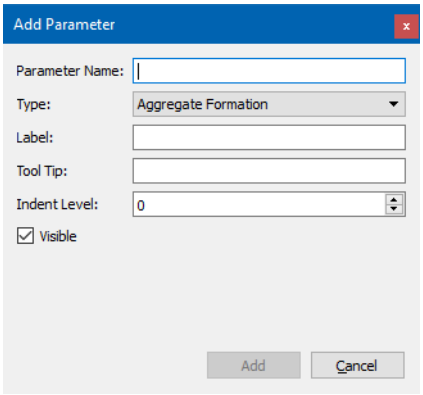


Figure 31-4. Add Parameter dialog box

2. In the Parameter Name box specify a name for the parameter. This is the name used in scripts to get the parameter value.
3. Select the type of parameter from the Type list. If appropriate, the dialog box redisplays to request the details required for the parameter type you selected.
4. Specify the details for the parameter. Table 31-2 explains how to fill out the properties for the various parameter types.

Table 31-2: Scripted task parameter properties

Parameter property	Description
Parameter Name	The name to be used by the Lua script for this parameter. For example, if you give a parameter the name taskDestination, in the Lua code for the script, the parameter will be accessed as `taskParameters.taskDestination`.
Label	This is the text used on the dialog box that is generated for the task. If you leave this field blank, the parameter name is used.
Button Group	For Choice (Single Option Button) parameters, specifies the button group that the option buttons are a part of.

Table 31-2: Scripted task parameter properties

Parameter property	Description
Tool Tip	Text for a tool tip. (Optional)
Indent Level	The level of indentation for this parameter. The indentation level is multiplied by the Indent setting (in pixels) on the Application Settings dialog box, Script Settings page. This allows you to apply some formatting to the dialog box. For details about setting the indentation distance, please see "Setting the Indentation Distance for Script Dialog Boxes," on page 31-22.
Required	For Simulation Object lists only, specifies that this parameter must be set before the OK button is enabled. If the check box is cleared, the requirement is not in effect.
Visible check box	Specifies that the parameter be included on the script dialog box. (You might want a parameter to be available in the code, but not have it set by a user when the script is assigned.)
Default value or state	Many, but not all, parameters have an optional default value.
Range Bottom	The lowest acceptable value for this parameter, if applicable.
Range Top	The highest acceptable value for this parameter, if applicable.
Default value for boolean parameter	The Boolean parameter type lets you create a check box. You can specify that the default be selected or cleared.
Affects Controls	If selected, some fields are populated with the actual simulation object state values. For details about the Affects Controls option for the simulation object parameter, please see "Using the Affects Controls Option," on page 31-21.
Choice	Adds choice option buttons. For details about adding choices to a Choice parameter, please see "Specifying Choices for the Choice Parameters," on page 31-20.
Filter category	Lets you add categories for filtering object lists. For details about adding categories to a filtered list, please see "Specifying Categories for a Simulation Object," on page 31-20.

5. Click Add.

Choosing an Option Button Type

Scripted tasks can use two types of option buttons – multiple or single. The multiple option button choice is a typical option list, as illustrated in (Figure 31-5). In a multiple option list, the choices are mutually exclusive and are automatically associated with each other. You cannot enable or disable other input parameters based on the selection of one of these options.

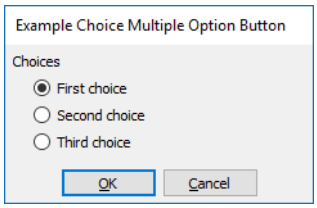


Figure 31-5. Choice (Multiple Option Buttons)

A single button option puts one option button on the dialog box. You can create multiple single button options and place them on the dialog box wherever you want. This allows you to associate other parameter inputs with the option button using the dialog box layout (Figure 31-6). To ensure that these options are mutually exclusive, you associate them by specifying a button group.

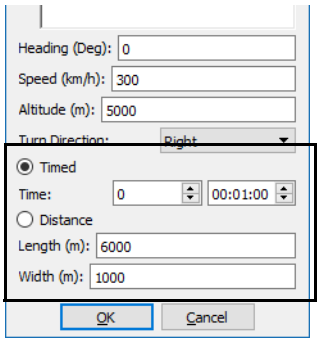


Figure 31-6. Choice (Single Option Button)

You can create multiple sets of options and multiple buttons groups if you need to.

Although you can lay out single option buttons and associated parameters using the default GUI builder for scripts, to associate parameters with the options, you must use Qt Designer. For details about laying out dialog boxes using Qt Designer, please see Chapter 33, *Using Qt Designer for Script Dialog Boxes*.

Specifying Choices for the Choice Parameters

The Choice (Drop-Down List), Choice (Image Drop-Down List), and Choice (Multiple Option Buttons) parameters require a list of choices, which get added to the dialog box as option buttons or as a list. For the image drop down list, you specify images. For the other two, you specify text. You can add choices, edit them, and delete them.

To add a text choice:

1. Click the Add Parameter button ( ) on the Choice Box Values line. The New Choice dialog box opens.
2. Type the text for the choice.
3. Click OK.
4. Repeat this procedure for each choice that you want to have available. (The last choice added becomes the default choice.)

To add an image choice:

1. Click the Add Parameter button ( ) on the Choice Box Images line. The Select Image dialog box opens.
2. Type the path for an image or click the Edit button and select an image in the file browser. The image gets displayed in the dialog box.
3. Click OK. The image is added to the list next to Choice Box Images.
4. Repeat this procedure for each image that you want to have available. (The last choice added becomes the default choice.)

Specifying Categories for a Simulation Object

The Simulation Object parameter lets you build a list of categories on which to filter the display of simulation object types in a dialog box.

To add a category to the simulation object list:

1. Click the Add Parameter button ( ) on the Filters line. The Choose Filter dialog box opens.
2. Select a filter from the Filters list.
3. Click OK.
4. Repeat the procedure to add as many filters as you want.

Editing the Category List for a Simulation Object Parameter

You can edit the list of filter categories for a Simulation Object parameter. The edit operation is essentially a replacement of the current category with a new one. You can achieve the same effect by deleting the current category and adding a new one.

To edit the list of filter categories in a Simulation Object parameter:

1. Select the parameter you want to edit in the Parameters list on the New Scripted Task dialog box.
2. Click the Edit button (). The Script Variable dialog box opens.
3. In the Filters list, select the filter that you want to edit.
4. Click the Edit button (). The Choose Filter dialog box opens.
5. Select the filter category that you want to use.
6. Click OK. The original category is removed and the new one is added.

To delete a category from a Simulation Object parameter:

1. Select the parameter you want to edit in the Parameters list on the New Scripted Task dialog box.
2. Click the Edit button (). The Script Variable dialog box opens.
3. In the Filters list, select the filter that you want to delete.
4. Click the Delete button (). The category is removed from the list.

Using the Affects Controls Option

The Simulation Object (Single) and Simulation Objects (Multiple) parameters have a check box labeled Affects Controls. If you select this check box, when you select a simulation object in the task dialog box, the values for the following parameter types update to the correct values for the selected simulation object:

- DI-Guy Animation
- DI-Guy Appearance
- Embedded Entity
- Emitter
- Force
- Formation
- Location
- Overlay
- Weapon.

If you use the Simulation Objects (Multiple) parameter, it only affects controls if you select just one simulation object.

Setting the Indentation Distance for Script Dialog Boxes

When you add parameters to a script dialog box, you can specify an indent level. The indent level is the level of indentation, not the distance of indentation. The distance that the parameter is indented on the dialog box is the indent level multiplied by the Indent value (in pin the Application Settings dialog box, Script Settings page.

To set the indentation distance for script dialog boxes:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Script Settings page (Figure 31-7).

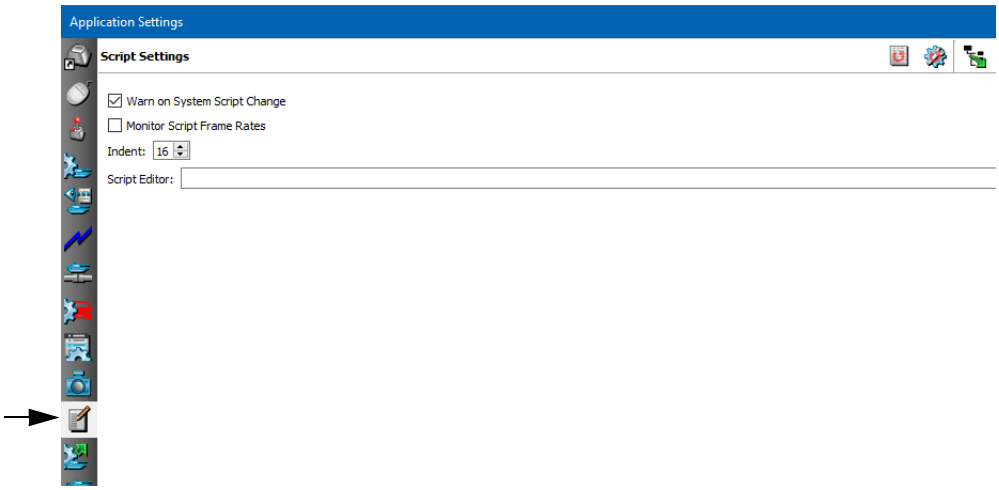


Figure 31-7. Script Settings page

3. Change the value (in pixels) in the Indent box.
4. Click Close.

31.2.7. Specifying a Menu Location

Scripted tasks get added to the Task menu unless you specify that they not be shown (by clearing the Show Task on Task Menu check box). Scripted set data requests get added to the Set menu. The default location is the top level of the menu. However, you will probably want to organize your tasks and sets using the submenus provided by VR-Forces or in newly created submenus. The New Script dialog box displays the menu under which the script is placed. It does not show the full path from the top level menu.

You can also add a scripted task or set to the top level shortcut menu. The script can be listed in both places.



- When you add a script to a menu, it does not appear on the menu until you reload the scenario.
- By default, VR-Forces alphabetizes all Task and Set menu commands. Therefore even if you specify a specific location on a menu, the task or set will be listed in alphabetical order unless you disable alphabetization. For information about alphabetizing Task and Set menu commands, please see "Alphabetizing the Task and Set Menus," on page 84-3.

To specify a menu location for a script:

1. On the Menu Location line, click the Edit button (). For task scripts, the Select Location on Task Menu dialog box opens (Figure 31-1). For set scripts, the Select Location on Set Menu dialog box opens. The two dialog boxes are functionally identical.

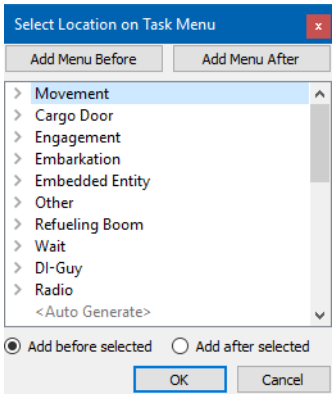


Figure 31-8. Select Location on Task Menu dialog box

2. To add the script to an existing submenu:
   a. Expand the submenu on which you want to put the script.
   b. Select the menu item that you want the new script to be next to.
   c. To put the new script above the selected menu item, select the Add Before Selected option. To put the new script below the selected menu item, select the Add After Selected option.
   d. Click OK.
3. To create a new submenu for the script:
   a. Select the submenu next to where you want to add the new submenu.
   b. To add the new menu before the selected submenu, click Add Menu Before. To add the new menu after the selected submenu, click Add Menu After. The New Menu Name dialog box opens.
   c. Type a name for the new submenu.
   d. Click OK.
   e. Add the script to the new submenu using the procedure in step 2.

31.2.8. Specifying Action Categories

Some VR-Forces tasks can run at the same time as other tasks, while some are mutually exclusive. This is called task concurrency. (For details about concurrent tasks, please see "Concurrent Task Execution," on page 25-4.) The C++ tasks provided with VR-Forces are organized in groups to help manage concurrency. They are configured in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/default-task-rules.tsk*.

In the New Script dialog box, these groups are represented as action categories. The action categories displayed in the New Script dialog box are configured in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/actionCategories.tsk*. (For details about configuring action categories, please see "Configuring Action Categories," on page 25-34.)

When you create or edit a scripted task you can specify which groups of tasks the new task can run concurrently with and which it will interrupt. By default, the Movement category is selected, which means that if you do not change the setting, the new task will interrupt movement tasks.

➤ To specify which tasks a scripted task will interrupt, in the Action Categories list, select the categories to interrupt. For example, if the new task involves movement, you probably want it to interrupt other movement tasks, so you would select the Movement category check box. If it is a new type of message task, you probably want to let it run concurrently with movement tasks, so you would clear the Movement category check box.

31.2.9. Configuring a Script's Availability to a Simulation Object Type

Scripts can be available to a simulation object based on the following criteria:

- The script is available for certain simulation object types. For details, please see "Specifying the Valid Simulation Object Types for a Script," on page 31-26.
- The script is part of a behavior set. For details, please see "Specifying Availability by Behavior Set," on page 31-27.
- A task or set is specified in a system definition. (For example, the Naval Mine Sweep task is available to entities with a Naval Mine Sweep system.) This state is not shown in the Edit Script dialog box. For details, please see "Specifying a Scripted Task or Set in a System Definition," on page 31-27.

The default setting for new scripts is that they are available to all simulation objects.

Availability by simulation object type and by behavior set is configured in the New Script and Edit Script dialog boxes. Table 31-3 shows the interaction of these settings.

Table 31-3: Scripted task availability

Available by Entity Type	Available by Behavior Set	Availability
☑	☐	The script is available only to those simulation object types listed.
☐	☑	The script is available to all simulation objects, but only if one of the selected behavior sets is applied to the simulation object's force.
☑	☑	The script is available to the specified simulation object types, but only if one of the selected behavior sets is applied to the simulation object's force.

Specifying the Valid Simulation Object Types for a Script

You can restrict a script to specific simulation object types. If you do this, the script will not appear on the Task or Set menu when you select a simulation object for which it is not valid. Just as with built-in VR-Forces tasks and sets, there are cases, such as for global plans, where VR-Forces cannot determine the simulation object context and the script will be available on the menu even though it is not appropriate to the simulation object that you selected. In those cases it is up to you to know that a task or set is valid.

To specify the valid entity types for a script:

1. Select the Make Available Based on Entity Type check box.
2. Click the Add button ( ) on the Valid for Entity Types line. The Entity Type dialog box opens (Figure 31-9). You can build an enumeration by selecting options from the enumeration field lists or you can type in the enumeration directly.

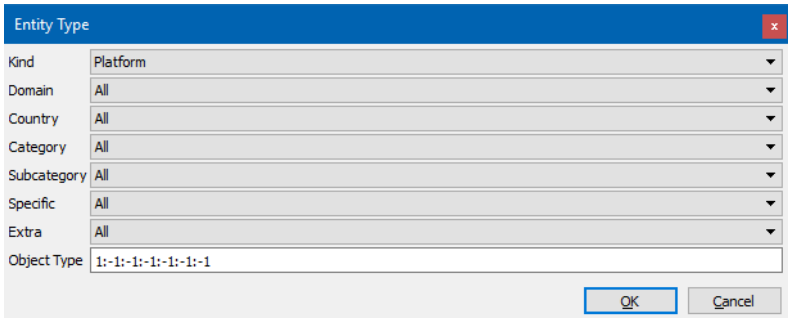


Figure 31-9. Entity Type dialog box

3. For each field of the simulation object type (Kind, Domain, Country, and so on), select an option from the list.

   The options are drawn from the SISO Enumerations document. As you select an option, the enumeration at the bottom of the dialog box changes to reflect your choice. You do not have to make a selection for each field. VR-Forces uses wild carding to match simulation objects against the final enumeration.
4. If appropriate, add additional enumerations for each simulation object type that is valid for the script.

Specifying Availability by Behavior Set

You can restrict a script to specific behavior sets. If you do this, the task will not appear on a menu when you select a simulation object whose force does not have the behavior set assigned to it.

To specify that a script is available by behavior set:

1. Select the Make Available Based on Behavior Set check box.
2. Select each behavior set for which you want this script to be available.



You must create the behavior sets before you can assign scripts to them. All available behavior sets are automatically listed in the dialog box window.

Specifying a Scripted Task or Set in a System Definition

Some scripted tasks and sets are available to simulation objects because they are specified in a system that the simulation object uses. They are not assigned to simulation object types as part of the script metadata.



This method is used for many of the system scripted tasks supplied with VR-Forces.

Scripted tasks and sets are specified in a system definition by their script IDs in the script-ids parameter. (In *AggregateLevel.sms*, some scripted tasks for combat engineering objects are specified in the create-script-id, improve-script-id, and destroy-script-id parameters.) Script IDs can be specified in a script-enable-controller and in controllers derived from it.

The following example is from *ground-aggregate-movement.sysdef* in *AggregateLevel.sms*. Any simulation object with this system definition can use the Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, and Move_To_Waypoint_Plan_Path scripted tasks. The variables described in the script-variables block are accessible to the script through the vrf:getScriptAttribute() function.

```
(script-enable-controller
  (component-descriptor-type "script-enable-controller-descriptor")
  ...
  (script-ids "Maximum_Speed_Modifier" "Halt_Movement"
    "Move_To_Location_Plan_Path" "Move_To_Waypoint_Plan_Path")
  (script-variables
    (maximum-speed-modifier-variables
      (script-id "Maximum_Speed_Modifier")
      (variables
        (DtRwReal Rain-Modifier-By-Intensity
          $rain-modifier-by-intensity)
        (DtRwReal Snow-Modifier-By-Intensity
          $snow-modifier-by-intensity)
        (DtRwReal Visibility-Degrades-When-Below
          $visibility-degrades-when-below)
```

```
          (DtRwReal Visibility-Can-Degrade-Speed-Up-To
            $visibility-can-degrade-speed-up-to)
        )
      )
      (move-to-location-variables
        (script-id "Move_To_Location_Plan_Path")
        (variables
          (DtRwString Obstacle-Feature-Query "MAK_OBSTACLE")
          (DtRwString Path-Feature-Query "MAK_ROAD")
        )
      )
    ...
    )
)
```

The following example is from the *aggregate-5-inch-gun.sysdef* in *AggregateLevel.sms*. It adds the script-ids parameter to the `aggregate-indirect-fire-controller`, which is derived from the `script-enable-controller`. Any simulation object with this system can use the Indirect Fire and Limited Munition Attack scripted tasks.

```
(weapon-indirect-fire
  (systems )
  (sensors )
  (controllers
    (aggregate-indirect-fire-controller
      (component-descriptor-type "aggregate-indirect-fire-controller-
descriptor")
      (component-type  "aggregate-indirect-fire-controller")
      (min-tick-period  -1.000000)
      (min-tick-period-variance  -1.000000)
      (process-state-repository-name "aggregate-indirect-fire-
controller-process-state-repository-default")
      (process-state-repository-type  "aggregate-indirect-fire-
controller-process-state-repository")
      (is-enabled True)
      (script-ids "Indirect_Fire" "Limited_Munition_Attack")
      (weapon-name  "5 inch shell")
      ...
    )
  )
```

The following example is from *obstacle-builder.sysdef* and shows the create-script-id and improve-script-id parameters:

```
(creatable-objects
    (affectable-object
      (object-type  1 (16 1 0 62 3 0 0))
      (time-to-complete $time-to-create-abatis)
      (range $creation-range-abatis)
      (create-script-id "Construct_Abatis")
      (improve-script-id "Improve_Obstacle")
    )
    ...
)
```

31.2.10. Specifying Equivalent Scripts for a Toolbar Button

There may be occasions when you want to have tasks or sets with the same name that work differently depending on the simulation object you give them to. For example you might have a movement task that works one way for an individual entity and another way for a unit. You can ensure that when you select a simulation object it gets the correct version of the script by specifying the simulation object type that the script is available to. To simplify the Task or Set toolbars, you can map these different scripts to the same toolbar button. This is done by specifying that one script is equivalent to another script.

If you map multiple scripts to the same toolbar button, when you select a simulation object, the toolbar button is enabled and you can give the simulation object the task or set. VR-Forces chooses the correct version of the script. If you select multiple simulation objects that would have different versions of the script, the toolbar button becomes disabled.

To specify an equivalent script for a Task or Set Toolbar button:

1. Create a new script.
2. On the New Script dialog box (or Edit Script dialog box), select Show Task on Task Toolbar (or Show Set on Set Toolbar).
3. Click the Toolbar Location edit button (). The Select Location on Task (Set) Toolbar dialog box opens.
4. Select the task or set that you want this script to be equivalent to. The Equivalent To button becomes active.
5. Click the Equivalent To button. Text is added to the script's entry in the list indicating the script it is equivalent to (Figure 31-10).

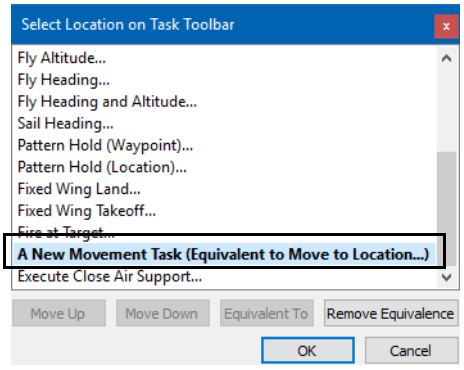


Figure 31-10. Select Location on Task Toolbar dialog box



Do not move the new script to a different location on the toolbar. Moving it breaks the equivalency.

6. Click OK.

7. Save the script.

31.2.11. Specifying the Programming Language for a Script

The Language list in the New Script dialog box has two options, Lua and C++. In most cases you will choose Lua and write a script to implement the task or set. However, since the script interface automatically generates dialog boxes and updates menus, it can be a convenient way to add the GUI component for new tasks or sets that developers have written in C++ with the VR-Forces Toolkit. Developers who take this approach do not have to rebuild the VR-Forces front-end or write a plug-in to add their new task or set to the menu and code a dialog box.



The Language option only applies to Task and Set type scripted tasks. Reactive tasks and background processes are always written in Lua.

If a developer adds a new C++ task controller using this approach, it looks for the script ID of the scripted task. For example, the Deploy and Recover tasks for embedded entities use the scripted task interface to create their dialog boxes. The addTask example in the VR-Forces Toolkit demonstrates this method of adding tasks. For more information, please see *VR-Forces Developers Guide*.

You can display C++ tasks and sets in the Scripts dialog box by selecting the Show C++ Tasks check box (Figure 31-11).

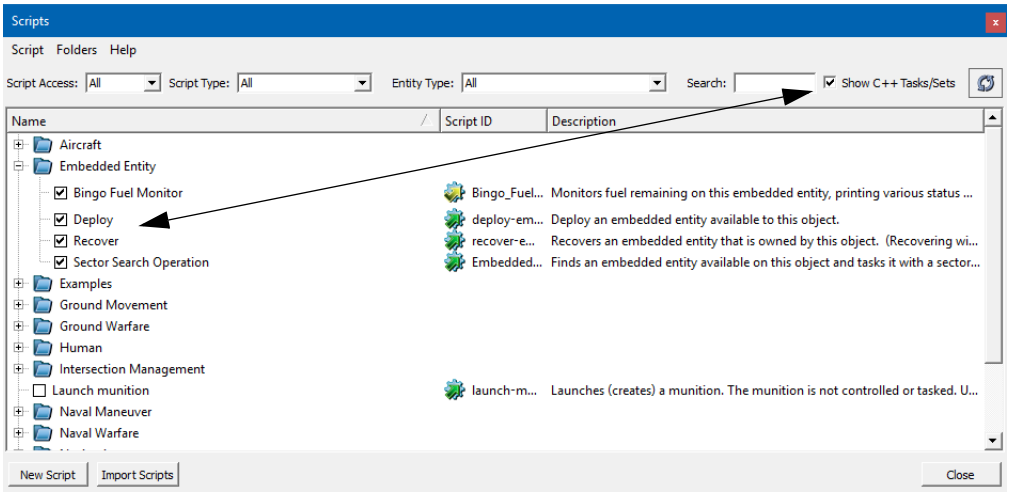


Figure 31-11. C++ tasks

31.2.12. Configuring Scripted Tasks to Run Concurrently

You can specify that an entity be able to run more than one instance of a task at the same time. The concurrent task feature is intended for tasks that require an entity to manage two non-conflicting activities, such as when an air base launches a flight in an aggregate-level scenario. If an air base receives two launch flight tasks in close succession and concurrent tasks are allowed, both tasks would complete. If this option is not in effect, the second task would cause the air base to stop executing the first instance of the task.



This setting overrides the action category settings. It is your responsibility to make sure that you do not allow concurrent task execution for tasks that would not make sense to run concurrently.

➤ To specify that a task can be run concurrently, select the Can Be Executed More Than Once at a Time check box on the Edit Scripts dialog box.

31.2.13. Creating Scripted Sets

The metadata for a scripted set is identical to that of a scripted task except that a scripted set does not have action categories.

To create a scripted set:

1. Choose **Simulation** → **Scripts**. The Scripts dialog box opens (Figure 31-1).
2. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, please see "Organizing Scripts into Folders," on page 31-38.)
3. Choose **Script** → **New Script** or click the New Script button. The New Script dialog box opens (Figure 31-2).
4. In the Script Type list, select Set. The New Script dialog box redisplays to show the metadata fields for a scripted set (Figure 31-13).

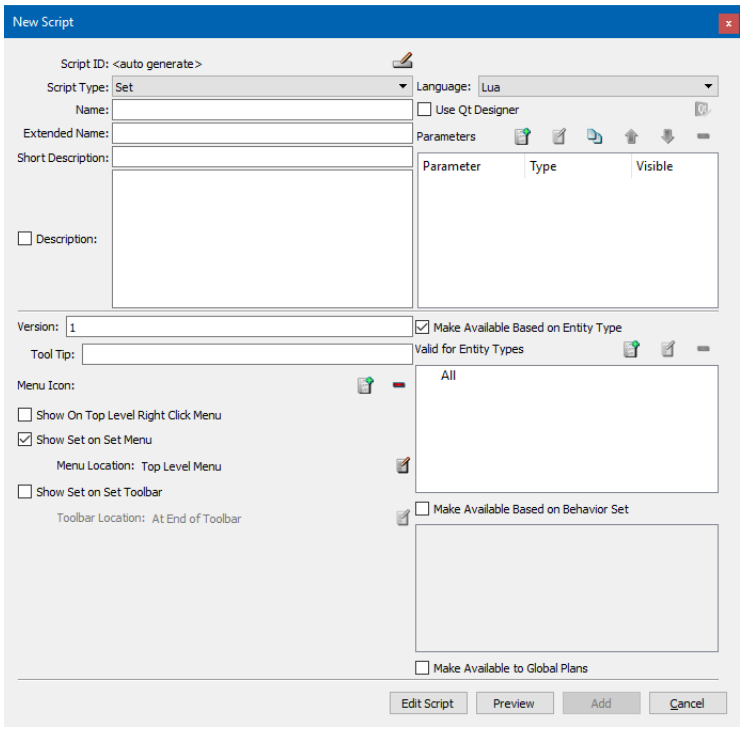


Figure 31-12. New Script dialog box, scripted set

5. Complete the fields in the dialog box. Fields that are common to all scripts are described in Table 31-1.

31.2.14. Creating Reactive Tasks

The process for creating a reactive task is largely the same as for creating other scripts. This procedure focuses on the differences.

To create a reactive task:

1. Choose **Simulation** → **Scripts**. The Scripts dialog box opens (Figure 31-1).
2. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, please see "Organizing Scripts into Folders," on page 31-38.)
3. Choose **Script** → **New Script** or click the New Script button. The New Script dialog box opens (Figure 31-2).
4. In the Script Type list, select Reactive Task. The New Script dialog box redisplays to show the metadata fields for a reactive task (Figure 31-13).

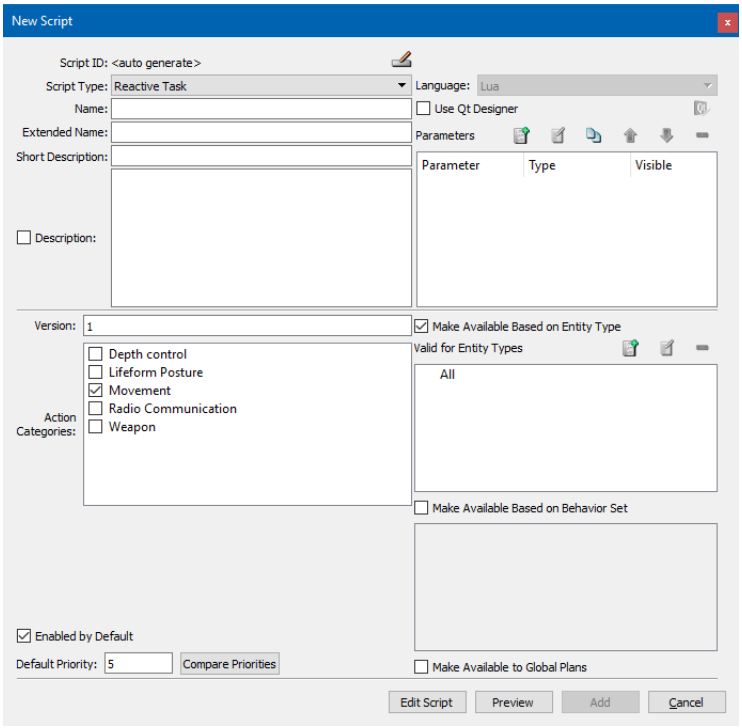


Figure 31-13. New Script dialog box, reactive task

5. Complete the fields in the dialog box. Fields that are common to all scripts are described in Table 31-1.
6. To make this reactive task enabled by default, select the Enabled By Default check box.
7. In the Default Priority box, type the priority for this task. The lower the number, the higher the priority. To help you set the priority, you can view a list of all reactive tasks that are valid for the simulation object types of this task so that you can consider their relative importance and set an appropriate priority.
8. Optionally, click Compare Priorities to see a list of reactive tasks and their priorities.
9. Click Add.

31.2.15. Creating Background Processes

The procedure for creating a background process is essentially the same as for other script types. The main difference is that there are fewer fields to complete in the New Script dialog box. For details about writing a background process script, please see "Background Processes," on page 32-32.

To create a background process:

1. Choose **Simulation** → **Scripts**. The Scripts dialog box opens (Figure 31-1).
2. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, please see "Organizing Scripts into Folders," on page 31-38.)
3. Choose **Script** → **New Script** or click the New Script button. The New Script dialog box opens (Figure 31-2).
4. In the Script Type list, select Background Process. The New Script dialog box redisplays to show the metadata fields for a background process (Figure 31-13).
5. Complete the fields in the dialog box. Fields that are common to all scripts are described in Table 31-1.

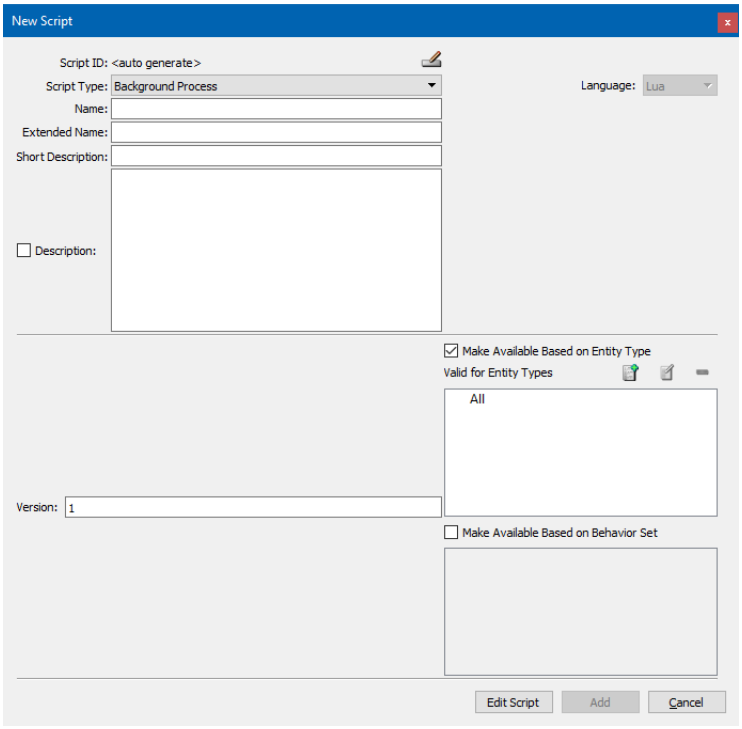


Figure 31-14. New Script dialog box, background process

6. Click Add.

31.3. Saving Scripts

When you create or edit a script, its listing in the Scripts window is in italics, prefaced by an asterisk. When you save the scenario, or promote a script to a system script, the script gets saved and the italics and asterisk get removed.

It is likely that you will test scripts by running a scenario and you will want to edit the script as you find problems. If you create or edit a script while a scenario is running and you decide to rewind the scenario without saving it (which would be normal behavior for rewinding a scenario), you are prompted to preserve the changes to the script after the scenario rewinds. The script is still not saved and you will, therefore, receive the standard "scenario modified" prompt when you start the scenario again. You do not have to save the scenario if the only thing that changed is the script. You can continue to run the scenario and rewind while you are working on the script. Eventually you will have to save the scenario if you want to save your changes to the script.

Metadata for system scripts gets saved to a file in the format *./data/simulationModelSets/<model_set>/scripts/script_name.xml*. The script name is always saved as lower case.

➤ To save a scenario script, save the scenario.

➤ To save a system script, after updating the script in the Edit Script dialog box, click Update.

31.4. Editing a Script

You can edit scripts.

! If you edit a system script, the changes affect all scenarios that use the script. You are responsible for ensuring that any changes have the intended results for all scenarios that are affected.

To edit the metadata for a scenario script:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Select the script that you want to edit.

---

i In the Script ID column the icon indicates the type of script, Task (), Reactive Task (), Set (), or Background Process ().

---

3. Choose **Script** → **Edit Script Meta Data**, or double-click the script name. The Edit Script dialog box opens. (If you are editing a system script, you are prompted to confirm that you want to edit it.)
4. Edit the metadata as desired.
5. Click Update.

31.5. Filtering the List of Scripts

You can filter the tasks listed in the Scripts dialog box by selecting options from the lists below the menu bar, as follows:

- Script Access: Show system scripts, scenario scripts, or all scripts.
- Script Type: Show scripted tasks, reactive tasks, sets, background processes, or all scripts.
- Entity Type: Show scripts that apply to the selected simulation object type.

---

i This filter is based on the Valid for Entity Types list for each script. Some scripts do not have any simulation object types listed. These scripts, such as Drop Naval Depth Charge, become available to simulation objects based on the systems that they have configured. Therefore, this list may be somewhat misleading in terms of the overall applicability of scripts to the different platforms.

- Show C++ Tasks: Toggles the display of tasks and sets that are written in C++, but whose GUIs are created using the script mechanism.

31.6. Organizing Scripts into Folders

You can create folders and organize scripts by folder.

31.6.1. Adding a Folder

To add a folder:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Choose **Folders** → **Add Folder**. The Folder Name dialog box opens.
3. Type a name for the folder.
4. Click OK.

31.6.2. Renaming a Folder

To rename a folder:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Select the folder that you want to rename.
3. Choose **Folders** → **Rename Folder**. The Folder Name dialog box opens.
4. Type a new name for the folder.
5. Click OK.

31.6.3. Deleting a Folder

You cannot delete a folder that has scripts in it.

To delete a folder:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Select the folder that you want to delete.
3. Choose **Folders** → **Delete Folder**.

31.6.4. Adding Scripts to a Folder

➤ To add a script to a folder, drag it onto the folder.

31.6.5. Removing a Script from a Folder

➤ To remove a script from a folder, drag it away from the folder.

31.7. Exporting and Importing Scripts

Scenario scripts are saved in a scenario's SPT file, not as individual scripts. If you want to make a scenario script available to other scenarios, but you do not want to save it as a system script, you can export the script from the scenario in which you created it and import it into any other scenario. You can export multiple scripts to one file as a script "package".



You cannot add or remove scripts from a script package. To change the contents you would have to create a new package.

If scripts are organized in folders, when you save them the folder structure is also saved.

To export scripts:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Select the scripts that you want to export. (The window supports typical methods for selecting multiple entries in a list.)
3. Choose **Script** → **Export Script Package**. The Choose Script File dialog box opens. By default it opens in *./userData/taskScripts*.
4. Type a name for the scripts file.
5. Click Save. The scripts gets saved to a file with the extension *.spt*. The file contains the metadata and the script code.

31.7.1. Importing a Script Package

If you import a script package that has multiple scripts, you can choose which scripts to import. Scripts get imported with the folder hierarchy that they were saved in.

To import scenario scripts:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Choose **Script** → **Import Script Package** or click Import Scripts. The Choose Script File dialog box opens.
3. Select the script file that you want to import.
4. Click Open. The Import dialog box opens (Figure 31-15). It lists the scripts saved in the script file.

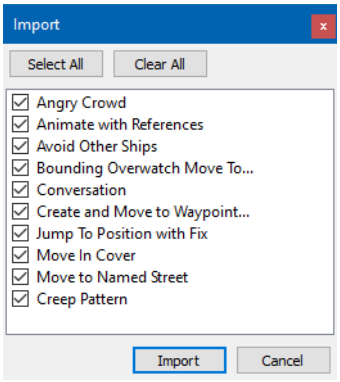


Figure 31-15. Import dialog box

5. Select the scripts that you want to import.
6. Click Import.

31.8. Copying a Script

If you want to create a new script that is similar to an existing one, you can copy an existing script and then edit it for its new purpose.

To copy a script:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Select the script that you want to copy.
3. Choose **Script** → **Duplicate**. The New Script dialog box opens. If you copied a scenario script, the duplicate script is assigned a new script ID. If you copied a system script, the script ID is not changed. You will have to change the script ID manually.



If you do not change the script ID for a copied system script, the system script gets removed from the list of scripts for this scenario and the new script is added as a scenario script. This lets you experiment with changes to a system script without actually changing it. The original system script will still be available to other scenarios.

4. Edit the script.
5. Click Duplicate. The script is added to the list.

31.9. Deleting a Script

If you delete a script that is assigned to a simulation object in a scenario, a task assignment, whether as an independent task or as part of a plan, will fail. If a scripted task is used as a subtask in another scripted task, that scripted task will fail. Scripted sets also fail. VR-Forces does not warn you if either of these cases exist. It is your responsibility to consider the effects of deleting a script.

To delete a script:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Select the script that you want to delete.
3. Choose **Script** → **Delete Script**.

31.10. Creating a System Script

To create a system script, you create a scenario script and then save it as a system script. System scripts are saved in *./data/simulationModel-Sets/<model_set>/scripts*. A system script has two files. The metadata is saved in a file ending with the extension *.xml*. The code is saved in a file with the extension *.lua*.

To save a scenario script as a system script:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Select the script that you want to save as a system script.
3. Choose **Script** → **Promote to System Script**. You are prompted to confirm the change.
4. Click Yes.

31.10.1. Including System Scripts on the Task and Set Menus

Most of the system scripted tasks and sets included with VR-Forces are listed on the Task and Set menus. The scripts in the Examples folder and a few others are not listed. You can specify whether or not particular scripted tasks or sets are listed. You might want to do this to reduce clutter on the menus. Or you might want to limit the tasks available to players in a particular deployment of VR-Forces.

A system script's inclusion on a menu is controlled by the following settings:

- The Show Task on Task Menu (or Show Set on Set Menu) check box on the New Script and Edit Script dialog boxes.
- The check box next to the script's entry on the Scripts dialog box (Figure 31-16).
- Its membership in a behavior set. The effect of behavior sets on script availability is additive to the configuration discussed in this section. For details about using behavior sets, please see "Using Behavior Sets to Manage Scripts," on page 25-18.

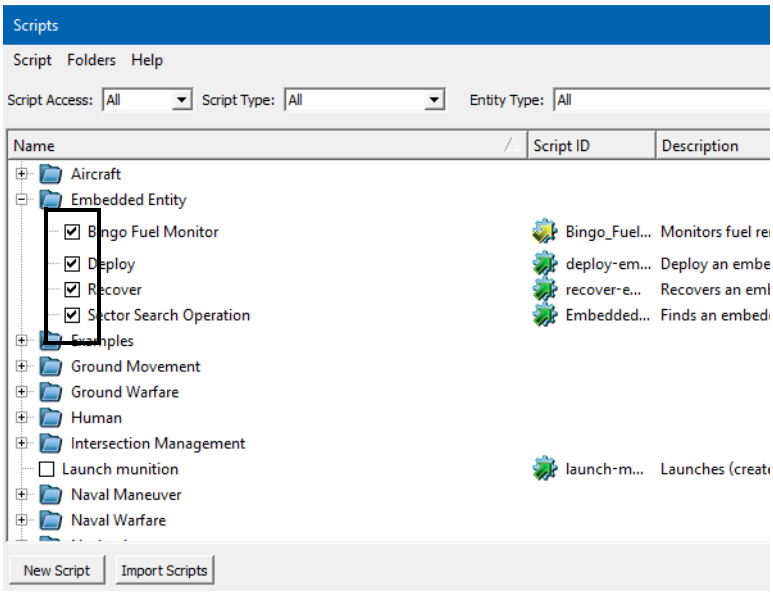


Figure 31-16. Show script on menu check box

Table 31-4 shows when a scripted task is listed on a menu based on the configuration check boxes selected.

Table 31-4: System scripted task inclusion on Task menu

Show Script on Menu check box	System script list check box	Task included on menu
☑	☑	Yes
☑	☐	No
☐	☑	Yes, for this scenario.
☐	☐	No

To specify the availability of a system script on a menu:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. In the list of system scripts, select the script whose availability you want to change. Use Table 31-4 to determine which check box configuration you need for your desired menu status.
3. Choose **Script** → **Edit Script Meta Data**. The System Script Editing prompt opens. (If you have disabled this prompt, the Edit Script dialog box opens. Skip to step 5.)
4. In the System Script Editing prompt, click Yes. The Edit Script dialog box opens.
5. Select or clear the Show Task on Task Menu check box (or the equivalent for sets).

6. Click Update.
7. In the list of scripts, select the check box next to the system script you are editing to make it available. Clear the check box to make it unavailable.

31.11. Specifying a Script Editor

VR-Forces installs SciTE as the default script editor for scripted tasks. However, you can use any text editor that you want.

To specify the editor to use for scripts:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Script Settings page (Figure 31-7).
3. Click the Browse button (...) next to the Script Editor box. The Choose Script Editor dialog box opens.
4. Select the application that you want to use as the editor.
5. Click Open.

31.12. Editing Lua Files

To edit the code for scenario scripts you need to work through the Script GUI because the scripts are embedded in the scenario scripts file rather than being saved as individual Lua files. System scripts are saved as Lua files, but even in this case it is better to edit them in the context of the entire script process rather than by editing the Lua file. However, VR-Forces includes some libraries of Lua functions that are independent of any of the scripts. They are available to be included in your scripts. If you want to edit the functions in these libraries or add to them, you need to edit the individual Lua files. The function libraries are in *./userData/scripts*. You can open them in any editor, or access them through the Scripts dialog box.

To edit a Lua file from the Scripts window:

1. Choose **Simulation** → **Scripts**. The Scripts window opens (Figure 31-1).
2. Choose **Script** → **Edit External Script**. A file chooser dialog box opens.
3. Select the script that you want to edit. The file opens in a text editor.
4. Edit the file.
5. Save the file.



32. Writing Scripts

This chapter explains how to write Lua scripts for scripted tasks and sets.

The VR-Forces Lua Interface 32-2
- Lua Classes 32-2
- Lua Modules 32-5

Script Loading and Execution 32-5
- Script Entry Points 32-6
- The Script Execution Sequence 32-8
- Limitations for Checkpointing Scripts 32-11
- Editing Scripts While a Scenario is Running 32-13

Tasks and Subtasks 32-14
- Monitoring the Status of Tasks and Subtasks 32-15
- Stopping Tasks 32-15
- Task Parameters 32-16

Geometry 32-19
- Location3D 32-19
- Vector3D 32-19
- VectorOffset3D 32-22
- VectorGeoc3D 32-22

Coding Messages for Translation 32-23

Error Detection 32-23

Interactive Scripts 32-25

A Basic Scripted Task 32-25
- Create and Move to Waypoint Metadata 32-26
- The Create and Move to Waypoint Lua Script 32-28

A Simple Scripted Set 32-30

A Simple Reactive Task 32-31

Background Processes 32-32

32.1. The VR-Forces Lua Interface

The VR-Forces Lua interface provides predefined elements that you can use to create scripts. These elements include object classes that represent the simulation, simulation objects, and geometric objects, plus Lua modules that provide tools for common tasks.

The classes and functions in the Lua interface are documented in the VR-Forces Lua API Documentation. This is a set of HTML files similar to the VR-Forces Toolkit documentation. It is accessible from the Windows Start menu or in *./doc/luadoc/index.html*.

32.1.1. Lua Classes

The Lua interface defines classes for common objects and functions. The classes in the interface are not fully featured classes as in the C++ language, but they do abstract the representation of information associated with an object and collect a set of methods related to the object. In Lua, to call a method `fn` on object `obj`, write:

```
obj:fn()
```

Several of the classes defined in the interface do not have constructors, because their objects are pre-defined or created only as return values to other functions. The classes and pre-defined objects in the interface are as follows:

- *ComponentSystem*. A component system is a capability that is assigned to a simulation object in the Simulation Object Editor. Systems represent movement and damage capabilities, and represent individual weapons and sensors. In scripts, system objects are obtained from the ownship simulation object ("this") with a call to getSystem(), passing the name of the system (as used in the Simulation Object Editor) as an argument. The script can then access various attributes of the system through getAttribute calls.
- *FeatureSet*. This class represents a set of features obtained from a query to the terrain database. A *FeatureSet* is created with a *vrf* query function. *FeatureSet* methods allow a script to examine the features to determine their attribute values, vertex locations, and so on.
- *Location3D*, *Vector3D*, *VectorGeoc3D*, *VectorOffset3D*. These classes represent geometric objects. For more information, please see "Geometry," on page 32-19.
- *ScriptAnimationModel*. *ScriptAnimationModel* refers to an animation model that can be used by a scripted task to animate an entity's movement. This class is serialized to the simulation and might contain movement data if supplied by the developer.
- *SensorGimbal*. *SensorGimbal* is a class of Lua object that represents a sensor that rotates on a gimbal.

- *SimObject*. *SimObject* refers to a single object in the simulation. It may be an entity, waypoint, route, or any other simulation object. These objects may be VR-Forces simulated objects, or remote objects received over DIS or HLA by VR-Forces. There is no constructor for SimObjects in the interface, but there are *vrf* functions that create objects, that get objects by name, that get nearby objects, and so on. *SimObject* methods are provided for getting information about the object such as location, force type, and embarkation status.
- *SwingingArtPart*. *SwingingArtPart* is a class of Lua object that represents a rotating articulated part.
- *TranslatableArtPart*. *TranslatableArtPart* is a class of Lua object that represents a translating articulated part.
- *TranslatableStream*. *TranslatableStream* is a text stream class that can be translated by VR-Forces for display in the GUI as text that is translated to the current language the user is displaying (as long as those translations exist.)
- *this*. *this* is an object of class *SimObject* that is defined for all Lua scripts. It represents the simulation object running the script, also known as "ownship". It is not possible to construct a *this* object in a script. In addition to being able to use all of the *SimObject* methods, a Lua script can change the location and heading of *this*, manipulate its resources, and examine the contacts detected by the simulation object's sensors.
- *vrf*. *vrf* is defined as a global object for all Lua scripts. It is an instance of a class that represents the Lua interface; other instances cannot be created in a script. The *vrf* object provides functions that control and access information in the VR-Forces simulation.



In Lua, variables assigned to objects actually just hold a reference to an object, so that if one variable is assigned to another they will both hold a reference to the same object. In the following example, `firstObj` and `secondObj` refer to the same object.

```
firstObj = getObjectByName("APC 1")
secondObj = firstObj
```

In most cases this is what a Lua programmer wants. However, for the geometric objects, a Lua programmer may sometimes want to make a copy of an object. A simple assignment does not make a copy in Lua, so the geometric objects have makeCopy() methods to provide copies.

Reserved Words

Table 32-1 lists reserved words (in addition to those reserved by Lua), which should not be used except for their intended purpose.

Table 32-1: Reserved words

Name	Module	Function
vrf	vrf	Module singleton
this	SimObject	Ownship name
ScriptVrfObject	SimObject	Class name
Location3D	Location3D	Class name
Vector3D	Vector3D	Class name
VectorGeoc3D	VectorGeoc3D	Class name
VectorOffset3D	VectorOffset3D	Class name
ScriptFeatureSet	FeatureSet	Class name
ScriptAnimationModel	ScriptAnimationModel	Global function
vrfprint	vrfutil	Global function
printDebug	vrfutil	Global function
printVerbose	vrfutil	Global function
printInfo	vrfutil	Global function
printWarn	vrfutil	Global function
printError	vrfutil	Global function
EntityType	vrfutil	Global table of functions
bhaveLoaded	vrfutil	Global function
new	fsm	Function name

32.1.2. Lua Modules

The Lua interface includes Lua modules for use in Lua scripts. A module `stuff` can be accessed in a Lua script by writing

```
require "stuff"
```

The modules available in the interface are as follows:

- *vrfutil*. Technically, *vrfutil* is not a Lua module. (It is not defined with a module function.) It is just a set of global utility functions. The Lua script template that opens when a new script is created contains:

  ```
  require "vrfutil"
  ```

  so that these functions are available to the script by default.
- *fsm*. The *fsm* module defines a class that implements finite state machine logic. There is one function in this module, new(), which creates an *fsm* object.
- *exectool*. The *exectool* module is a package of functions that supports the use of a single Lua function to describe the control flow of a scripted task. To use this tool, the task logic is described in a function we call a scriptFunction. This function is written as if it will execute up to a subtask invocation, and then wait until that subtask is completed before continuing.

The *.lua* module files are located in *./userData/scripts*. You can add user-created modules to this directory and then import them into scripts with a `require` statement.

32.2. Script Loading and Execution

This section describes script entry point functions and the execution sequence for scripted tasks and sets. It also describes how VR-Forces checkpoints scripted tasks and executes them when a checkpointed scenario is run.

32.2.1. Script Entry Points

The Lua API includes functions that we call "entry points" to a scripted task. When you create a new script, they are included in the default script template. Unlike the Lua functions that you might write as part of a script, you do not have to call entry point functions from within the script. They are called by the simulation engine. (However, if an entry point function is not defined in a script, it does not get called.)

- init(). Use init() to set up the initial state of your scripted task's environment. The init() function is the first function called when a scripted task is executed. It is called when a task is assigned to a simulation object, even if the scenario is paused. It is never called again for the life of the scripted task.
- tick(). The tick() function is the heart of every scripted task. It is called at every tick of the simulation engine once the simulation object has completed its initialization. Most of a scripted task's logic should be written into the tick() function as it is the only entry point function consistently called in the script.
- suspend(). The suspend() function gets executed when a reactive task is triggered for a simulation object and the reactive task cannot run concurrently with the current task. The typical behavior, as embodied in the scripted task template, is to stop all tasks and subtasks.
- resume(). The resume() function gets executed when a reactive task completes. The typical behavior is to reinitialize the suspended task.

As you can see from their descriptions, the suspend() and resume() functions do not exactly suspend and resume tasks as those terms are commonly understood. In most cases (this is task dependent), VR-Forces does not preserve the state of a simulation object's tasks when they are suspended, it stops the tasks. Nor does it resume the previous task from the point at which it was suspended, it restarts the task from the current state of the simulation object. However, VR-Forces always saves the lua state of scripted tasks, so when a scripted task resumes, its variables may need to be reset in init() or checkInit().

- saveState(). If a script is running, the saveState() function is called just before a scenario is about to be saved. VR-Forces automatically saves the state of script execution. (For details, please see "Limitations for Checkpointing Scripts," on page 32-11.) This function lets you do any additional processing to save the state of your scripted task. Most scripts do not use this function.
- loadState(). If a script was running when a scenario was saved, the loadState() function is called just after a scenario is loaded or rewound. VR-Forces automatically restores the state of a scripted task that was running when a scenario was saved. This function lets you do any additional processing required to restore the state of the script. If your script uses the saveState() function, you probably want to include complementary processing in loadState(). Most scripts do not use this function.



If your loadState() function assumes the presence of other simulation objects, you must take into account that other simulation objects might not have loaded yet when the function is called.

- shutdown(). The shutdown() function is called when the task is ending for any reason. You will not typically need to write any code in this function, however it is available if there is a case in which you want to do anything before the task ends.
- receiveTextMessage(). The receiveTextMessage() function is called when the simulation object executing this task receives a text message. The function provides some data to determine what the text message is and which simulation object the text message came from. This function takes two arguments, the message and the sender. The message is a text string and the sender is the simObject that sent the message.
- entityDestroyed(). The entityDestroyed() function is called once when the entity executing the script is destroyed.
- receiveInfluence(influencer, influenceName, influenceParams). The receiveInfluence() function is called in aggregate-level scenarios when a simulation object receives an Influence interaction from another simulation object. Influence interactions represent different ways an object might affect another. These include engagement, jamming, providing supplies, improving engineering objects, or destroying engineering objects.

  The receiveInfluence() function has the following parameters:

  – influencer. The simulation object that is performing this influence action.
  – influenceName. The name of the influence being performed. It matches one of the subclasses of the Influence interaction FOM class.
  – influenceParameters. A table of parameters of the influence. These parameters correspond to the parameters available to the Influence interaction FOM subclass, as specified by influenceName. Different subclasses have different parameters.

The following script entry points are only used in reactive tasks:

- check(). The check() function defines the condition under which the reactive task gets triggered. It keeps checking the condition until it becomes true, at which point it starts the reactive task.
- checkInit(). The checkInit() function initializes the reactive task when it is enabled. It sets the initial conditions for the check() function. checkInit() runs when a reactive task is first enabled and runs again after the reactive task completes so that it can begin checking the condition again.

32.2.2. The Script Execution Sequence

A Lua script has several distinct sections, including the entry points described in "Script Entry Points," on page 32-6 and the global commands in the script. When a scripted task is assigned to a simulation object, the following actions happen, even if the scenario is paused:

1. The script environment is initialized. This step defines the *vrf* and *this* objects and all of the class functions in the Lua API.
2. The script text is loaded. All Lua statements in the global scope are executed. Normally, the script defines (assigns initial values to) global variables here.
3. The init() function is called.

This sequence is followed whenever a scripted task is assigned, including the following situations:

- A scripted task is assigned while the scenario is running.
- A simulation object has a script assigned at the beginning of a scenario and the scenario is loaded or rewound.
- You modify the script or script metadata while the script is running.

If the simulation is running, after the init() function is called VR-Forces begins calling the tick() function at the period specified by the setTickPeriod() function (default: 0.5 seconds).

Suppose a script "test" contains the following code:

```
print("Global statements")
t = {}
function init()
  print("Init function")
  vrf:setTickPeriod(0.5)
  t[1] = 0
end
function tick()
  table.insert(t, vrf:getExerciseTime())
  print("Tick ", #t)
end
```

If the script is assigned to a simulation object and then the scenario runs for a short time, the console output would be as follows:

```
Global statements
Init function
DtCgfDispatcher::controlScenario: run
Tick 2
Tick 3
Tick 4
DtCgfDispatcher::controlScenario: pause
```

Figure 32-1 illustrates the control flow for a scripted task.

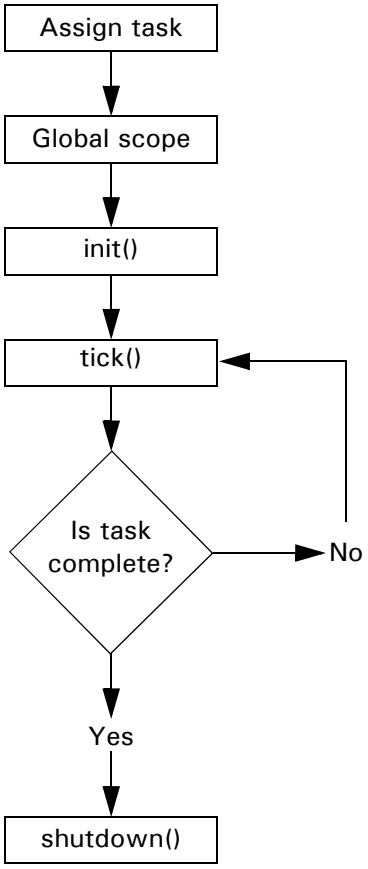


Figure 32-1. Scripted task control flow

The Script Execution Sequence for Reactive Tasks

Figure 32-2 illustrates the script execution sequence for reactive tasks. It is distinguished from that for scripted tasks by the addition of the checking process.

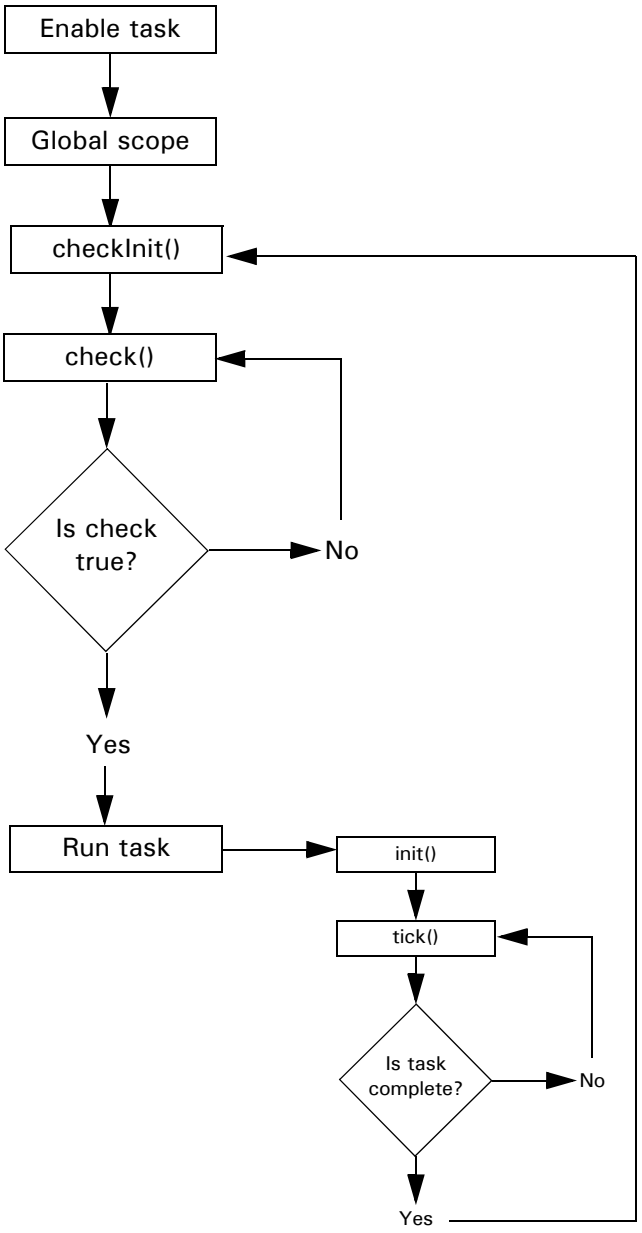


Figure 32-2. Reactive task control flow

The Script Execution Sequence for Saved Scenarios

When a scenario is saved (checkpointed), VR-Forces first runs saveState() and then saves the Lua state in the scenario file. (Checkpointing a scripted task has some limitations. For details, please see "Limitations for Checkpointing Scripts," on page 32-11. For general details about checkpointing, please see "Saving a Scenario," on page 7-21.) When a saved scenario is loaded, the following steps are taken to restore the scripted task:

1. The script environment is initialized. This step defines the *vrf* and *this* objects and all of the class functions in the Lua API.
2. The script text is loaded. All Lua statements in the global scope are executed. Global variables defined and initialized in this part of the script are re-initialized.
3. VR-Forces restores the values of global variables to the saved values. These values replace the initial values assigned in step 2.
4. The loadState() function is called.



The init() function is not run when a scenario is reloaded.

Assume that the "test" scripted task described in the previous section is saved at the point shown in the sample code (t = 4). When the scenario is run from the checkpoint, the console output will be as follows:

```
DtCgfDispatcher::controlScenario: run
Tick 5
Tick 6
DtCgfDispatcher::controlScenario: pause
```

32.2.3. Limitations for Checkpointing Scripts

When VR-Forces saves (or checkpoints) a scenario, it saves the following aspects of the Lua state in the checkpoint:

- The values of global variables that contain strings, numbers, and objects (of the classes listed in "Lua Classes," on page 32-2).
- Tables, but only table values that contain strings, numbers, objects, and tables. Tables cannot contain values that refer to the same table, either directly or indirectly through another table.

The following aspects of the Lua state do not get saved in a checkpoint:

- Variables containing functions or threads.
- Variables declared in the global scope with the keyword `local` preceding the declaration.
- Table entries that do not use numbers or strings as keys.

Lua scripts that contain elements that do not get saved can still be restored after a checkpoint if the global statements in the script reset the elements. For example, a variable containing a function is not saved in the scenario file, but if this variable is initialized in the global Lua statements, it is reset when those statements are loaded. However, if the value of this variable was changed during script execution (in the init() or tick() functions), then its value will not be restored correctly.

Tables that reference themselves in the table do not get saved properly.

To demonstrate how scripted tasks get saved, assume the following script:

```
print("Global statements")
a = 0
t = {foo = 0, bar = 1}
f = function () print("hello") end
function init()
  print("Init function")
  vrf:setTickPeriod(0.5)
  a = 100  -- change a
  t.foo= 10 -- change t.foo
  t.bar = nil -- remove t.bar
  f = function () print("world") end -- change f
end
function tick()
  print ("Tick. a = ",a, "t.foo = ", t.foo, "t.bar = ", t.bar)
  f()
  print()
end
```

The output the first time the scripted task runs is as follows:

```
Global statements
Init function
DtCgfDispatcher::controlScenario: run
Tick. a = 100     t.foo = 10      t.bar = nil
world
```

When the scenario is saved and then reloaded, and the simulation is run again, the following is printed:

```
DtCgfDispatcher::controlScenario: run
Tick. a = 100     t.foo = 10      t.bar = 1
hello
```

A comparison of the outputs reveals that variable `a` and table entry `t.foo` were restored to the correct values. However, table entry `t.bar` and function `f` were re-initialized in the global statements, but were not restored to the values they had when the scenario was saved.

Saving the State of Global Variables

Global variables get saved when a scenario gets checkpointed in one of the following ways:

- If the checkpoint mode is AllGlobals, all global variables defined are saved.
- If the checkpoint mode is CheckpointStateOnly, the script only saves variables that are part of the checkpointState table. If you remove this value, it defaults to the behavior of saving all globals.

If you want to change the mode, call setCheckpointMode(AllGlobals) to save all globals or setCheckpointMode(CheckpointStateOnly) to save only those variables in the checkpointState table. They get reinitialized when a checkpointed scenario is loaded.

```
vrf:setCheckpointMode(CheckpointStateOnly)
```

The following code, from *vrfutil.lua*, defines some global values:

```
-- Global booleans to help define which checkpoint state to use
AllGlobals = false
CheckpointStateOnly = true
```

For more information about vrf:setCheckpointMode, please see the VR-Forces Lua documentation.

32.2.4. Editing Scripts While a Scenario is Running

You can add, remove, and edit scripts during scenario creation or while a scenario is running. If a script is not being used by any simulation objects, editing or deleting it has no effect on the scenario. If you change a script that is being executed by a simulation object, when you save the change the script may be stopped, restarted, or continue to execute depending on what has changed.

Table 32-2 lists the effect on scripts and scripted tasks running as subtasks if you edit the code or metadata:

Table 32-2: Side effects of editing scripts

Change to script	Result
Change code or any metadata except the script ID. Entity type is still supported.	The script restarts. Changes to the Action Category take effect when the scenario restarts.
Entity types supported changes.	The script stops for simulation objects that no longer support it.
Script ID changes.	The script continues to execute. Any script that uses this scripted task as a subtask will fail until its code is updated to use the new script ID.

If you changed the simulation object types that can use the script, then it is added to or removed from the menu for the appropriate simulation object types. (This change takes place immediately.)

32.3. Tasks and Subtasks

The primary way that a scripted task carries out actions is through starting subtasks for the simulation object running it or sending tasks to other simulation objects. All C++ tasks and other scripted tasks are available for sending as subtasks or tasks.

If a scripted task needs to give a task to the simulation object that is running it, it does so as a subtask. (The simulation object's current task is the scripted task itself, so any action that it needs to take is a subtask to the scripted task.) Subtasks are started using the vrf:startSubtask() function.

If a scripted task needs to give a task to a simulation object other than the one running it, it uses the vrf:sendTask() function.

A startSubtask() or sendTask() function specifies the task name and task parameters, if any. The name must be the name of one of the C++ tasks, as listed in the Task page of the Lua API Documentation, or the script ID of another scripted task.

The task parameters are passed as a Lua table, with parameter names as the indices and the parameter values as the values. The valid parameters for C++ tasks are listed in the Lua API documentation. For scripted tasks, the parameter names are the parameter names defined for that task. You can view them by opening a scripted task in the Edit Script dialog box. Any parameters that are not specified when starting the task use their default values.

The following example subtask gives a Move to Waypoint task to the simulation object running the scripted task:

```
vrf:startSubtask("move-to", {destination = "Waypoint 1"})
```



If a task does not have any parameters, startSubtask() must represent the parameters as {}. For example:

```
vrf:startSubtask("wait", {})
```

The following example task sends a Patrol Between Waypoints task to the simulation object named "someEntity":

```
vrf:sendTask(someEntity, "patrol-two-points",
  {first_control_point="Waypoint 1", second_control_point="Waypoint 2"})
```

If a task is started with an invalid name or invalid parameters, a runtime error will occur in the script.

Whenever a task or subtask is started, a task ID integer is returned. This task ID is used to monitor or stop the task.



If a task is started on a simulation object that cannot execute it, the task will enter the Complete (failed) state shortly after the start call is made. This may not happen until a subsequent tick of the scripted task.

32.3.1. Monitoring the Status of Tasks and Subtasks

Once a task is started, the scripted task that started it can monitor it using its task ID. A task can be in one of three states: Running, Complete, or Canceled, as follows:

- Running. The task was started, but has not completed or been canceled.
- Complete (success). The task has completed, and called endTask(true) to end itself.
- Complete (failure):
  - The task has ended itself by calling endTask(false).
  - The task failed to start because the simulation object was incapable of executing this task.
- Canceled:
  - The task was explicitly canceled using stopTask().
  - The task was replaced by a conflicting task.
  - The user issued a Skip Task.



The success or failure of task completion is defined by the task being executed, and may not be consistent between task types.

Subtasks are monitored with the functions:

- isSubtaskComplete(taskID)
- subtaskResult(taskID)
- isSubtaskRunning(taskID)
- isSubtaskCanceled(taskID).

Tasks are monitored with the functions:

- isTaskComplete(taskID)
- taskResult(taskID)
- isTaskRunning(taskID)
- isTaskCanceled(taskID).

32.3.2. Stopping Tasks

Any task or subtask that has been started from a scripted task can be stopped by that task before it has completed. This is done using stopTask(taskID) or stopSubtask(taskID).

32.3.3. Task Parameters

Task parameters are the parameters that are gathered from the user, or set by a parent task, when a task is started. They are passed into the new task when it starts. A script gets the value of a parameter with the syntax `taskParameters.`*`parameterName`*, as in the following example:

```
waypoint = vrf:createWaypoint({location=taskParameters.waypointLocation})
```

For scripted tasks, the task parameters are available in a Lua table named taskParameters available in the global scope. The table contains items named using the parameter names specified in the parameters section of the Scripted Task dialog box when the task was created. The types of these parameters depend on the parameter types selected by the user in this dialog box.

Table 32-3 lists the parameter types you can specify when you create a scripted task and the corresponding Lua types in the taskParameters table.

Table 32-3: Task parameters and Lua types

Parameter Type	Lua Type
Active Sonar Mode	
Aggregate Formation	String.
Altitude Label	View only. No Lua type.
Altitude MSL	Number (meters).
Altitude Rate Change	Double (meters/second).
Animated Movement	String.
Bomb Resource	String (resource name).
Boolean	Boolean. Must be true or false (lowercase).
Choice (Drop-Down List)	Number (Index of the option chosen).
Choice (Image Drop-Down List)	Number (Index of the option chosen).
Choice (Multiple Option Buttons)	Number (Index of the option chosen).
Choice (Single Option Button)	Number (Index of the option chosen).
Color	Color: rgba (name red, green, blue, alpha from 0 - 255).
Date/Time	Integer (seconds in UTC).
Depth MSL	Double (meters).
DI-Guy Animation	String.
DI-Guy Appearance	String.
DI-Guy Gesture	String (gesture name).
Direct Fire Weapon	String (weapon name).
Distance	Number (meters).

Table 32-3: Task parameters and Lua types

Parameter Type	Lua Type
Embarkation Slot	Slot: <name>.slot, <name>.chooseAny (where slot is an integer and chooseAny is a boolean which means choose any other if that slot is not available.)
Embedded Entity	String (embedded entity name).
Emitter	Number (Index of the emitter system).
Entity Type	String (entity type as x:x:x:x:x:x:x).
Entity Types	Table (list) of entity types.
Filename	String (pathname).
Simulation Object Type	String (DIS type enumeration, colon separated).
Force	String.
Heading	Number (radians).
Heading Label	View only. No Lua type.
Indirect Fire Weapon	String (name of weapon).
Integer	Number.
Integer Map	Table of key of string and value of integer.
Jammer Selector	Interger (jammer selected, like Radar Mode).
Jamming Mode	Integer (mode selected, like Radar Mode).
Loadout	String.
Location (with altitude)	Location3D.
Location (without altitude)	Location3D.
Munition Resource	String (resource name).
Naval Mine Resource	String (resource name).
New Tactical Graphic	A table pf Dt:pcatopm3D Lua objects.
Observer Mode	String.
Observer Saved View	A string that contains the XML ouptut of an observer view record.
Offset Location	VectorOffset3D.
Overlay Parent	String.
Radar Mode	Integer.
Radar Selector	Integer (radar selected, like Radar Mode).
Range Label	View only. No Lua type.
Rate Label	View only. No Lua type.
Real	Number.

Table 32-3: Task parameters and Lua types

Parameter Type	Lua Type
Real Map	Map of key/value of key of string and value of double.
Resource	String (resource name).
Separator	Not applicable.
Simulation Object (Single)	SimObject.
Simulation Objects (Multiple)	Table of SimObjects.
Speed	Number (meters / second).
Sphere	Sphere: variable.center (Location3D), variable.radius (radius in meters).
String	String.
String Map	Map of key/value of key of string and value of string.
Time	Number (seconds).
Topographic Orientation	A Vector3D with bearing and inclination. Does not include roll.
Turn Rate	Number (radians / second).
Turn Rate Label	View only. No Lua type.
Visual Meta Data	String Map that contains key/value pairs of the marking and the value for the marking.
Weapon	String.

Getting the Task Parameters for a Script

You can quickly get a list of the parameters available in a scripted task or set, their type, and a description, if one was created. You can copy the list to the clipboard so that you can paste it into your script.

To get a list of the task parameters in a script:

1. Choose **Simulation** → **Scripts**. The Scripts dialog box opens.
2. Select the script whose parameters you want.
3. Choose **Script** → **Edit Script Meta Data**. If this is a system script, you might be prompted to continue. If so, click Yes. The Edit Script dialog box opens.
4. Click the taskParameters button (). A window opens that lists the parameters in the script .

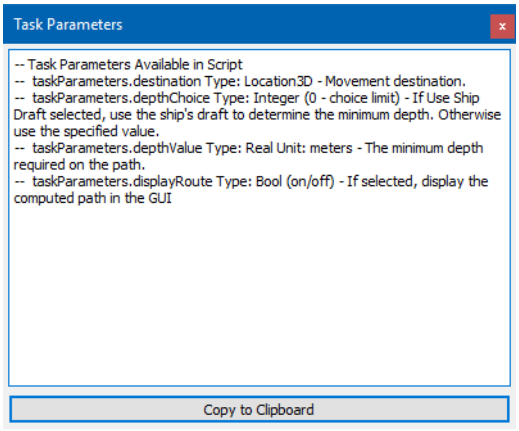


Figure 32-3. Task parameters list

5. Optionally, click Copy to Clipboard.

32.4. Geometry

The Lua interface includes classes representing geometric objects. The classes represent locations and vectors in three dimensional space. Different classes are defined for vectors in different contexts so that functions that manipulate them know what their context is and can convert between contexts automatically. Therefore, a script does not have to, for example, create rotation matrices to perform coordinate transformations between vectors that are in an entity-body context and vectors that are in a world context.

32.4.1. *Location3D*

A *Location3D* represents a point in three-dimensional (3D) space. Scripts can access the three components of the location as latitude, longitude, and altitude. There is no other access to components in any other coordinate system. Methods are available to construct a *Location3D*, to create a vector from the difference between two locations, and to generate a new point by adding a vector to a location.

32.4.2. *Vector3D*

A *Vector3D* is a direction in a north-east-down coordinate system. Scripts can access these three components of a *Vector3D*, or access bearing-inclination-range (Figure 32-4). This is the type of vector that scripts will use most often to find directions between locations, to create new locations relative to other locations, and so on.

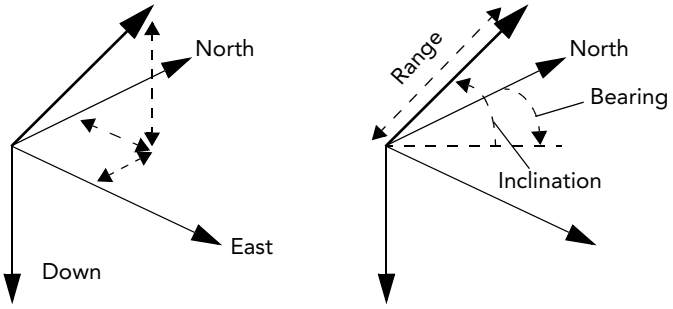


Figure 32-4. Vector3D: North-East-Down, or Bearing-Inclination-Range

When viewed from the perspective of a 3D Cartesian coordinate system that is fixed to the earth, there is no single north-east-down coordinate system. This is because "north" and "east" directions vary depending on where the observer is located on the earth. The *Vector3D* is therefore similar to a vector in topographic coordinates. At higher latitudes, "north" for two different simulation objects may not be the same direction (for example, two planes near the north pole, but at different longitudes, that are both flying "north," are not on parallel courses).

Figure 32-5 shows two north-east-down coordinate systems with origins at different locations on the earth. The coordinate systems have different orientations with respect to a coordinate system fixed to the earth.

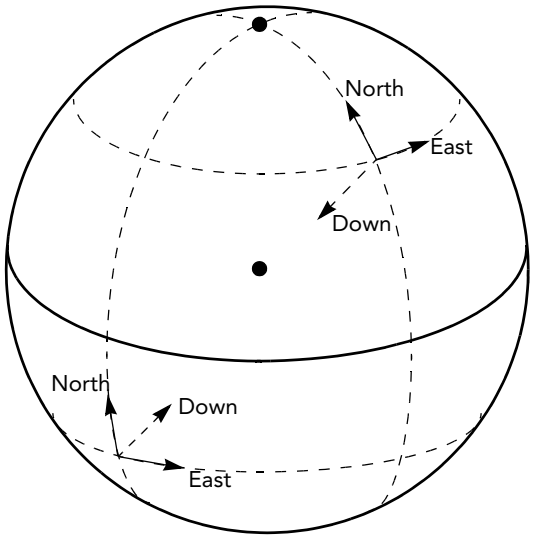


Figure 32-5. Coordinate systems with different origins

A *Vector3D* is not strictly a vector, in that it is not a straight line in a Cartesian coordinate system fixed to the earth. Just as the direction "east" changes (viewed in the Cartesian coordinate system) to follow the curvature of the earth, so does a *Vector3D*. This definition is different from topographic coordinates, which are Cartesian coordinates fixed to a point on the surface of the earth. Implications of this definition include the following:

- The heading of a *Vector3D* from one location to another is the heading of the shortest route around the earth (the great circle route) between the locations. The length is the great circle distance.
- The inclination of a *Vector3D* between two locations with the same altitude is 0.0, even when one location is over the horizon from the other.
- A *Vector3D* with length equal to the circumference of the earth, and inclination 0.0, when added to a location on the surface of the earth will produce a location coincident with the first location.

32.4.3. *VectorOffset3D*

A *VectorOffset3D* is a set of three values that define an offset from some reference direction defined by a *Vector3D*. The three values represent *right-forward-up*, where:

- *forward* is in the same direction as the reference.
- *right* is to the right in a horizontal plane. Horizontal means the plane defined by an inclination of zero; this definition implies that the "roll" component of the reference direction is assumed to be zero.
- *up* is perpendicular to *forward* and *right*, that is, *right* X *forward*.

A *VectorOffset3D* is appropriate to use, for example, to define the offsets of simulation object formation positions from a central location. Actual formation positions could be generated by transforming the offsets to *Vector3Ds* using the direction vector of the simulation object, and then adding the *Vector3Ds* to the simulation object location.

32.4.4. *VectorGeoc3D*

A *VectorGeoc3D* is a vector in a Cartesian coordinate system fixed to the earth (geocentric coordinates). The actual coordinate meanings and values are not accessible to the script, because they are considered to be abstract. A *VectorGeoc3D* can be generated only by taking the difference of two locations, or by transforming a *Vector3D* at a particular location.

A *VectorGeoc3D* between two locations is the line-of-sight vector between the locations. Therefore:

- A *VectorGeoc3D* between two points on opposite sides of the earth, at the same altitude, will go through the earth. (A *Vector3D*, by contrast, will go around the earth parallel to the surface.)
- The angle between two *VectorGeoc3D* vectors corresponding to north for two simulation objects near the north pole will be the difference in their longitudes. (The angle between their *Vector3Ds* for north, by contrast, will be zero.)

The Lua interface does not have any simulation functions that require *VectorGeoc3Ds*. However, they may be manipulated to compute angles and so forth.

32.5. Coding Messages for Translation

VR-Forces has a utility, translationFileCreate, that can extract the user-visible text in a script dialog box and make it available for translation. (For details, please see "Localizing the Graphical User Interface," on page 2-11.) If a script contains messages such as printWarn, printInfo, and so on, you can code these messages so that they are also extracted to the translation file.

To code messages for translation, use the following syntax:

```
printWarn(vrf:trUtf8("Text message."))
```

You can include variable data in the message, as follows:

```
printWarn(vrf:trUtf8("Some text %1. Some more text
  %2):arg(variable1):arg(variable2))
```

where an argument can be a string or number.

For more information about coding for translation, please see the !HowToPrint page in the Lua documentation (*./doc/luadoc/index.html*).

32.6. Error Detection

The VR-Forces Lua implementation can detect syntax errors and runtime errors.

When you save a script, it is checked for syntax errors. If an error is found, it is reported in an error message. In Figure 32-6, the `if` statement in line 38 is missing the `then` keyword.

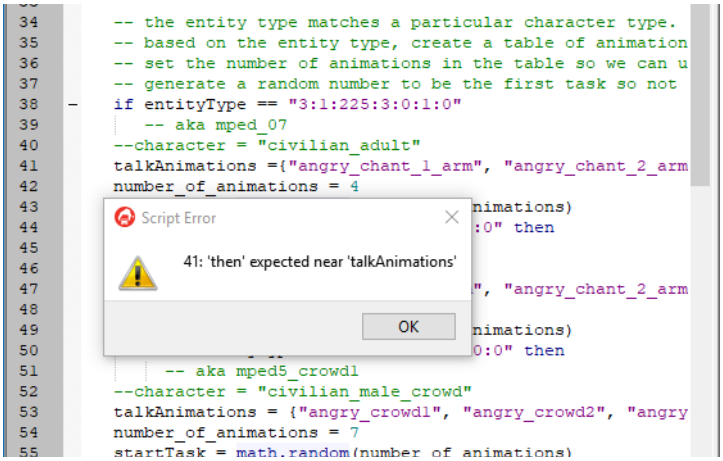


Figure 32-6. Lua syntax error



Detection of a syntax error does not prevent the script from being saved. If you do not fix the error and try to use the script, the back-end generates an error.

When you run a script, the back-end can detect errors due to incorrect use of user functions and the Lua API functions, such as calling an object that does not exist, or trying to access feature data before it is paged in. When a runtime error is detected, the back-end generates an error message. When the back-end generates an error, the script that caused the error is displayed in bold, red type in the Scripts dialog box (Figure 32-7). Therefore, you may want to keep the dialog box open while you are testing new or edited scripts.

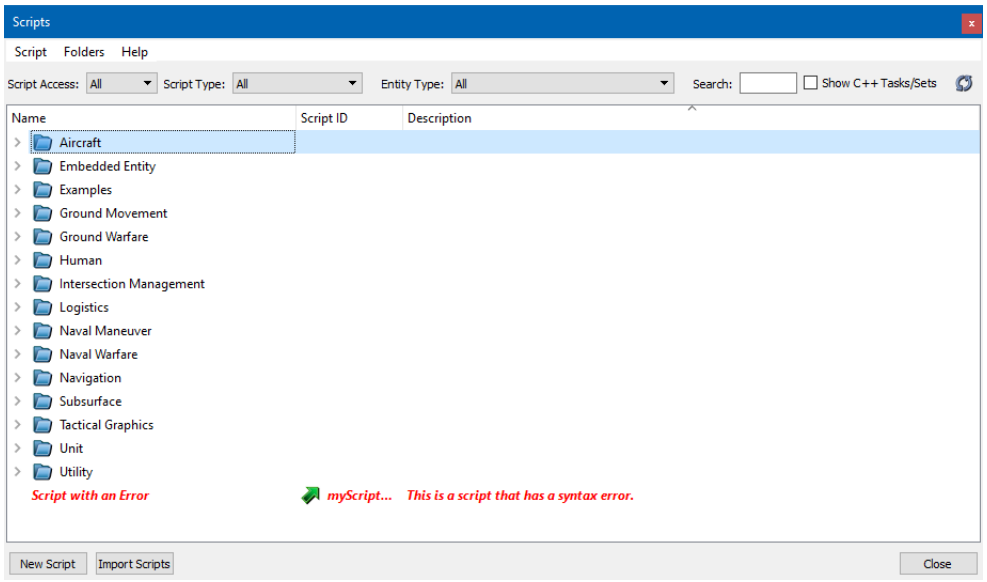


Figure 32-7. Lua error message

To view an error message:

1. In the Scripts dialog box, select the script that has generated an error.
2. Choose **Script** → **View Script Error**. An error message is displayed.

Script errors are also sent to the console in the Information dialog box.

32.7. Interactive Scripts

You can write scripts that request input from the VR-Forces user. Script questions are sent to the simulation object's object console and to the Object Console Summary Panel. Users can answer questions from the Object Console Summary Panel.

To create interactivity in a script, use the vrf:askUserQuestion() Lua function. When a script sends a question to the console, the Answer Questions button becomes active. When the user clicks the button, a set of choices is displayed. The choices can be presented as buttons, as option buttons, or as a drop-down list.

The script can do whatever it wants with the user input.

32.8. A Basic Scripted Task

VR-Forces includes many scripted tasks, all of which can serve as examples of how to write one. The Create and Move to Waypoint scripted task is a basic example that shows you how to:

- Set up metadata.
- Include the VRF Utilities module.
- Initialize a subtask ID.
- Use the init() function.
- Create a waypoint using input from the task's dialog box.
- Send a set data request to a simulation object, using input from the task's dialog box.
- Use the tick() function.
- Send a subtask and get the subtask ID.
- Check for completion of the task.
- Stop the task.

The Lua code is comprehensively commented. This section supplements the comments.

32.8.1. Create and Move to Waypoint Metadata

Figure 32-8 shows the metadata for the Create and Move to Waypoint scripted task.

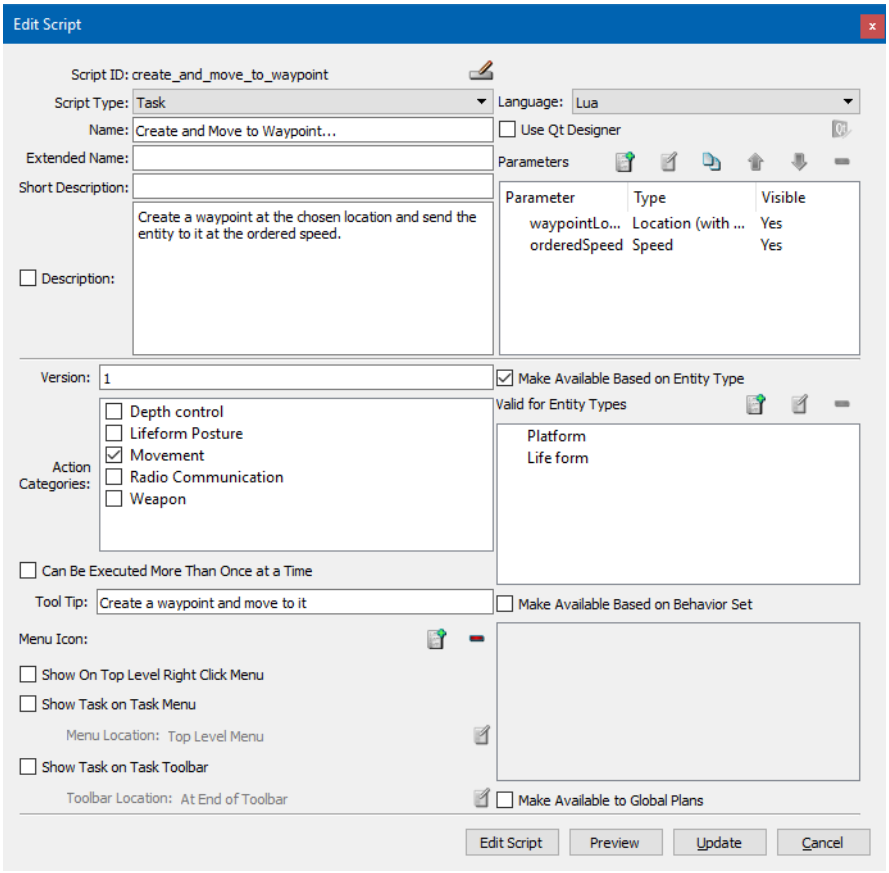


Figure 32-8. Create and Move to Waypoint script

To view the metadata:

1. Choose **Simulation** → **Scripts**. The Scripts dialog box opens.
2. Expand the Examples folder (Figure 32-9).

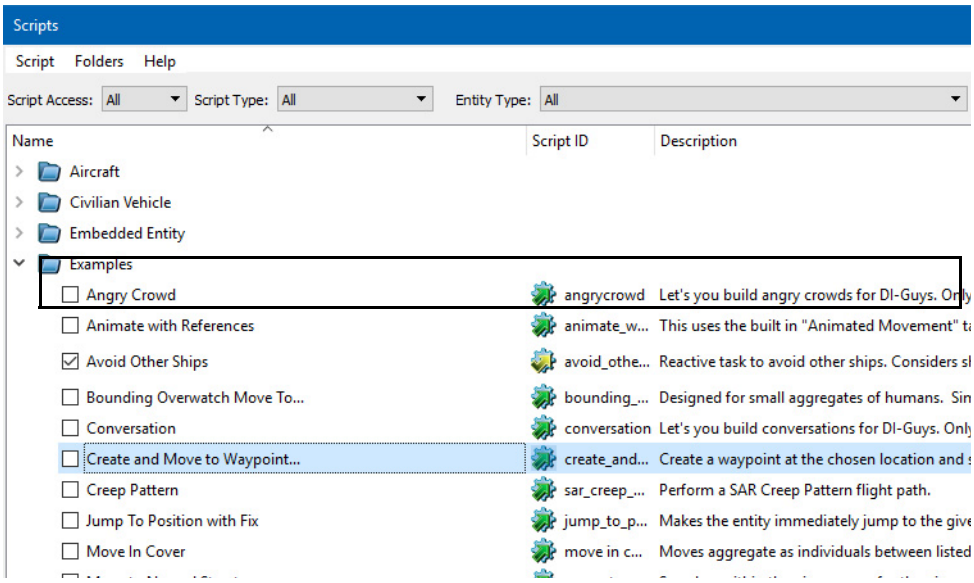


Figure 32-9. Scripts dialog box

3. Select Create and Move to Waypoint.
4. Choose **Script** → **Edit Script Meta Data** or double-click the script name. The Edit Script dialog box opens (Figure 32-8).

Note how the Name column of the Create and Move to Waypoint entry in the Scripts dialog box matches the Menu Text in the Edit Script dialog box. The Script ID and Description fields are also drawn from the scripted tasks meta-data.

Now let's look at the parameters. This scripted task has two parameters, waypointLocation and orderedSpeed. To see the definition of a parameter, select it and click the Edit button (). Figure 32-10 shows the dialog boxes that specify the parameters.

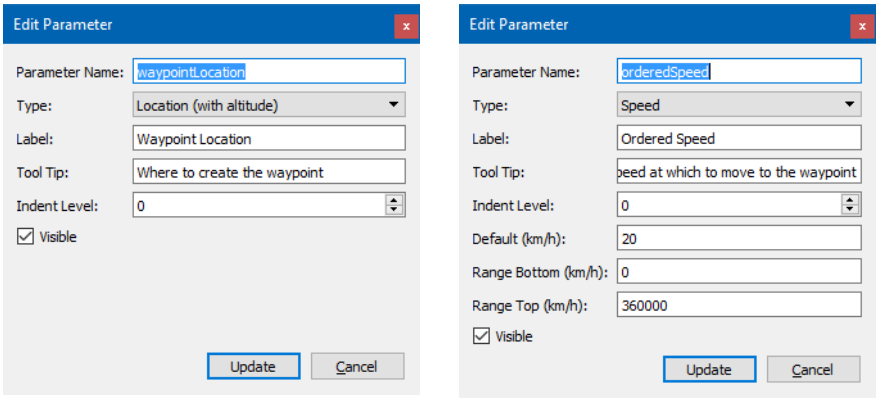


Figure 32-10. Parameter definitions

Figure 32-11 shows the dialog box that is displayed when you assign a simulation object this task. Note that the description of the task on the dialog box matches the description in the metadata. The parameter labels in the dialog box match the labels specified for the two parameters. The labels are not the same thing as the parameter name. The parameter name is the name used in the Lua script for this scripted task.

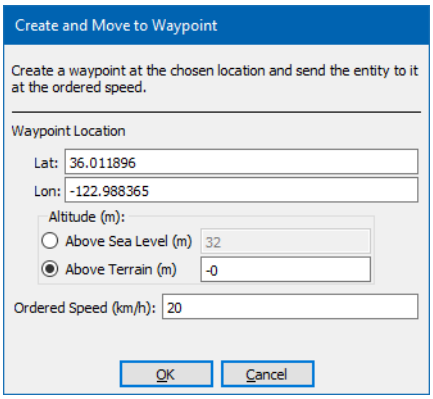


Figure 32-11. Create and Move to Waypoint dialog box

When this scripted task is assigned to a simulation object, the user will specify the waypoint and the ordered speed. Then the scripted task executes. Now we will look at the Lua script to see what happens.

32.8.2. The Create and Move to Waypoint Lua Script

The entire Lua script for this scripted task (without comments) is as follows:

```
1    require "vrfutil"
2    moveToWaypointSubtaskId = -1
3    function init()
4      vrf:setTickPeriod(0.5)
5
6      waypoint =
       vrf:createWaypoint({location=taskParameters.waypointLocation,
7        force=this:getForceType()})
8
9      vrf:sendSetData(this, "set-speed",
10       {speed=taskParameters.orderedSpeed})
11   end
12   function tick()
13     if moveToWaypointSubtaskId == -1 then
14        moveToWaypointSubtaskId = vrf:startSubtask("move-to",
15          {control_point=waypoint})
16     end
17     if vrf:isSubtaskComplete(moveToWaypointSubtaskId) then
18        vrf:endTask(true)
19     end
20   end
21   function shutdown()
22     vrf:deleteObject(waypoint)
23   end
```

The comments in the script describe the control flow. The essence is that using the input from the task dialog box, the waypoint gets created and the speed gets set in init() (line 3). The first time the task is ticked, it sends the Move to Waypoint subtask. Thereafter, each time the task is ticked (line 12), it checks to see if the subtask is complete (line 13). When the subtask is complete (line 17), the task is ended and the waypoint is removed.

The functions in this script are all in the *vrf* class, which is described in "Lua Classes," on page 32-2. The createWaypoint() function (line 6) takes its location from the waypointLocation parameter that was defined in the scripted task metadata and whose value was provided in the Create and Move to Waypoint dialog box when the task was assigned. Similarly, the ordered speed is set with the sendSetData() function (line 9) and the value is taken from the orderedSpeed parameter for the task. The script gets the parameter values with the `taskParameters.`*`parameter`* syntax.

This script uses three of the entry point functions – init(), tick(), and shutdown(). If you were to checkpoint the scenario while the script was executing, the standard save actions would take place, but no script-specific actions would be taken, because there is no saveState() function in the script. Similarly, no special action would be taken when the scenario was loaded again because loadState() is not used.

For details about the parameters and return values of the functions, please see the VR-Forces Lua API documentation. Similarly, the tasks and set data requests that you can assign with the startSubtask() and sendSetData() functions are described on the Task and Set pages of the Lua API documentation. You should review the lists of functions to see what you can do in a scripted task.

32.9. A Simple Scripted Set

This example script shows how to create a scripted set. It takes two input values from a dialog box using setParameters. Then it uses the sendSetData() function to set the values for the entity. It needs an endSet() function to finish the process.

```
require "vrfutil"
function init()
  vrf:setTickWhilePaused(true)
  vrf:setTickPeriod(0.5)

end
function tick()
  vrf:sendSetData(this, "set-speed", {speed=setParameters.setSpeed})
  vrf:sendSetData(this, "set-force", {force=setParameters.setForce})
  vrf:endSet()
end
```

The following scripted task does exactly the same thing as the scripted set. It sets the speed and force of the selected simulation object. The difference is that since this is a scripted task, it interrupts the task that the simulation object is executing. Try these two scripts to see the difference.

```
require "vrfutil"
function init()
  vrf:setTickWhilePaused(true)
  vrf:setTickPeriod(0.5)

end
function tick()
  vrf:sendSetData(this, "set-speed", {speed=taskParameters.setSpeed})
  vrf:sendSetData(this, "set-force", {force=taskParameters.setForce})
  vrf:endTask()
end
```



As you can see there is nothing in the scripts that identifies them as a scripted task or scripted set other than the endTask() and endSet() functions. VR-Forces determines the type of script from the myScriptType value in the metadata. Scripted task types are defined in *./include/vrfutil/scriptedTaskMetaData.h*.

32.10. A Simple Reactive Task

The difference between a scripted task and a reactive task is the use of the check() and checkInit() functions in the reactive task. The check() function tests a condition at whatever tick rate you specify. If the condition returns true, the task runs.

The simple reactive task in this section monitors the speed of a simulation object. When it exceeds 15 meters per second, the simulation object is destroyed. The script works as follows:

1. The checkInit() function sets the tick rate to 1.0 second.
2. In check(), VR-Forces gets the speed of the simulation object and prints it to the console.
3. It tests to see if the speed is greater than 15 meters per second. If the condition is met, the function returns true.
4. If the function returns true, the script executes init() and begins to tick().
5. In the tick() function, the simulation object is destroyed and the task is complete. Note that in the executeSetData() function, set-destroy does not have parameters, but the command must include empty braces.

```
function checkInit()
  vrf:setTickPeriod(1.0)
end

function check()
  currentSpeed = this:getSpeed()
  print("speed is:", currentSpeed)
  if currentSpeed > 15 then
    return true
  end

  return false
end

function init()
  vrf:setTickPeriod(0.5)
end

function tick()
  vrf:executeSetData("set-destroy", {})
  vrf:endTask(true)
end
```

32.11. Background Processes

Background processes are scripts that run continuously in the background. When you create a background process, you specify which simulation object types it is valid for. Thereafter, whenever you create those simulation object types the background process initializes as soon as the simulation object is created. The supported simulation object type can be as general or as specific as you want it to be, based on the entity type enumeration that you specify.

Background processes are used for simulation object actions that happen continuously in a scenario, such as consuming or receiving supplies, calculating the simulation object's footprint, and automatically attacking opposing forces.

Background processes have one important restriction - they cannot issue tasks to simulation objects or stop their tasks.

There is no indication in the GUI that a background process is running. The only way to know which background processes are available and which simulation objects they might be affecting is to examine them in the Scripts dialog box.

The following code is a very simple background process that checks a simulation object's speed every five seconds. If the speed exceeds three meters/second, it prints a message to the console. The script's metadata specifies that it applies to lifeforms. Therefore, this background process will run for each lifeform entity in the scenario.

```
require "vrfutil"
function init()
  vrf:setTickPeriod(5.0)
end

function tick()
  currentSpeed = this:getSpeed()
  if currentSpeed > 3 then
    print("Speed is:", currentSpeed, ". Going too fast.")
  else
    print ("Speed is OK."
  end
end
```



33. Using Qt Designer for Script Dialog Boxes

You can let VR-Forces create dialog boxes for scripted tasks and sets using default layout rules or you can design the dialog boxes yourself using Qt Designer.

Introduction to Using Qt Designer for Scripts 33-2
Selecting Qt Designer for Dialog Box Layout 33-2
Adding Parameters When Using Qt Designer 33-2
Specifying Choices for the Choice Parameters in Qt Designer 33-6
Specifying Categories for a Simulation Object in Qt Designer 33-7
Adding Design Elements 33-8

33.1. Introduction to Using Qt Designer for Scripts

When you create a scripted task or set that requires input from a user, VR-Forces can automatically build a dialog box with fields for each parameter. These dialog boxes are built using a default layout scheme. This layout scheme is functional, but it may not always capture relationships between different parameters, particularly dependencies. Also, the resulting dialog boxes may not be as esthetically pleasing as well-designed dialog boxes should be. As an alternative to automatic dialog box creation, you can use Qt Designer to lay out dialog boxes. This capability is integrated into the script creation process.

The VR-Forces graphical user interface is built using Qt, a cross-platform GUI SDK. Qt Designer is a utility that developers can use to quickly lay out GUI design elements. It is beyond the scope of this manual to fully explain how to use Qt Designer. This chapter will provide enough information about it to get you started.

33.2. Selecting Qt Designer for Dialog Box Layout

By default, VR-Forces automatically builds dialog boxes for scripted tasks and sets.

➤ To specify use of Qt Designer for dialog box layout, in the New Script or Edit Script dialog box, select the Use Qt Designer check box.

33.3. Adding Parameters When Using Qt Designer

When you create or edit a script, you can add parameters in the New Script or Edit Script dialog box. If you select Qt Designer for dialog box layout, any parameters that you have already defined in the New Script dialog box are implemented in Qt Designer using the default dialog box layout (which you can change). However, once you select Qt Designer, you cannot add any more parameters in the New Script dialog box. You can only add parameters in Qt Designer.

If you decide to use Qt Designer before you add parameters in the New Script dialog box, then all parameters get added using Qt Designer.

! If you lay out a dialog box using Qt Designer and then unselect the Use Qt Designer check box, all the layout information that you added in Designer is lost, except that if you rearranged the order of parameters, those changes affect the order in which the parameters are listed in the Edit Script dialog box. Any parameters that you added in Designer are still valid and part of the script.

To add parameters in Qt Designer:

1. In the New Script or Edit Script dialog box, select the Use Qt Designer check box (Figure 33-1).

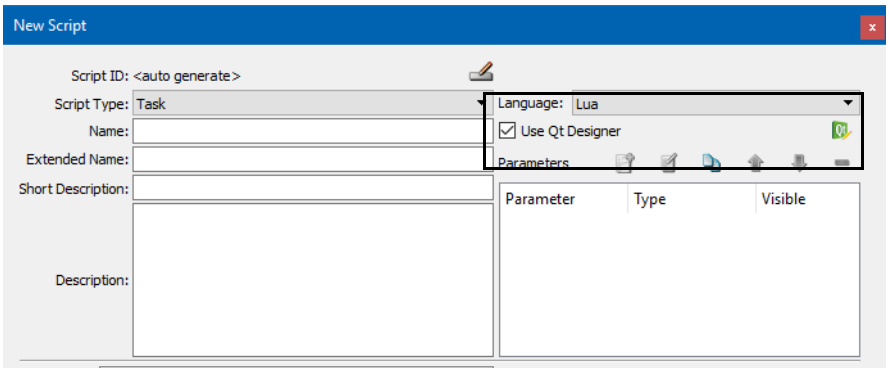


Figure 33-1. New Script dialog box

2. Click the Qt button ( ) to the right of the Qt Designer check box. Qt Designer opens (Figure 33-2). If the script already has parameters, it displays a default dialog box layout for the script. If there are no parameters, it displays a blank canvas.

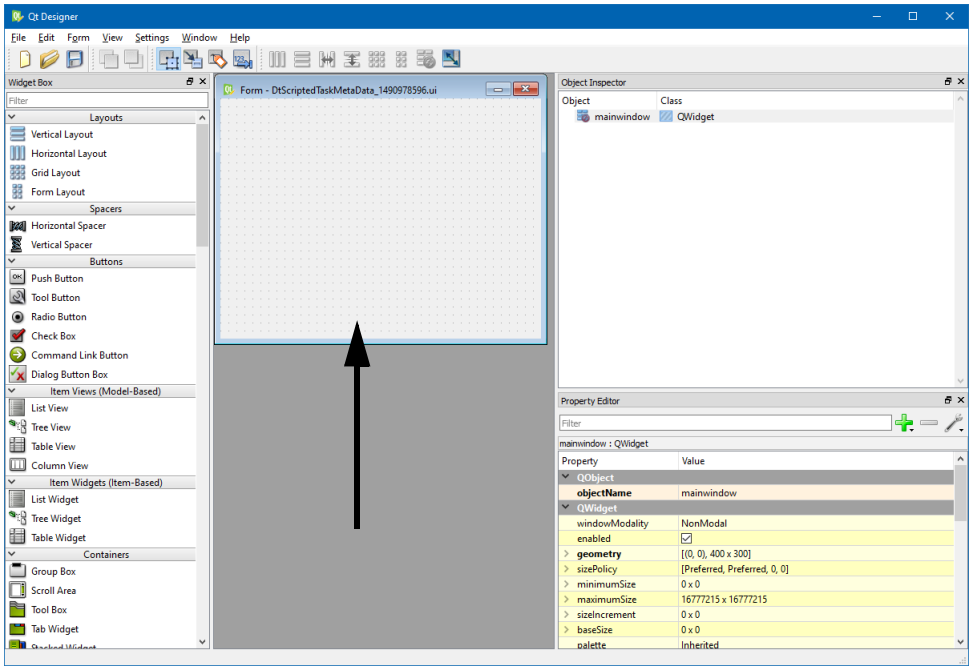


Figure 33-2. Qt Designer with blank canvas

> **!** You must use the canvas that gets created when you open Designer. If you close it and open a new one from the File menu, it might not work.

3. Scroll through the widget box until you see the list of widgets for VR-Forces script parameters (Figure 33-3).

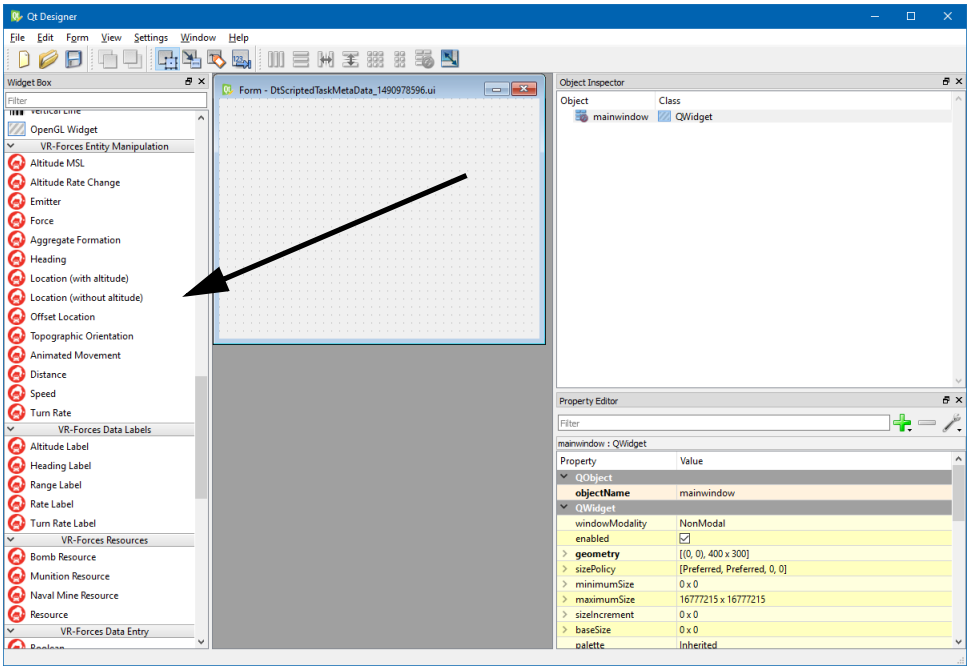


Figure 33-3. VR-Forces widgets

4. Drag a parameter onto the dialog box canvas.

5. In the Property Editor, specify the required and optional information for the parameter type, such as a label, tooltip, or default value.

   All of the parameters have the following properties:

   – **objectName**. The name to be used by the Lua script for this parameter. For example, if you give a parameter the name **taskDestination**, in the Lua code for the script, the parameter will be accessed as `taskParameters.taskDestination`.

   – **label**. This is the text used on the dialog box that is generated for the task. If you leave this field blank, the parameter name is used.

   – **toolTip**. Optional text for a tool tip.

   – **indent**. The number of pixels to indent the parameter on the task dialog box. This allows you to apply some formatting to the dialog box.

   – **isVisible**. Specifies that the parameter be included on the script dialog box. (You might want a parameter to be available in the code, but not have it set by a user when the script is assigned.)

   Many parameters let you specify a default value or state.

   Some parameters let you specify a range for the acceptable values:

   – **rangeBottom**. The lowest acceptable value for this parameter.

   – **rangeTop**. The highest acceptable value for this parameter.

   The Boolean parameter type lets you create a check box. You can specify that the default be checked or unchecked.

6. Add design elements such as dividers and labels as desired.



Nothing in Qt Designer will prevent you from adding non-VR-Forces properties. However, you cannot use non-VR-Forces properties in scripts.

7. Choose a layout option on the Form menu. Doing so preserves your layout. If you do not select a layout option, VR-Forces might rearrange the layout when it displays the dialog box for the script.

8. In Qt Designer, choose **File** → **Save** or click the Save button. If this is a system script, Qt Designer saves a UI file with the layout information. If this is a scenario script, the layout information gets saved in the scenario.

9. Close Qt Designer.

33.3.1. Specifying Choices for the Choice Parameters in Qt Designer

The Choice parameters require a list of choices, which get added to the dialog box as option buttons or as a list. You can add choices, edit them, and delete them.

Option buttons have a property called **buttonGroup**. Buttons within the same button group are mutually exclusive. If you add more than one group of option buttons to a script, you must edit the **buttonGroup** property to make sure that the different groups have a different value for the property.

To add a drop-down list of choices:

1. From the VR-Forces Data Entry section, drag the Choice (Drop-Down List) widget onto the canvas.
2. In the Property Editor, click the Value column for the **values** property. The text changes to Change String List.
3. Click Change String List. The Dialog - Qt Designer dialog box opens. Add a string for each choice you want to make available, as follows:
   a. Click New. A text box is added to the list.
   b. Type the string you want to be displayed.
   c. Repeat this procedure for each string you want to be in the drop-down list.
   d. Click OK. The dialog box closes. The strings you entered are listed in the Value column for the values property.

To add option button choices:

1. From the VR-Forces Data Entry section, drag the Choice (Option Button) widget onto the canvas. It creates a widget with one button that has the text Radio Button.
2. In the Property Editor, click the Value column for the **values** property. The text changes to Change String List.
3. Click Change String List. The Dialog - Qt Designer dialog box opens. Add a string for each choice you want to make available, as follows:
   a. Double-click the text Radio Button to make it editable.
   b. Change the text to something meaningful for this parameter.
   c. Click New. A text box is added to the list.
   d. Type the string you want to be displayed for the next option button.
   e. Repeat step c and d for each option button you want.
   f. Click OK. The dialog box closes. The strings you entered are listed on the canvas and in the Value column for the values property.

33.3.2. Specifying Categories for a Simulation Object in Qt Designer

The Simulation Object parameter lets you build a list of categories on which to filter the display of simulation object types in a dialog box.

To add a category to the simulation object list:

1. Double-click the Filter widget on the form. A dialog box opens. The top line reads Filters and it has several icons.
2. Click the Add button ( ) on the Filters line. The Choose Filter dialog box opens.
3. Select a filter from the Filters list.
4. Click OK.
5. Repeat the procedure to add as many filters as you want.
6. Click OK.

Editing the Category List for a Simulation Object Parameter

You can edit the list of filter categories for a Simulation Object parameter. The edit operation is essentially a replacement of the current category with a new one. You can achieve the same effect by deleting the current category and adding a new one.

To edit the list of filter categories in a Simulation Object parameter:

1. Double-click the Filter widget on the form. A dialog box opens. The top line reads Filters and it has several icons.
2. In the window, select the filter that you want to edit.
3. Click the Edit button ( ) on the Filters line. The Choose Filter dialog box opens.
4. Select the new filter category.
5. Click OK. The original category is removed and the new one is added.
6. Click OK.

To delete a category from a Simulation Object parameter:

1. Double-click the Filter widget on the form. A dialog box opens. The top line reads Filters and it has several icons.
2. In the window, select the filter that you want to delete.
3. Click the Delete button ( ). The category is removed from the list.
4. Click OK.

33.4. Adding Design Elements

You can add design elements such as dividers, group boxes, and indentation. You do so by dragging the appropriate widgets from the list onto the canvas and setting their properties. Please see Qt Designer documentation for details about how to design dialog boxes.



VI. Set Data Requests

VI.1. Setting Simulation Object State and Attributes

Set data requests affect the state or behavior of a simulation object, such as its speed or posture. They take effect immediately. Assignment of a set data request does not override the task that a simulation object is executing. By contrast, if you assign a simulation object a task, it stops its current task and begins the new task. You can send set data requests from a plan or from the Set menu.

You can create new set data requests using the same Lua scripting process that you can use to create scripted tasks.

The Set menu is context sensitive. It only shows the options that apply to the selected simulation object.

The graphical user interface does not necessarily know about the state of a simulation object. When you open a dialog box for a Set command, any data in the dialog box represents either default values or the most recently used value. Do not assume that a dialog box shows current values.



You can check the current status of most simulation object state data in the Information dialog box.

You can filter simulation object selection lists. For details, please see "Filtering the Object Selection Lists," on page 25-9.

The instructions for setting simulation object state direct you to use options on the Set menu. However, all the options available on the main menu are also available on context-sensitive menus. A limited number of set data requests are available on the Sets Toolbar and on the Last Selected Objects Panel.

You can also set the state of tactical graphics. For details, please see Chapter 41, *Setting the State of Tactical Graphics*.



34. Setting the State of Entity-Level Simulation Objects

This chapter explains how to set simulation object states for entities and units in entity-level scenarios. Some of the set data requests also apply to aggregate-level scenarios. They are cross referenced from Chapter 35, *Setting the State of Aggregate-Level Simulation Objects*.

Active Sonar Mode 34-4
Add Jamming Target 34-4
AI Enabled 34-5
Aiming 34-6
Altitude 34-7
Appearance 34-8
Armed 34-9
Assume Unknown is Hostile 34-9
Attention Grabbing 34-9
Automatically Close Doors 34-10
Capabilities 34-10
Collision Avoidance Types 34-11
Concealed 34-12
Counter Measures Auto Launch 34-12
Current Speed 34-13
Destroyed 34-13
Detonation Fuse Type 34-14
DI-Guy Characteristics (Appearance, Head, Body Weight) 34-15
DI-Guy Hand Item 34-15
Disembarked 34-16
Embarked 34-16

Emitter 34-17
Engagement Zones 34-18
Force 34-18
Formation 34-19
Heading 34-20
Hide Occupants 34-20
IFF 34-21
Invisible 34-22
Invulnerable 34-22
Jammer State 34-22
Jamming Mode 34-23
Lase Autonomous 34-24
Laser Code 34-24
Laser Power and Wavelength 34-25
Location 34-26
Look At 34-26
Markings 34-27
Navigation Enabled 34-27
Navigation Preferences 34-28
Notify Level 34-28
Orbit 34-29
Ordered Speed 34-30
Posture (Human) 34-31
Preferred Trajectory Height 34-32
Radar Mode 34-32
Remote Sensor 34-33
Remove Jamming Target 34-33
Reorganize 34-33
Resources 34-34
Restore 34-34
Rules of Engagement 34-35
    How Fire-When-Fired-Upon Works 34-35
Sector Blanking 34-36
Sector of Responsibility 34-37
Sensor Aim 34-38
Sensor Enabled 34-38
Sensor Field of View Color 34-39
Sensor Zoom 34-40
Sonar Depth 34-40
Spot Reports 34-41

Stow Hand Item 34-42
Superior 34-42
Surrendered 34-42
Synchronize Laser Code 34-43
Target 34-43
Tasked by Superior 34-44
TDL Messaging 34-44
Tracer Use 34-44
Unit State 34-45

34.1. Active Sonar Mode

If an entity has an active sonar system, you can set the sonar mode.



RPR FOM 1.0 does not support active sonar.

To set the active sonar mode:

1. Select the entity.
2. Choose **Set** → **Sensors** → **Active Sonar Mode**. The Active Sonar Mode dialog box opens.
3. In the System ID list, select the sonar system that you are setting. The entities provided with VR-Forces just have one active sonar system.
4. In the Mode list, select a sonar mode or select Off to disable active sonar.
5. Click OK.

34.2. Add Jamming Target

The Add Jamming Target set data request lets you specify a target for an entity to jam. You can only specify one target at a time using this set, but you can issue it multiple times to specify multiple jamming targets.

To specify a jamming target:

1. Select the entity that has a jamming capability.
2. Choose **Set** → **Jammer** → **Add Jamming Target**. The Add Jamming Target dialog box opens.
3. Select the entity that you want to jam.
4. Click OK.

34.3. AI Enabled

The AI Enabled set data request lets you disable intelligent decision making by simulation objects. By default, simulation objects can respond to their environment by avoiding obstacles, firing on targets, and so on. In some cases, exercise participants do not want simulation objects to take any actions other than those specifically tasked to them by plans or human players. AI Enabled lets you remove independent action from simulation object behavior.

AI Enabled disables the following behaviors:

- Collision avoidance.
- Automatic firing at other entities.
- Reacting to being fired upon. That is, there is no suppression or fleeing.

To enable or disable artificial intelligence:

1. Select the simulation object.
2. Choose **Set** → **Action** → **AI Enabled**, or click the AI Enabled button () on the Sets Toolbar. The AI Enabled dialog box opens.
3. In the Enabled list, select Yes or No.
4. Click OK.

34.4. Aiming

The Aiming set data request tells an entity to aim at a point, an object, or an angle. It does not cause the entity to shoot. If the entity acquires a target, it aims at the target and fires. Then it returns to the aiming behavior specified by the set data request.

An entity can move while it is aiming. If an entity is moving and it can no longer reasonably aim in the specified direction, it stops aiming. For example if the entity is aiming at a point and the entity's movement causes the point to be behind it, the entity stops aiming (unless it has a turret that can aim in that direction). If the entity stopped aiming while it was moving, when it stops moving, it turns to aim in the specified way.

You can also use this set data request to stop the aiming behavior.

To have an entity aim at a location or object:

1. Select the entity.
2. Choose **Set** → **Engagement** → **Aiming**. The Aiming dialog box opens.
3. Select the aiming option you want.
4. Specify the appropriate parameter for the aiming choice:
   - Aim at Point – specify the location.
   - Aim at Object – select the object.
   - Aim at Angle – specify the azimuth and elevation.
   - Cease Aiming – no parameters required.
5. Click OK.

34.5. Altitude

The Altitude request changes the altitude of simulation objects immediately. If you want a fixed-wing or rotary-wing entity to move to an altitude, use the Move to Altitude or Fly Altitude task. You can set the altitude manually or place an entity at a specific level in a multi-level terrain feature, such as a building.

To set an entity's altitude:

1. Select the entity.
2. Choose **Set** → **Position** → **Altitude**, or click the Altitude button () on the Sets Toolbar. The Altitude dialog box opens. It displays the entity's current altitude.
3. Select MSL (mean sea level) or AGL (above ground level) to specify the base for the altitude value.
4. Type the new altitude.
5. Click OK.

34.6. Appearance

You can set appearance attributes for simulation objects. The appearance attributes that are available for a simulation object are determined by its entry in *./appData/settings/vrfGui/SISO-REF-010.xml*. The default list of appearance attributes is based on the DIS standard.

The Lifeform State field of the Appearance dialog box sets the posture for an entity. If VR-Forces does not support a selected posture, it simulates the entity using the closest similar posture. If a posture involves walking or running, VR-Forces chooses the posture to use based on the entity's speed. For more information about setting posture, please see "Posture (Human)," on page 34-31.

If you want to set the appearance of a human character, use the DI-Guy Appearance set data request, described in ("DI-Guy Characteristics (Appearance, Head, Body Weight)," on page 34-15).



In aggregate-level scenarios, the Appearance set data request is included in the list of commands for the Send Radio Set task and the list of commands available on the Command menu in plans. However, it is not supported and if used, will fail. Its presence is due to the lack of context sensitivity for these menus.

To set appearance values:

1. Select the simulation object.
2. Choose **Set** → **Appearance** → **Appearance**. The Appearance dialog box opens. The settings displayed are the current settings for the simulation object.
3. Change appearance settings as desired.
4. Click OK.

34.7. Armed

IEDs, car bombs, and suicide bombers must be armed to detonate. They are all armed by default. You can disarm them and re-arm them.



IEDs, car bombs, and suicide bombers have radios. Therefore, you can arm and disarm them using the Send Radio Set task.

To arm or disarm an explosive device:

1. Select the explosive device.
2. Choose **Set** → **Engagement** → **Armed**. The Armed dialog box opens.
3. Select an option from the list.
4. Click OK.

34.8. Assume Unknown is Hostile

The Assume Unknown is Hostile set data request means that a simulation object associated with an engagement zone can assume that any unknown simulation object that enters the engagement zone is hostile.

To set unknown objects in an engagement zone to be hostile:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Assume Unknown is Hostile**. The Assume Unknown is Hostile dialog box opens.
3. Select No or Yes.
4. Click OK.

34.9. Attention Grabbing

The Attention Grabbing set data request specifies how interesting an object should be to humans as they randomly pay attention to other objects in their environment.

To set the level of attention for an object:

1. Select the object whose level of attention you want to set.
2. Choose **Set** → **Disposition** → **Attention Grabbing**. The Attention Grabbing dialog box opens.
3. Select an option for the amount of attention that this object should get.
4. Click OK.

34.10. Automatically Close Doors

If Automatically Close Doors is enabled, when a human entity goes through a door that can be opened or closed and there is no one else behind it, the door closes automatically. Default: enabled.

You can also close doors with the Close Door task ("Close Door," on page 28-3). For more information about opening and closing doors, please see Chapter 62, *Dynamic Terrain*.

To enable or disable automatically closing doors:

1. Select the entity.
2. Choose **Set** → **Action** → **Automatically Close Doors**. The Automatically Close Doors dialog box opens.
3. Select Yes to enable, No to disable.

34.11. Capabilities

The capabilities that are available for a simulation object are determined by its entry in *./appData/settings/vrfGui/SISO-REF-010.xml*. The default list of capabilities is based on the DIS standard.

To set simulation object capabilities:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Capabilities**. The Capabilities dialog box opens. It displays the current settings for the simulation object.
3. Change the settings you want to change.
4. Click OK.

34.12. Collision Avoidance Types

Sometimes entities exhibit undesired collision avoidance behavior because they try to avoid objects that are close to their line of travel, but not necessarily blocking it. You can enable and disable avoidance of entities, buildings, and trees on a per-entity basis. This option overrides the collision avoidance settings in the object parameter database entry for the entity.



- Collision avoidance settings are saved in an entity's process state repository. However, there is no indication in the GUI what these settings are. Therefore, we recommend that you not change collision avoidance settings through the GUI indiscriminately because you may lose track of which entities are avoiding which types of objects. To permanently set collision avoidance for an entity, edit its components in the object parameter database.
- When the Collision Avoidance Types dialog box opens, it does not reflect the settings for the selected entities. It only provides a default setting. Make sure that you review all the settings in the dialog box to be sure you do not inadvertently enable or disable a setting that you do not want to change.
- The Collision Avoidance Types set data request lets you specify collision avoidance only for broad categories of objects. You can set collision avoidance more specifically in the object parameter database.

To set collision avoidance for specific object types:

1. Select an entity.
2. Choose **Set** → **Action** → **Collision Avoidance Types**. The Collision Avoidance Types dialog box opens.
3. Select the check boxes for object types to avoid and clear the check boxes for the object types that the entity should not try to avoid.
4. Click OK.

34.13. Concealed

The Concealed request sets a simulation object's concealment attribute to True or False. For entity-level scenarios, this only affects the DIS or RPR FOM data sent over the network. It does not affect the simulation. For aggregate-level scenarios, when a simulation object is concealed, its sensor signature is changed so that it is harder to detect.

To set the concealment attribute:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Concealed**. The Concealed dialog box opens.
3. Select an option from the list.
4. Click OK.

34.14. Counter Measures Auto Launch

The Counter Measures Auto Launch set data request lets you enable and disable automatic launching of counter measures. If you disable automatic counter measures, you can still launch them manually using the Launch Counter Measures task ("Launch Counter Measures," on page 27-15).

To enable or disable automatic launch of counter measures:

1. Select the entity.
2. Choose **Set** → **Engagement** → **Counter Measures Auto Launch**. The Auto Launch dialog box opens.
3. Select an option from the list.
4. Click OK.

34.15. Current Speed

The Current Speed set data request sets the speed of a simulation object immediately and changes the ordered speed to this new speed value. By contrast, Ordered Speed ("Ordered Speed," on page 34-30) causes a simulation object to accelerate or decelerate to the desired speed.

A negative value causes the entity to go backwards if it supports backwards movement.



For all simulation objects except humans and fixed-wing aircraft that are in the air, if you set the current speed and the object does not have a task, it begins to move at that speed and then slows down until the speed reaches 0. Fixed-wing aircraft that are in a default orbit mode change their speed to the new current speed. Humans might take a step, but then will stop moving.

To set current speed:

1. Select the simulation object.
2. Choose **Set** → **Action** → **Current Speed**. The Current Speed dialog box opens.
3. In the Current Speed text box, set the speed.
4. Click OK.

34.16. Destroyed

You can set a simulation object's state to be destroyed. This can be helpful when you want a simulation object to be destroyed without having to first simulate its destruction by another simulation object.

To set an object's state to destroyed:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Destroyed**. The simulation object is immediately destroyed.

34.17. Detonation Fuse Type

The Detonation Fuse Type set data request lets you configure detonation of improvised explosive devices (roadside IED entity), suicide bombers, and car bombs. You can set them to detonate immediately, at a certain time delay, or based on the proximity of a target entity. If you specify proximity detonation, you can set the radius of the proximate area. If you specify time, you must set the time in seconds until detonation. Detonation Fuse Type implicitly arms the device. (For details about arming devices, please see "Armed," on page 34-9.)



IEDs, car bombs, and suicide bombers have radios. Therefore, you can detonate them using the Send Radio Set task.

To set the detonation fuse type:

1. Select the object that you want to configure.
2. Choose **Set** → **Engagement** → **Detonation Fuse Type**. The Detonation Fuse Type dialog box opens.
3. In the Fuse Type list, select the fuse type (Timed, Immediate, or Proximity).
4. If you selected a Timed fuse, in the Time box, specify the elapsed time, in seconds, at which the IED should detonate.

   If you selected a Proximity fuse, in the Proximity box, specify the radius of the area in which a simulation object will trigger the detonation.
5. Click OK.

34.18. DI-Guy Characteristics (Appearance, Head, Body Weight)

The DI-Guy Characteristics set data request determines the clothing and physical characteristics of a lifeform when it is displayed in a 3D simulation that uses DI-Guy characters.

The DI-Guy Characteristics set data request applies only to lifeform entities.

To set DI-Guy appearance, head, or body weight:

1. Select the entity.
2. Choose **Set** → **Appearance** → **DI-Guy Characteristics**. The DI-Guy Characteristics dialog box opens.
3. Optionally, in the 3D view only, select the Apply Selected Appearance Immediately check box. This displays the selected entity with the new appearance so that you can see what it looks like before you apply the change.
4. Optionally, filter the appearance list by choosing options in the various filter lists.
5. To change the appearance, select an appearance from the DI-Guy Appearance list.
6. To change the head, select an option in the DI-Guy Head list.
7. To change the weight of the character, slide the DI-Guy Body Weight slider or change the value in the text box (the range is 0.5 - 5.0).
8. Click OK.

34.19. DI-Guy Hand Item

The DI-Guy Hand Item set data request lets you change the hand item that a character has. DI-Guys can have a variety of hand items. For example, a civilian can have a cell phone, a pistol, or a camcorder, while a soldier can have a variety of weapons.

To set a DI-Guy's hand item:

1. Select the entity.
2. Choose **Set** → **Appearance** → **Hand Item**. The DI-Guy Hand Item dialog box opens.
3. Select a hand item from the list.
4. Optionally, in the 3D view only, select the Apply Selected Hand Item Immediately check box. This displays the selected entity with the new hand item so that you can see what it looks like before you apply the change.
5. Click OK.

34.20. Disembarked

The Disembarked set data request immediately disembarks a simulation object (as opposed to simulating disembarkation, as done by the Disembark task). It is the equivalent of the **Objects** → **Disembark** command. However, Disembarked allows you to include immediate disembarkation in a plan.

To set a simulation object to be disembarked:

1. Select the simulation object. (If you are using the 2D icon model, embarked simulation objects are not displayed, so you have to select it in the Objects List Panel.)
2. Choose **Set** → **Embarkation** → **Disembarked**.

34.21. Embarked

The Embarked set data request immediately embarks a simulation object (as opposed to simulating embarkation, as done by the Embark task). It is the equivalent of the **Objects** → **Embarkation** → **Embark On** command. However, Embarked allows you to include immediate embarkation in a plan.

The default location for this set data request is the center of the parent simulation object (0,0,0). It does not use preconfigured slots like the Embark task does. If you are using a Stealth observer mode and want placement of embarked simulation objects to look good, you will have to calculate offsets for each simulation object you embark.

If you are using HLA and want to use the embarkation feature, you must use RPR FOM 2, draft 17 or later.

To set a simulation object to be embarked:

1. Select the simulation object to be embarked.
2. Choose **Set** → **Embarkation** → **Embarked**. The Embarked dialog box opens.
3. Select the simulation object to embark on.
4. Optionally, specify an embarkation offset.
5. Click OK.

34.22. Emitter

The Emitter set data request enables or disables emitters on simulation objects that have them.

The radar sensor has an emitter component that models basic electromagnetic emissions. A simulation object can have multiple emitter components. Each emitter component can have a list of modes and each mode can have multiple beams.

Emission properties (azimuth, elevation, effective radiated power, and so on) for each beam are specified in the emitter component descriptor in the object parameter database. Emission properties are published by the emitter component.

An emitter component can publish itself as a standard beam type or a tracking beam type. The type is determined by the setting for the targeting-capable parameter in the object parameter database. It applies to all beams emitted by the specific component. If targeting-capable is False, the beam is a standard beam. If targeting-capable is True, it is a tracking beam.

If an emitter component has targeting-capable set to True, it registers for set-target messages. If you set a target for the simulation object on which the emitter is mounted, it publishes the target entity as its tracking target for all of the components beams.



- Setting a target does not cause the beam to actually track the target. The beam's behavior is determined by the parameters in its description in the object parameter database.
- The emitter continues to publish the targeted entity as its tracking target until you set another target, regardless of the status of that entity in the exercise. There is no way to set the target to "none".
- Please see "Configuring Emitters," on page 79-2, for an explanation of how to configure individual and multiple emitters in the object parameter database.

Each emitter has an ID that is assigned automatically, starting with 0. Unless you change the system definition files, all default fixed-wing aircraft have one emitter with ID 0.

To turn on electromagnetic emissions:

1. Select the simulation object.
2. Choose **Set** → **Sensors** → **Emitter**. The Emitter dialog box opens.
3. Type the emitter ID. If the simulation object only has one emitter, its ID is 0. If it has more than one emitter, IDs get incremented by 1 for each additional emitter. There is no way to identify the emitter ID for a particular emitter, so you will have to use a trial-and-error approach to identify the correct emitter ID to use.
4. In the Emitter list, select On.

34.23. Engagement Zones

The Engagement Zone set data request lets you associate one or more engagement zones with a simulation object.

To associate a simulation object with an engagement zone:

1. Select the simulation object.
2. Choose **Set** → **Engagement** → **Engagement Zones**. The Engagement Zones dialog box opens.
3. Select one or more engagement zones in the list.
4. Click OK.

34.24. Force

You can change a simulation object's force during runtime. This might be useful in a simulation in which neutral forces begin assisting one force or another, or in which opposing forces capture friendly force weaponry and begin using it. You can also set the force for combat engineering objects in aggregate-level scenarios.

When you change a simulation object's force, its:

- Echelon ID changes to reflect the new force.
- Icon changes to the appropriate force color and type.
- Force ID changes.

The entity enumeration does not change.

To set the force of an object:

1. Choose **Set** → **Disposition** → **Force**. The Force dialog box opens.
2. Select a force from the list.
3. Click OK.

34.25. Formation

VR-Forces supports the following formations:

- Column (default)
- Line
- Wedge
- Vee
- User-defined formations.

Formations are defined in formation files in the directory *./data/simulationModelSets/<model_set>/vrfSim/formation*. Formation files have the extension *.frm*. In addition to the named formations provided by MAK, VR-Forces supports user-defined formations. If you create a new formation in the Simulation Object Editor, it automatically gets added to the formation list for the unit types that support it.



The vee and wedge formations provided with VR-Forces place the unit leader at the end of one of the arms of the formation. If a unit has fewer than seven or eight members, it will not be symmetrical.

When you set the formation for a disaggregated unit, the simulation objects that make up the unit snap to formation immediately. If you want simulation objects to simulate moving into formation, rather than snapping to formation, use the Move Into Formation task.

If you set the formation for an aggregated unit, the unit remembers the formation and uses it the next time it disaggregates.

To set a unit's formation:

1. Select the unit whose formation you want to set.
2. Choose **Set** → **Unit** → **Formation**. The Formation dialog box opens.
3. In the Formation list, select the formation that you want the unit to use.
4. Click OK.

For information about configuring formations and about how to create user-defined formations, please see "Configuring Formations," on page 78-2.

34.26. Heading

The Heading request instantly sets a simulation object's heading. If you want a simulation object to move to a new heading by pivoting or executing a K-turn, use the Turn To Heading task. You can also set a simulation object's heading manually. For details, please see "Specifying an Object's Heading Dynamically," on page 14-20.

To set an object's heading:

1. Select the simulation object.
2. Choose **Set** → **Position** → **Heading**, or click the Heading button (ⓘ) on the Sets Toolbar. The Heading dialog box opens. It displays the simulation object's current heading.
3. In the Heading text box, type a heading, in degrees.
4. In the Relative To list, select one of the following options:
   - North. Turn to the specified absolute heading.
   - Current Heading. Turn the specified number of degrees relative to the current heading.
   - Host (if embarked). If embarked, turn to the specified heading relative to the host simulation object.
5. Click OK.

34.27. Hide Occupants

The Hide Occupants set data request overrides the Show Occupants setting on the Entity Display Settings page for the entity for which you set it. VR-Forces can automatically display civilians in the driver's seat of civilian vehicles. These occupants are not simulated objects, they are just visualized. You might want to disable this setting for performance reasons or because you are embarking a simulated person in the vehicle.

For more information about displaying occupants in a vehicle, please see "Displaying DI-Guys in Vehicles," on page 44-8.

To display or hide occupants of a vehicle:

1. Select the simulation object.
2. Choose **Set** → **Appearance** → **Hide Occupants**. The Hide Occupants dialog box opens.
3. Select an option in the Occupant Visibility list.
4. Click OK.

34.28. IFF

Friendly fixed-wing entities can model a NATO Identification Friend from Foe (IFF) transponder.

- Mode 5 and mode C are equivalent.
- The dialog box does not force you to enter valid codes. It truncates invalid entries to the correct length (but not necessarily valid data).
- When you open the dialog box, it reflects the current IFF settings for the entity, if any.
- You can view an entity's IFF settings on the IFF tab of its Information dialog box.

If an aircraft has an IFF Code Changer system and it enables the IFF Code Changer reactive task, its IFF code will change every half hour based on the entries in a CSV file. By default, entities are not configured with this system and the task is inactive. A sample CSV file is provided: *./data/simulationModelSets/EntityLevel/extra/iff-codes.csv*.

To set the IFF transponder:

1. Select the entity.
2. Choose **Set** → **Sensors** → **IFF**. The IFF dialog box opens (Figure 34-1).

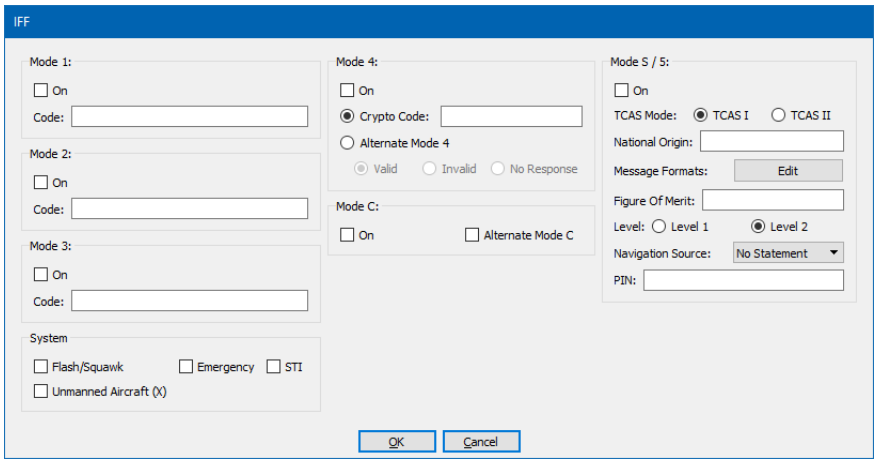


Figure 34-1. IFF dialog box

3. Fill out fields as desired.
4. Click OK.

34.29. Invisible

The Invisible set data request causes simulation objects to be invisible on the map display in 2D or 3D. They are still listed in the Objects List Panel.

To set a object to be invisible:

1. Select the simulation object.
2. Choose **Set** → **Appearance** → **Invisible**. The Invisible dialog box opens.
3. Select an option from the Invisible list.
4. Click OK.

34.30. Invulnerable

If a simulation object is invulnerable, it does not take damage from munitions.

To set an object to be invulnerable:

1. Select the entity.
2. Choose **Set** → **Disposition** → **Invulnerable**, or click the Invulnerable button ( ) on the Sets Toolbar. The Invulnerable dialog box opens.
3. Select an option from the list.

34.31. Jammer State

The Jammer State set data request lets you enable and disable jamming for an entity. For more information about jamming, please see "Electronic Warfare," on page 9-20.

To set the jammer state:

1. Select the entity.
2. Choose **Set** → **Jammer** → **Jammer State**. The Jammer State dialog box opens.
3. In the State list, select On or Off.
4. Click OK.

34.32. Jamming Mode

The Jamming Mode set data request lets you specify the emitter ID for a jammer and the mode it uses. The options are:

- Deception jamming.
- Mixed mode jamming.
- Mixed noise.

For more information about jamming, please see "Electronic Warfare," on page 9-20.

To specify the jamming mode:

1. Select the jamming entity.
2. Choose **Set** → **Jammer** → **Jamming Mode**. The Jamming Mode dialog box opens.
3. The Emitter ID is Jamming Emitter.
4. Select a mode in the Mode list.
5. Click OK.

34.33. Lase Autonomous

By default, entities with laser capability lase targets autonomously as part of the built-in target acquisition and firing process. You can specifically enable and disable autonomous lasing.

To enable or disable autonomous lasing:

1. Select the entity.
2. Choose **Set** → **Laser Designator** → **Lase Autonomous**. The Lase Autonomous dialog box opens.



The Lase Autonomous dialog box does not show the current lasing state of the entity.

3. Select an option from the list (Off or On).
4. Click OK.

34.34. Laser Code

The Laser Code set data request sets the code for laser beams sent by an entity and the code that the entity's missiles seek when they fire. By default, VR-Forces calculates a laser code for each laser-capable entity when it is created. You can change the laser code to a code of your choice. For information about the consequences of setting laser codes, please see "Lasing Targets," on page 9-18.

To set a laser code:

1. Select the entity.
2. Choose **Set** → **Laser Designator** → **Laser Code**. The Laser Code dialog box opens.
3. Type a number between 1111 and 1788, using only the digits 1-8.
4. Click OK.

34.35. Laser Power and Wavelength

The Laser Power and Wavelength set data request lets you specify the power and wavelength for a laser designator. VR-Forces does not use these settings when it simulates laser designators. However the information is sent over the network for use by applications that can use it. For example, the color of the laser designator can be set based on the wavelength if it is visualized in another application.

The dialog box does not report the current value, so if you only want to change one value, set the other one to -1 to keep the current value.

To set the laser power and wavelength for a lasing entity:

1. Select the entity.
2. Choose **Set** → **Laser Designator** → **Laser Power and Wavelength**. The Laser Power and Wavelength dialog box opens.
3. Specify the Power value, in watts. Setting it to -1 preserves the current value.
4. Specify the Wavelength value, in microns. Setting it to -1 preserves the current value.
5. Click OK.

34.36. Location

To move a simulation object instantly from one place to another, you can set its location or you can drag it to a new location. For information about dragging simulation objects, please see "Dragging an Object to a New Location," on page 14-23. For ground entities, specifying an altitude lets you place an entity at different levels (floors) in buildings that have interior geometry.

To set location:

1. Select the simulation object.
2. Choose **Set** → **Position** → **Location**. The Location dialog box opens. It displays the simulation object's current location. The cursor changes to input mode.
3. Click on the terrain where you want to move the simulation object, or enter the coordinates of the location in the Location dialog box.
4. Optionally, set the altitude, as described in "Specifying an Object's Altitude," on page 14-18.
5. Click OK.

34.37. Look At

The Look At set data request tells a human entity to look at a point, an object, or an angle. This set causes the entity's head to move, not its body. It will move up to 90 degrees to either side (azimuth) and up to 45 degrees of elevation (where looking straight ahead is 0 degrees). If the target object or location is at a greater angle than the maximum allowed amount, the head does not turn.



The range of head motion is configurable in *./data/simulationModelSets/EntityLevel/vrfSim/platforms/human.ope* in the `look-at-actuator`. It is specified in radians.

You can also use this set data request to stop the look at behavior.

To have an entity look at a location or object:

1. Select the entity.
2. Choose **Set** → **Action** → **Look At**. The Look At dialog box opens.
3. Select the Look At option you want.
4. Specify the appropriate parameter for the aiming choice:
   - Look at Point – specify the location.
   - Look at Object – select the object.
   - Look at Angle – specify the azimuth (heading) and elevation.
   - Cease Looking – no parameters required.
5. Click OK.

34.38. Markings

Some vehicles are configured with texture maps that let you dynamically change exterior markings, such as a national flag, unit type, and other emblems. You can change the markings at run time. Figure 34-2 shows a Fenneck vehicle with German markings and the same model with Canadian markings.



Figure 34-2. Changing markings

If you assign this set to a unit, the markings are passed to its subordinates.

For details about configuring markings, please see "Specifying Visual Markings," on page 68-23 and "Adding Dynamically Changeable Markings to Models," on page 90-24.

To set a simulation object's markings:

1. Select the entity.
2. Choose **Set** → **Appearance** → **Markings**. The Markings dialog box opens.
3. In each list, select the markings that you want to set for this object.
4. Click OK.

34.39. Navigation Enabled

By default, ground vehicles and humans use the navigation data in navigation areas to plan movement. The Navigation Enabled set data request lets you disable use of navigation data.

Navigation is disabled on a per-entity basis. If you disable navigation for an entity, other entities still treat that entity as an obstacle for navigation purposes.

To enable or disable use of navigation data:

1. Select the entity.
2. Choose **Set** → **Action** → **Navigation Enabled**. The Navigation Enabled dialog box opens.
3. Select an option from the Enabled list.
4. Click OK.

34.40. Navigation Preferences

The Navigation Preferences set data request allows you to specify that when an entity plans its path using navigation data, it should take into account road data, if it is available.

When an entity is set to Prefer Roads, the entity gives preference to all segments along the preferred graph over those on the non-preferred movement graph. This is typically used to have vehicles prefer movement along roads, rather than cutting across open terrain, even if the distance along roads is longer.

If the movement mode is Avoid Roads, entities try not to move along roads. If they must cross a road, they take the shortest path across the road.

Entities can also ignore roads.

You can configure an entity to start up with a navigation preference or to ignore roads by editing the initial-road-preference parameter in the navigation-preference controller in the movement system definition file for this entity type.

To set navigation preferences:

1. Select the entity.
2. Choose **Set** → **Action** → **Navigation Preferences**. The Navigation Preferences dialog box opens.
3. To have the entity take road data into account, select Prefer Roads or Avoid Roads from the Navigation Preferences list. To have an entity plan its path as if there were no road data, select Ignore Roads.
4. Click OK.

34.41. Notify Level

You can set the notification level for simulation object console messages without opening the Information dialog box. You can also set the notification level for combat engineering objects in aggregate-level scenarios.

To set the notification level:

1. Choose **Set** → **Other** → **Notify Level**. The Notify Level dialog box opens.
2. Select a notification level in the list.
3. Click OK.

34.42. Orbit

The Orbit set data request lets you specify the orbit for a satellite. You can specify a Kepler orbit or a geostationary orbit. Kepler orbits are specified by two-line elements. They are available at http://celestrak.com/ and https://www.space-track.org.

To set the orbit for a satellite:

1. Select the satellite.
2. Choose **Set** → **Position** → **Orbit**. The Orbit dialog box opens.
3. To set a Kepler orbit:
   a. Select the Kepler Orbit option. The input window is enabled.
   b. Paste in the two-line element that you want to use.

   To set a geostationary orbit:
   a. Select the Geostationary Orbit option.
   b. Type the longitude of the orbit, in degrees.
4. Click OK.

34.43. Ordered Speed

The ordered speed is the speed at which you want a simulation object to move. It may not move at that actual speed due to soil type, slope, and so on. In aggregate-level scenarios, speed may be affected by modifiers. If you set the speed while a simulation object is moving, it accelerates or decelerates from the current speed to the new ordered speed, if it can.

A negative value causes the entity to go backwards if it supports backwards movement.



- Regardless of the speed that you enter, a simulation object's speed cannot exceed the value set by the **max-speed** parameter in the object parameter database.
- The Ordered Speed request sets the ordered speed of a fixed-wing entity in its current context – air or ground. The entity will not exceed the maximum speed for that context. However, if the maximum allowable ground speed is greater than the lift speed of the entity, and you assign an ordered speed greater than the lift speed, when the entity reaches that speed, it tries to take off.
- If you want a simulation object's speed to change instantly, use the Current Speed set data request ("Current Speed," on page 34-13).
- If a simulation object is not moving, this set data request has no effect.

To set the ordered speed of an object:

1. Select the simulation object.
2. Choose **Set** → **Action** → **Ordered Speed**. The Ordered Speed dialog box opens.
3. In the Ordered Speed text box, set the speed.
4. Click OK.

34.44. Posture (Human)

A human entity (civilian or dismounted infantry) has a posture that determines its position when it is not moving and its type of movement when it is moving. Table 34-1 lists the position and movement relationships for each posture.

Table 34-1: Posture relationships

Posture	When it is not moving, it is:	When it is moving, it is:
standing	standing	walking or running
kneeling	kneeling	crawling
prone	prone	crawling
running	standing	walking or running
walking	standing	walking or running
crawling	prone	crawling
parachuting	standing	walking or running
jumping	standing	walking or running
sitting	standing	walking
squatting	prone	crawling
crouching	prone	crawling
wading	standing	walking or running

- For those postures listed as “walking or running”, VR-Forces sets the movement posture based on the entity’s speed.
- If an entity does not support a particular posture, it is set to the closest supported posture.
- Setting posture in the Lifeform State field of the Appearance dialog box has the same effect as setting the posture using the Posture set data request.
- Some embarkation slots specify the posture of occupants.

To set a lifeform’s posture:

1. Select the lifeform.
2. Choose **Set** → **Action** → **Posture**. The Posture dialog box opens.
3. Select a posture from the list.
4. Click OK.

34.45. Preferred Trajectory Height

The Preferred Trajectory Height set data request specifies the preferred height for indirect fire munitions, such as Fire for Effect tasks and munitions that are not triggered by a task. Indirect fire tasks override the preferred trajectory height if they need to use a different trajectory to hit the target.

To specify a preferred trajectory height:

1. Select an entity that has indirect fire capability.
2. Choose **Set** → **Engagement** → **Preferred Trajectory Height**. The Preferred Trajectory Height dialog box opens.
3. Specify the desired altitude.
4. Click OK.

34.46. Radar Mode

The Radar Mode set data request lets you change the type of beam that an entity emitter publishes. By default, each fixed-wing aircraft has one emitter that has two beam types, search and track.

Figure 34-3 illustrates the difference in appearance between the beams for the two radar modes, in the 2D view. (The beams are not displayed unless you set emitters to be on.)

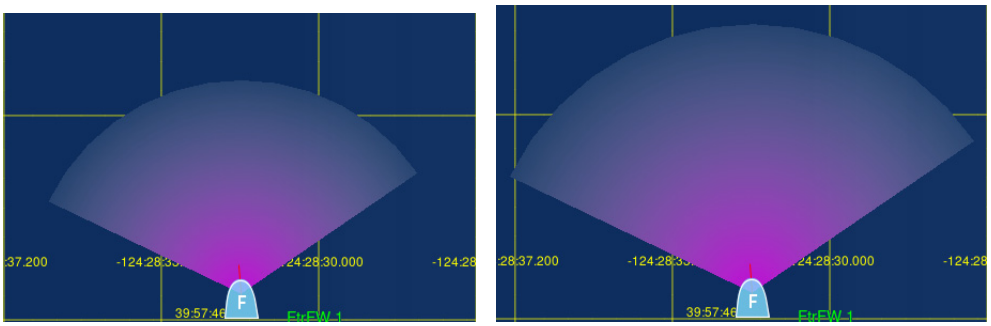


Figure 34-3. Radar mode beams

For more information about emitter beams, please see "Emitter," on page 34-17 and "Configuring Emitters," on page 79-2.

To set the radar mode:

1. Select the entity whose radar mode you want to set.
2. Choose **Set** → **Sensors** → **Radar Mode**. The Radar Mode dialog box opens.
3. If the entity has more than one emitter, select the emitter ID you want to set in the Emitter ID list.
4. Select the mode in the Mode list.

34.47. Remote Sensor

The Remote Sensor set data request lets an entity use another entity's sensors to discover sensor contacts. This set data request is only available to entities that have a Remote Sensor Receiver system.

An example of using a remote sensor is the Patriot Missile Battery. The Patriot Engagement Control System has a Remote Sensor Receiver system. It receives spot reports from the Patriot Radar AN/MPQ-65.

To specify a remote sensor for an entity:

1. Select the entity that needs to receive remote spot reports.
2. Choose **Set** → **Sensors** → **Remote Sensor**. The Remote Sensor dialog box opens.
3. Select the remote entity (the entity from which this entity wants to receive spot reports).
4. Click OK.

34.48. Remove Jamming Target

The Remove Jamming Target set data request tells an entity to stop jamming the specified target. You can only remove one target at a time using this set, but you can issue it multiple times to remove multiple jamming targets. The Remove Jamming Target dialog box lists all entities. There is no list available that only lists jamming targets.

To remove a jamming target:

1. Select the entity that is jamming.
2. Choose **Set** → **Jammer** → **Remove Jamming Target**. The Remove Jamming Target dialog box opens.
3. Select the target to stop jamming.
4. Click OK.

34.49. Reorganize

If automatic reorganization is not enabled, you can reorganize a unit by issuing the Reorganize set data request.

To reorganize a unit:

1. Select the unit that you want to reorganize.
2. Choose **Set** → **Unit** → **Reorganize**.

The change takes place immediately. For more information about reorganization, please see "Reorganizing Units," on page 21-4.

34.50. Resources

You can specify the value to assign to a resource. In entity-level scenarios, the resources provided with VR-Forces that you can set are fuel and ammunition. The object parameter database lists the ammunition types available for each type of simulation object. In aggregate-level scenarios a simulation object might not have any resources, depending on how it is configured in the Simulation Object Editor.

To specify a resource:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Resources**. The Resources dialog box opens.
3. Select a resource from the Resource list. The dialog box displays the maximum amount of the resource allowed, as configured in the object parameter database.
4. Specify a value for the resource.
5. Click OK.

34.51. Restore

Restoring a simulation object returns its resources to the values specified in the object parameter database. If the simulation object is destroyed, it returns it to life.

When you restore an entity-level unit that is in the aggregated state (using Restore), the original configuration of the unit is recreated. If you restore a unit that is in the disaggregated state, then only those subordinates still present in the simulation are restored. Any destroyed subordinates that were previously removed from the simulation as result of a disaggregate--> aggregate-->disaggregate transition are not restored. To fully recover all subordinates, restore the unit while in the aggregated state.

To restore an object:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Restore**.

34.52. Rules of Engagement

Rules of engagement specify the conditions under which a simulation object will fire at the enemy. The options are:

- Fire at Will. If an entity detects a target, it can fire automatically.
- Fire When Fired Upon. Do not fire at detected targets unless the simulation object has already been under fire.
- Fire When In Engagement Zone. If a detected target is in an engagement zone that is associated with the firing simulation object, it may fire at the target at will.
- Hold Fire. Do not fire at targets.



The rules of engagement for a fixed-wing entity are in effect whether it is on the ground or in the air. If you do not want fixed-wing entities to fire missiles while they are on the ground, you must change the rules of engagement.

To specify the rules of engagement:

1. Select the simulation object.
2. Choose **Set** → **Engagement** → **Rules of Engagement**, or click the Rules of Engagement button () on the Sets Toolbar. The Rules of Engagement dialog box opens.
3. In the Rules of Engagement list, select the rule you want to apply.
4. Click OK.

34.52.1. How Fire-When-Fired-Upon Works

A simulation object considers itself under fire if either of the following is true:

- An "Entity Impact" detonation interaction that targeted this entity has recently (within underFireTime seconds (default 120)) been received.
- Any detonation has happened within underFireDistance (default 1000 meters) of this entity.

There is no attempt to return fire specifically at the attacking entity. If a simulation object thinks that it is under fire, it will fire at any enemies that it can target.



The underFireTime and underFireDistance parameters are part of the under-fire-determination-sensor component in a simulation object's platform (OPE) file.

34.53. Sector Blanking

When Sector Blanking is enabled, targets within the specified sector are not spotted. This set data request is available to simulation objects that have a radar sensor. Sector blanking parameters are stored on a per-entity basis and are saved in the scenario and checkpoint files.

Sectors are defined by a start azimuth and end azimuth. They are relative to the entity orientation, with 0 degrees being directly ahead.

To enable or disable sector blanking:

1. Select a simulation object that has a radar sensor.
2. Choose **Set** → **Sensors** → **Sector Blanking**. The Sector Blanking dialog box opens.
3. Select or clear the Enable Sector Blanking check box.
4. If you are enabling sector blanking, specify a start azimuth and end azimuth, with 0 degrees being directly ahead of the simulation object.
5. Click OK.

34.54. Sector of Responsibility

A simulation object's sector of responsibility is used with detection tables to determine the priority of targets. Normally, the priority of targets that are outside a simulation object's sector of responsibility is lower than the priority of targets in its sector of responsibility. For information about the target detection tables, please see "Detection Tables," on page 75-16.

You must specify a sector center and a sector size. Sector center is the center of the area you wish to designate, and sector size is the size of the area, both in degrees. Degrees are measured as 0 to the front (based on the heading), increasing clockwise. If the Turret Scan Controller is mounted on an entity with an articulating turret, then the turret scans the sector of responsibility specified. Sector center is the midpoint of the area. Figure 34-4 illustrates these parameters. The sector size is approximately 140 degrees, a bit less than half the circle. The sector center is at 0 degrees.

The default sector of responsibility is 90 degrees, centered at zero degrees.

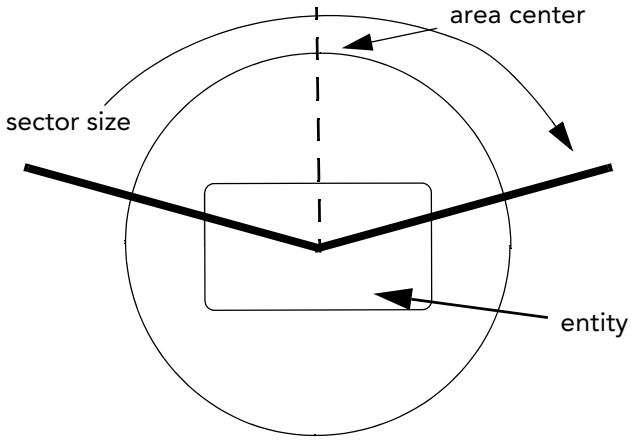


Figure 34-4. Sector of responsibility parameters

To set an object's sector of responsibility:

1. Select the simulation object.
2. Choose **Set** → **Engagement** → **Sector of Responsibility**. The Sector of Responsibility dialog box opens.
3. In the Size text box, set the size of the area to be scanned, in degrees.
4. In the Center text box, set the center of the scan area, in degrees.
5. Click OK.

34.55. Sensor Aim

The Sensor Aim set data request specifies where a gimbaled sensor should look.

You can also aim a gimbaled sensor from the Sensor View Control Panel. For details, please see "Controlling the Sensor View," on page 53-10.

To aim a sensor:

1. Select an entity that has a gimbaled visual sensor.
2. Choose **Set** → **Sensors** → **Sensor Aim**. The Sensor Aim dialog box opens.
3. Choose an aiming option and provide the required input, as follows:
   - Aim at Point. Click on the terrain or type in coordinates. By default, the altitude is the altitude of the entity.
   - Aim at Object. Select the simulation object to aim at.
   - Aim at Angle. Specify the Azimuth (the horizontal angle relative to the sensing entity) and elevation (the vertical angle that the sensor aims at).
   - Scan. By default, the sensor scans back and forth within a 180 degree range. You can change the elevation and azimuth angles in the gimbaled sensor system definition file.
4. Click OK.

34.56. Sensor Enabled

The Sensor Enabled set data request lets you enable and disable individual sensors on a simulation object.

To enable or disable a sensor:

1. Select the simulation object.
2. Choose **Set** → **Sensors** → **Sensor Enabled**. The Sensor Enabled dialog box opens.
3. In the Sensor list, select the sensor you want to enable or disable, or select All.
4. In the Enabled list, select Yes or No.
5. Click OK.

34.57. Sensor Field of View Color

The Sensor Field of View Color set data request lets you override the color that VR-Forces automatically assigns to the field of view.

To specify the color of a sensor field of view:

1. Select an entity that has a sensor.
2. Choose **Set** → **Sensors** → **Sensor Field of View Color**. The Sensor Field of View Color dialog box opens.
3. In the Sensor list, select the sensor you want to specify the color of.
4. Click the color swatch. A color picker dialog box opens.
5. Select the color you want.
6. Click OK to close the color picker.
7. Click OK.

34.58. Sensor Zoom

The Sensor Zoom set data request lets you change the field of view of a gimbaled visual sensor, essentially zooming in on the aim point.

You can also zoom a gimbaled sensor from the Sensor View Control Panel. For details, please see "Controlling the Sensor View," on page 53-10.

To zoom a visual sensor:

1. Select the entity whose sensor you want to zoom.
2. Choose **Set** → **Sensors** → **Sensor Zoom**. The Sensor Zoom dialog box opens.
3. If the entity has more than one sensor, in the Sensor list, select the sensor you want to zoom.
4. Type the zoom level you want or drag the slider.
5. Click OK.

34.59. Sonar Depth

The Sonar Depth set data request lets you set the depth at which an entity's sonar operates. This allows you to simulate a helicopter dipping a sonar sensor into the water or a vessel dipping a sensor to a lower depth. Since sonar only works when the sensor is in the water, setting the sonar depth is the only way for helicopter-born sonar to work.

The sonar stays at this depth regardless of the altitude or depth of the entity. If you set the depth to be the depth of the entity with the Use Entity Depth option, it has the effect of retracting the sonar. If you set the sonar to the depth of the entity and the entity changes depth, the sonar stays at the assigned depth.

To set sonar depth:

1. Select the entity.
2. Choose **Set** → **Sensors** → **Sonar Depth**. The Sonar Depth dialog box opens.
3. Specify the depth.
4. Optionally, select Use Entity Depth to set a dipped sonar to the depth of the entity.
5. Click OK.

34.60. Spot Reports

You can set spot reports on or off for all simulation objects (global setting) from the Spot Report Settings page of the Application Settings dialog box. You can also set them for individual simulation objects. When you set spot reports on, it means that the simulation object sends them. Visualization of spot reports is enabled or disabled for the GUI as a whole on the Spot Report Settings page. For more information about spot reports, including setting them globally, please see "Displaying Spot Reports," on page 9-2.



You cannot set a unit in an entity-level scenario to send spot reports; you can only set spot reports for individual simulation objects. However, you can select multiple individual simulation objects and set spot reports for all of them at the same time.

To set spot reports on or off:

1. Select the simulation object.
2. Choose **Set** → **Sensors** → **Spot Reports**. The Spot Reports dialog box opens. If you are turning spot reports off, skip to step 5.
3. Select one of the following options:
   - Only to Front End. Do not send spot reports to other back-ends or to other simulation objects. This option can improve performance. However, since entities do not receive the spot reports, they cannot use that information for targeting purposes.
   - Broadcast. Send spot reports to all simulation objects.
   - Send to Specific Entities. Sends to specific simulation objects.
4. If you selected Send to Specific Entities, the simulation object list is enabled. Select the simulation objects to which you want this simulation object to send sport reports.
5. Select an option from the Spot Reports Enabled list. If you choose On or Off, the setting persists regardless of the global spot report setting. If you want the simulation object to use the global spot report setting, choose Use Global Setting.
6. Click OK.

34.61. Stow Hand Item

The Stow Hand Item set data request hides or displays a hand item for a human character.

To stow or display a hand item:

1. Select the entity.
2. Choose **Set** → **Appearance** → **Stow Hand Item**. The Stow Hand Item dialog box opens.
3. In the Stow Item list select Yes or No.
4. Click OK.

34.62. Superior

The Superior set data request lets you add a simulation object (individual or unit) to a unit or move a simulation object from a unit to the force level. The superior unit must already exist. For entity-level scenarios, the superior unit must be disaggregated. For aggregate-level scenarios, the superior must be a higher echelon unit.

To specify an object's superior:

1. Select the simulation object.
2. Choose **Set** → **Other** → **Superior**. The Superior dialog box opens.
3. Optionally, filter the list.
4. Select the unit that you want to be this simulation object's superior. If you want to move the simulation object out of a unit to the force level, select the Set Force As Superior check box.
5. Click OK.

34.63. Surrendered

The Surrendered set data request only applies to lifeforms. When a lifeform is in the surrendered state, it does not fire at other simulation objects and other simulation objects do not fire at it.

To set an entity to be surrendered:

1. Select the entity.
2. Choose **Set** → **Disposition** → **Surrendered**. The Surrendered dialog box opens.
3. Select an option from the list.
4. Click OK.

34.64. Synchronize Laser Code

The Synchronize Laser Code set data request lets an entity with laser-guided weapons easily synchronize its laser code with the laser code of the lasing entity. The alternative to this command would be to use the Laser Code set data request to set the code for the lasing entity and the firing entity.

To synchronize laser codes:

1. Select the entity that has a laser-guide weapon.
2. Choose **Set** → **Laser Designator** → **Synchronize Laser Code**. The Synchronize Laser Code dialog box opens.
3. Select the lasing entity that you want this entity to synchronize with.
4. Click OK. The laser code for the selected entity is changed to match that of the lasing entity.

34.65. Target

You can command an entity to specifically target another entity. For this set data request to be effective, the target must be visible to the targeting entity. When you set a target, you override the priorities set by the target selection controller. If the specified target is in the list of candidate targets, it gets the highest priority as a target. However, if the specified target is not visible as a target to the targeting entity, the entity fires at the other available targets. It does not keep looking for the specified target.

To set an entity's target:

1. Select the entity.
2. Choose **Set** → **Engagement** → **Target**. The Target dialog box opens.
3. Optionally, filter the entity selection list.
4. In the Target list, or on the terrain, select the entity to target.
5. Click OK.

34.66. Tasked by Superior

You can order a simulation object that is part of a unit, and which is executing its own plan or is conducting an independent task, to stop executing those tasks and take commands from its unit. When you order a simulation object to be tasked by its superior, it stops conducting independent tasks and waits for the next order from the unit. (It does not execute the unit's current task, if any.) For more information about being tasked by superior, please see "Independently Tasking Unit Members," on page 25-11.

To order an object to be tasked by its superior:

1. Select the simulation object.
2. Choose **Set** → **Other** → **Tasked by Superior**.

34.67. TDL Messaging

The TDL Messaging set data request enables or disables TDL messages for a simulation object or sets it to use the global setting.

To specify TDL messaging:

1. Select the simulation object.
2. Choose **Set** → **Other** → **TDL Messaging**. The TDL Messaging dialog box opens.
3. Select an option from the TDL Messaging Enabled list.
4. Click OK.

34.68. Tracer Use

You can specify use of tracer rounds for entities that have weapons that support them. To see tracers, you must enable them for the observer you are using. They are not supported in 2D observer modes.

To enable or disable user of tracers:

1. Select the entity for which you want to enable or disable use of tracers.
2. Choose **Set** → **Engagement** → **Tracer Use**. The Tracer Use dialog box opens.
3. Choose an option from the Use Tracers list.
4. Click OK.

34.69. Unit State

The Unit State request lets you change an entity-level unit from the aggregated state to the disaggregated state and the reverse.

To set a unit's aggregation state:

1. Select the unit.
2. Choose **Set** → **Unit** → **Unit State**. The Unit State dialog box opens.
3. Select the desired state from the list.
4. Click OK.



35. Setting the State of Aggregate-Level Simulation Objects

This chapter explains how to set simulation object states for units in aggregate-level scenarios. Some of the set data requests also apply to entity-level scenarios. They are cross referenced to Chapter 34, *Setting the State of Entity-Level Simulation Objects*.

Activity 35-3
Ammunition 35-3
Ammunition Pacing/Tracking 35-4
Assign Aircraft 35-4
Editing Aircraft Assignments 35-5
Deleting Aircraft Assignments 35-6
Assume Unknown is Hostile 35-6
Capabilities 35-6
Concealed 35-6
Contamination 35-6
Current Speed 35-7
Destroyed 35-7
Disembarked 35-7
Embarked 35-7
Emitter 35-7
Engagement Zones 35-7
Engagement Results 35-7
Equipment Pacing/Tracking 35-8
Facilities 35-8
Food, Water, Fuel, Oil and Lubricant 35-9
Food, Water Fuel, Oil and Lubricant Pacing/Tracking 35-10

Force 35-10
Has Drag Chute 35-10
Heading 35-10
Health 35-11
IFF 35-11
Loadout 35-11
Location 35-11
MOPP Level 35-12
Morale 35-12
Notify Level 35-12
Ordered Speed 35-12
Percent Complete 35-12
Personnel and Supplies 35-13
Posture (Unit) 35-13
Preferred Targets 35-14
Radar Mode 35-14
Reorganize 35-14
Resources 35-14
Resources Pacing/Tracking 35-14
Restore 35-15
Resupply Mode 35-15
Role 35-15
Rules of Engagement 35-15
Sector of Responsibility 35-15
Sensor Enabled 35-16
Spot Reports 35-16
Superior 35-16
Supplying 35-16
Tasked by Superior 35-16
TDL Messaging 35-16
Weapons Pacing/Tracking 35-17

35.1. Activity

You can give a simulation object an activity. Activities are configured by simulation object type, so they may vary from one simulation object to another. Some of the available activities include the following:

- Unknown
- Attacking
- Bivouacking
- Emplacing obstacle
- Indirect fire
- Jamming
- Other
- Patrolling
- Probing
- Retreating.

To specify an activity:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Activity**. The Activity dialog box opens.
3. Select an activity from the list.
4. Click OK.

35.2. Ammunition

You can specify the amount of each type of ammunition that a simulation object has.

To set the ammunition level:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Ammunition**. The Ammunition dialog box opens.
3. Select the check box for each type of ammunition that you want to set.
4. For each selected type of ammunition, adjust the slider or type a value.
5. Click OK.

35.3. Ammunition Pacing/Tracking

Ammunition Pacing/Tracking lets you specify that a particular ammunition type be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

To specify pacing or tracking for a unit's ammunition:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Ammunition Pacing/Tracking**. The Ammunition Pacing/Tracking dialog box opens.
3. Select the check box for each type of ammunition for which you want to specify pacing or tracking.
4. For each selected ammunition type, select an option from the list.
5. Click OK.

35.4. Assign Aircraft

The Assign Aircraft set data request lets you assign aircraft to an air base, edit existing assignments, and remove aircraft from the air base.

The aircraft assignments process works differently depending on whether you are using it directly from the Set menu or as part of a plan. When you assign aircraft from the Set menu, the Assign Aircraft shows all of the aircraft types currently assigned to the air base and the amount of each type of aircraft. By default (the Set Assigned Aircraft to This List option), when you click OK the new or updated list of assigned aircraft replaces whatever the previous list was. This is the way set data requests typically work.

If you include an Assign Aircraft set data request in a plan, there is no way to know what the state of aircraft assignments will be when the plan statement is executed. Therefore, the default behavior for Assign Aircraft when used in a plan is to add or change only the specific aircraft assignments listed in the Assign Aircraft dialog box, rather than resetting all aircraft assignments (the Add or Change Listed Aircraft option).

The Assign Aircraft dialog box has a set of option buttons whose default settings support the different use cases described in previous paragraphs. However, you can choose the alternative options if you want to.

To add an aircraft type to an air base:

1. Select the airbase.
2. Choose **Set** → **Air Base** → **Assign Aircraft**. The Assign Aircraft dialog box opens (Figure 35-1). If the air base already has aircraft assigned to it, they are listed in the window along with the number of each type assigned. (If you are editing a plan, the current aircraft assignments are not listed.)

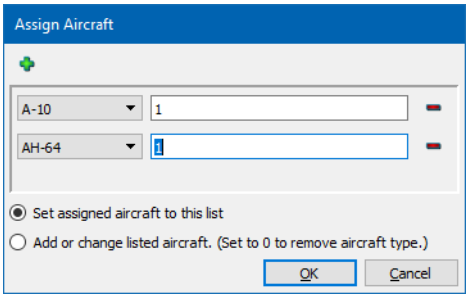


Figure 35-1. Assign Aircraft dialog box

3. For each aircraft type that you want to assign to the air base:
   a. Click the Add button (+). A row is added to the dialog box window.
   b. In the aircraft list, select the type of aircraft you want to add.
   c. In the text box, type the number of this type of aircraft you want to add.
4. Optionally, edit any existing aircraft assignments.
5. Select the appropriate option button. (It is usually best to accept the default option.)
6. Click OK.

35.4.1. Editing Aircraft Assignments

You can add or remove aircraft types for an air base and you can change the number of aircraft of a specific type. When you add aircraft, they are added in the Available state. No fuel, munitions, or air crews are added.

If you reduce the number of aircraft of a specific type, the aircraft are taken from the Available aircraft first, followed by In-Repair and Assigned. If Assigned aircraft are removed, their fuel, munitions, and air crews are returned to the air base.

35.4.2. Deleting Aircraft Assignments

To delete an aircraft assignment for an air base:

1. Select the airbase.
2. Choose **Set** → **Air Base** → **Assign Aircraft**. The Assign Aircraft dialog box opens. If the air base already has aircraft assigned to it, they are listed in the window along with the number of each type assigned. (If you are editing a plan, the current aircraft assignments are not listed.)
3. If you are not editing a plan, click the Delete button next to the row that you want to delete or set the number of aircraft to zero. If you are editing a plan, add the aircraft type that you want to delete and set its amount to 0.
4. Select the appropriate option button. (It is usually best to accept the default option.)
5. Click OK.

35.5. Assume Unknown is Hostile

For details, please see "Assume Unknown is Hostile," on page 34-9.

35.6. Capabilities

For details, please see "Capabilities," on page 34-10.

35.7. Concealed

For details, please see "Concealed," on page 34-12.

35.8. Contamination

The Contamination set data request lets you set the contamination type for a simulation object.

To set contamination:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Contamination**. The Contamination dialog box opens.
3. Select a contamination type from the list.
4. Click OK.

35.9. Current Speed

For details, please see "Current Speed," on page 34-13.

35.10. Destroyed

For details, please see "Destroyed," on page 34-13.

35.11. Disembarked

For details, please see "Disembarked," on page 34-16.

35.12. Embarked

For details, please see "Embarked," on page 34-16.

35.13. Emitter

For details, please see "Emitter," on page 34-17.

35.14. Engagement Zones

For details, please see "Engagement Zones," on page 34-18.

35.15. Engagement Results

An engagement result is a a text string that gets included in entity state update messages that get sent over the network. The purpose is for a human participant to comment on the current state of a simulation object as a simulation progresses. Once specified, the text string continues to get sent until the expiration time is reached or a new engagement result is set.

To set engagement results:

1. Select the simulation object.
2. Choose **Set** → **Engagement** → **Engagement Results**. The Engagement Results dialog box opens.
3. Specify the engagement result. This can be any text you want to enter.
4. Set the expiration time in days and hours:minutes:seconds.
5. Click OK.

35.16. Equipment Pacing/Tracking

Equipment Pacing/Tracking lets you specify that a particular type of equipment be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

To specify pacing or tracking for a unit's equipment:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Equipment Pacing/Tracking**. The Equipment Pacing/Tracking dialog box opens.
3. Select the check box for each type of equipment for which you want to specify pacing or tracking.
4. For each selected ammunition type, select an option from the list.
5. Click OK.

35.17. Facilities

The Facilities set data request lets you specify all of the capabilities of an air base, as follows:

- Runways. Specify a name, length, width, and main direction for each runway.
- Has Night Support. Able or not able to launch missions at night.
- ORP Aircraft Capacity. The number of units that can be in operational readiness platform parking spaces.
- Apron Area. The size of the apron. VR-Forces calculates the number of aircraft that can be on the apron based on their footprints.
- Repair Hangar Aircraft Capacity. The number of aircraft that can be in the repair hangar at one time.
- Aviation Fuel Capacity. The amount of aviation fuel the airbase can store.
- Munition Storage Capacity. The amount of munitions the air base can store.
- Barracks Capacity. The number of personnel that can be on the airbase and in the barracks.
- Is Apron Concealed.
- Is Hangar Concealed.

To specify air base facilities:

1. Select an air base.
2. Choose **Set** → **Air Base** → **Facilities**. The Facilities dialog box opens.
3. To add a runway:
   a. Click the Add button (+). A new runway is added to the runway list.
   b. Type values in each column of the runway description.
4. To delete a runway:
   a. Put the cursor in a cell in a runway row.
   b. Click the Delete button (−).
5. To enable night support, select Yes in the list.
6. For the storage capacities, enter the amount of that item that you want to store in the air base.
7. To revert any settings or the configured runways to their original values, click the Revert button (↺) for that setting.
8. To set the concealment status of the apron or hangar, select an option in the appropriate list.
9. Click OK.

35.18. Food, Water, Fuel, Oil and Lubricant

You can set the resources for a simulation object.

To set the level of food, water, fuels, and lubricants:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Food, Water, Fuel, Oil and Lubricant**. The Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant dialog box opens.
3. Select the check box for each type of resource you want to specify.
4. For each selected resource type a value.
5. Click OK.

35.19. Food, Water Fuel, Oil and Lubricant Pacing/Tracking

Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking lets you specify that a particular resource be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

To set pacing and tracking for food, water, fuel, and lubricants:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Food, Water, Fuel, Oil and Lubricant Pacing/Tracking**. The Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant Pacing/Tracking dialog box opens.
3. Select the check box for each type of resource for which you want to specify pacing or tracking.
4. For each selected resource select an option from the list.
5. Click OK.

35.20. Force

For details, please see "Force," on page 34-18.

35.21. Has Drag Chute

The Has Drag Chute set data request specifies whether or not the aircraft in an air unit have a drag chute.

To set a drag chute:

1. Select an aircraft unit.
2. Choose **Set** → **Disposition** → **Has Drag Chute**. The Has Drag Chute dialog box opens.
3. Select an option from the Has Drag Chute list.
4. Click OK.

35.22. Heading

For details, please see "Heading," on page 34-20.

35.23. Health

Health is a value based on a simulation object's total personnel, equipment, and weapons. The Health set data request lets you restore personnel to their full values and set the overall health of the simulation object. For details about simulation object health, please see "The Aggregate Warfare Model," on page 22-3.

To set an object's health:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Health**. The Health dialog box opens. It displays the current state of personnel, equipment, and weapons.
3. To restore all personnel, select Reset Kill, Captured, Wounded, Missing.
4. To specify the overall health of the simulation object, adjust the Health slider or type a number in the Health box. As you move the slider to the left, the amount of personnel, equipment, and weapons decreases.
5. Click OK.

35.24. IFF

For details, please see "IFF," on page 34-21.

35.25. Loadout

The Loadout set data request lets you specify the loadout for an air unit. It lists the contents of the selected loadout.

To set the loadout for an air unit:

1. Select an air unit.
2. Choose **Set** → **Disposition** → **Loadout**. The Loadout dialog box opens.
3. Select an option from the Loadout Name list.
4. Click OK.

35.26. Location

For details, please see "Location," on page 34-26.

35.27. MOPP Level

When you set a unit's MOPP level, the MOPP level changes instantly. To change MOPP level over time, use the Change MOPP Level task.

To set the MOPP level:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **MOPP Level**. The MOPP Level dialog box opens.
3. Select an option from the list.
4. Click OK.

35.28. Morale

To set a unit's morale:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Morale**. The Morale dialog box opens.
3. Adjust the slider or type a value from 0 through 100.
4. Click OK.

35.29. Notify Level

For details, please see "Notify Level," on page 34-28.

35.30. Ordered Speed

For details, please see "Ordered Speed," on page 34-30.

35.31. Percent Complete

Percent Complete lets you set the degree of completion for combat engineering objects.

To set the percent complete:

1. Select a combat engineering object.
2. Choose **Set** → **Disposition** → **Percent Complete**. The Percent Complete dialog box opens.
3. Adjust the slider or type a value in the box.
4. Click OK.

35.32. Personnel and Supplies

The Personnel and Supplies set data request lets you specify the number of air crews and ground crew available to an air base. You can also set the amount of munitions and aviation fuel. The total of air crews and ground crew cannot exceed the barracks capacity (set using the Facilities set data request). The maximum amount of munitions and aviation fuel that an air base can have is set with the Facilities set data request.

To specify personnel and supplies for an air base:

1. Select an air base.
2. Choose **Set** → **Air Base** → **Personnel**. The Personnel dialog box opens.
3. Optionally, type a value for Ground Crew Available.
4. Optionally, type a value for Air Crew Available.
5. Optionally, type a value for the amount of munitions.
6. Optionally, type a value for the amount of aviation fuel.
7. To revert any setting to its original value, click the Revert button () for that setting.
8. Click OK.

35.33. Posture (Unit)

You can set the posture of a unit. The posture changes instantly. To change posture over time, use the Change Posture task.

To set an object's posture in an aggregate-level scenario:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Posture**. The Posture dialog box opens.
3. Select an option from the list.
4. Click OK.

35.34. Preferred Targets

Units can target up to three simulation objects. By default, they target the three simulation objects that are most vulnerable to attack. If you want a simulation object to target specific simulation objects that it might not choose automatically, the Preferred Targets set data request lets you specify them as preferred targets.

To specify preferred targets:

1. Select a unit.
2. Choose **Set** → **Engagement** → **Preferred Targets**. The Preferred Targets dialog box opens.
3. Select up to three simulation objects.
4. Click OK.

35.35. Radar Mode

For details, please see "Radar Mode," on page 34-32.

35.36. Reorganize

For details, please see "Reorganize," on page 34-33.

The change takes place immediately. For more information about reorganization, please see "Reorganizing Units," on page 21-4.

35.37. Resources

For details, please see "Resources," on page 34-34.

35.38. Resources Pacing/Tracking

Resources Pacing/Tracking lets you specify that a resource type be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

To set pacing and tracking for a resource:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Resources Pacing/Tracking**. The Resources Pacing/Tracking dialog box opens.
3. Select Resources.
4. Click OK.

35.39. Restore

For details, please see "Restore," on page 34-34.

35.40. Resupply Mode

Resupply Mode specifies one of the following resupply modes for simulation objects:

- None. The simulation object does not resupply itself.
- From Supply Units. When a resupply unit is in range, the unit gets supplies from it.
- From Self-Continuous. The unit resupplies itself continuously based on its resupply rate parameters. For details, please see "Logistics," on page 22-15.
- From Self-Periodic. The unit periodically resets all of its supplies, except health, to the base level. This level is reduced by health losses.

To specify the resupply mode:

1. Select the unit.
2. Choose **Set** → **Logistics** → **Resupply Mode**. The Resupply Mode dialog box opens.
3. Select a resupply mode from the Resupply Mode list.
4. If you select From Self-Periodic, specify the resupply period in the Periodic Resupply Period boxes.
5. Click OK.

35.41. Role

The Role set data request lets you specify the role of an air unit, such as fighter, attack, bomber, or QRA.

To set the role of an air unit:

1. Select an air unit.
2. Choose **Set** → **Disposition** → **Role**. The Role dialog box opens.
3. Select an option in the Role list.
4. Click OK.

35.42. Rules of Engagement

For details, please see "Rules of Engagement," on page 34-35.

35.43. Sector of Responsibility

For details, please see "Sector of Responsibility," on page 34-37.

35.44. Sensor Enabled

For details, please see "Sensor Enabled," on page 34-38.

35.45. Spot Reports

For details, please see "Spot Reports," on page 34-41.

35.46. Superior

For details, please see "Superior," on page 34-42.

35.47. Supplying

The Supplying set data request sets a supply unit to stop providing supplies or to start providing supplies.

To stop or start a supply unit providing supplies:

1. Choose **Set** → **Logistics** → **Supplying**. The Supplying dialog box opens.
2. Select Stop or Start.
3. Click OK.

35.48. Tasked by Superior

For details, please see "Tasked by Superior," on page 34-44.

35.49. TDL Messaging

For details, please see "TDL Messaging," on page 34-44.

35.50. Weapons Pacing/Tracking

Weapons Pacing/Tracking lets you specify that a weapon be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

To set pacing or tracking for a unit's weapons:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Weapons Pacing/Tracking**. The Weapons Pacing/Tracking dialog box opens.
3. Select the check box for each weapon for which you want to specify pacing or tracking.
4. For each selected weapon type, select an option from the list.
5. Click OK.



VII. Plans

This section describes how to write plans.

VII.1. Introduction to Plans

VR-Forces has two kinds of plans, global plans and individual plans. Both kinds of plans can assign tasks and set data requests to a simulation object. Plans can also contain commands that create or delete simulation objects and tactical graphics. The commands in a plan can be conditional, that is, they will be executed only if a condition is satisfied. When you open a scenario, it is in pause mode (unless you start it with the `-l` command line option). When you start the simulation, the plans start executing.

An individual plan is a set of statements that order a particular simulation object to complete a sequence of tasks, change its state, or both. The simulation object carries out the tasks in order unless it is interrupted by an independent task or global command.

A simulation object owns its individual plan. Changing its name, echelon relationship, or force does not change its plan.

Global plans are created and executed independently of any particular simulation object. Commands sent to simulation objects from a global plan affect them as if they received an independent task through the Task menu — they interrupt the current task, if any, and cause the simulation object to abandon its plan, if it has one.

If a simulation object does not have a plan, it does not mean that it has nothing to do. A simulation object may be assigned tasks by its superior in the force hierarchy. It can also be assigned tasks interactively from the front-end. If you want a simulation object to take planned actions independently of those assigned by its superior, you need to add statements to its plan.

You can view a simulation object's plan at any time in the Plan window. The task that a simulation object is currently executing is highlighted. A simple plan might look like the following:

```
Move-To Waypoint:"red point"
Wait-Duration Seconds-ToWait:900
Move-To Waypoint:"white point"
Set Engagement-Rules="Fire-at-will"
```


You can edit plans while a simulation is running. However, when you change a plan during a simulation, the simulation object starts executing the plan from the beginning, which may cause its activities to be out of synchronization with the other simulation objects in the simulation.



36. Plan Concepts

This chapter describes plans and plan concepts.

Introduction to Individual Plans and Global Plans 36-2
Conditional Statements 36-3
- The If Statement 36-5
- Triggers 36-6
- While Statements 36-8
- Wait Until Statements 36-8
- Conditional Tests 36-9

Plan Variables 36-14
Viewing Plans 36-15
Considerations for Creating Plans 36-16
- Considerations for Using Triggers 36-16
- Using Concurrent Tasks in Plans 36-17
- Moving In Formation 36-17
- Using the Tasked-By-Superior Request in a Plan 36-17
- Following Simulation Objects 36-17
- Using Non-VR-Forces Simulation Objects in Plans 36-18

36.1. Introduction to Individual Plans and Global Plans

A plan contains one or more statements that get executed by VR-Forces without human intervention. You build these statements in dialog boxes. VR-Forces writes the statement to the plan, so you do not have to worry about syntax errors in your statements when you first create them. However, you must understand the meaning of individual commands and parameters and the implications of how you order them, to make sure that a plan does what you want it to do.

VR-Forces supports two types of plans – individual plans and global plans. They have the following characteristics:

- Individual plans apply to a specific simulation object. They contain tasks and set data requests that are specific to the simulation object and which get executed sequentially. The sequence of task execution can be interrupted by receipt of a task from an external actor (a global plan command or independent task). The task sequence can also be interrupted if a trigger (a form of conditional statement) or a reactive task is activated.

! Individual plans can include global commands (which are described later in this section). However, you must be careful not to use global commands, tasks, or sets for a simulation object in its own plan when you really want simulation object-specific tasks or sets. Sending a simulation object a global command, even from within its own plan, terminates its plan.

- Global plans are independent of any simulation object. The statements in a global plan affect the scenario as a whole (create and delete object commands) or a specific simulation object (task and set commands). Tasks sent to a simulation object from a global plan have the same effect on the simulation object as independent tasks assigned by a human player. They interrupt any task that the simulation object is engaged in and terminate its plan. Furthermore, if a global plan has a series of tasks for the same simulation object, they get sent in order without regard to completion of the previous task (as if you issued a series of commands from the Task menu, one immediately after another). In such a case, only the last command sent is likely to be completed.

Figure 36-1 shows a global plan and an individual plan. Note that the individual plan window (on the right) is tied to a particular simulation object (shown in the title bar).

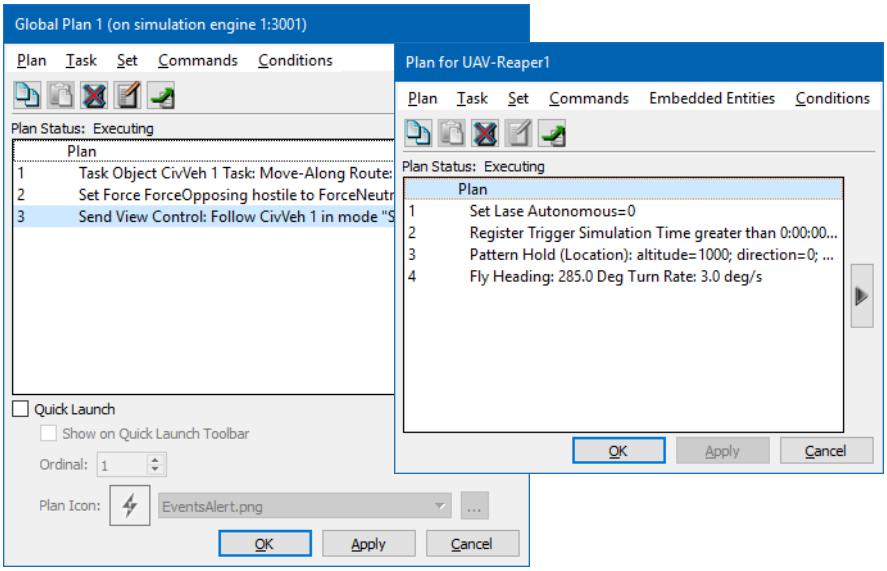


Figure 36-1. Global plan and individual plan

36.2. Conditional Statements

Many of the statements used in plans set a parameter or direct a simulation object to perform a specific task unconditionally. However, the plan language also includes conditional statements. A conditional statement specifies a condition and a block of plan statements that get executed if the condition is true. VR-Forces supports the following conditional statements:

- `If`. An `If` statement is evaluated once, when the plan execution flow arrives at the statement. If the condition is true at that point, VR-Forces executes the statements in the `If` block. For details, please see "The If Statement," on page 36-5.
- Trigger. A trigger statement lets you plan for a condition without knowing in advance when the condition will occur. It specifies a condition that is evaluated continuously by VR-Forces. If the condition becomes true, VR-Forces executes the statements in the trigger. For details, please see "Triggers," on page 36-6.
- `While`. A `While` statement executes a block of statements repeatedly while a condition remains true. VR-Forces evaluates the condition when the plan first arrives at the `While` statement. If the condition is true, it executes the statements in the block. When it completes the statements, it evaluates the condition again. If it is still true, VR-Forces executes the statements again. It continues to do so as long as the `While` condition is true. For details, please see "While Statements," on page 36-8.
- `Wait Until`. A `Wait Until` statement causes a simulation object to wait in place until the condition becomes true. Then the simulation object moves to the next statement in the plan. For details, please see "Wait Until Statements," on page 36-8.

The conditions that can be tested by conditional statements are:

- Is a simulation object in a given area? (Entity In Area)
- Is a simulation object to the left of a phase line? (Entity Left of Phase Line)
- Has a specific entity been created? (Entity Created)
- Is a simulation object destroyed? (Entity Destroyed)
- Is a simulation object under fire? (Entity Under Fire)
- Has a simulation object located a target? (Entity Has Target)
- Has a simulation object's embarkation state changed? (Entity Embarked)
- Is a simulation object above or below a specified altitude? (Entity Altitude)
- Has this simulation object detected another simulation object? (Detect Entity)
- Has this simulation object received a text message? (Receive Text Message)
- Has an engineering object been breached? (Engineering Object Breached)
- Has this simulation object received a missile approach warning? (Missile Approach Warning)
- Is a scenario event in process or concluded? (Scenario Event)
- Has a lifeform surrendered? (Lifeform Surrendered)
- Does a lifeform have a particular DI-Guy animation? (Entity Di-Guy Animation Check)
- Does a lifeform have a particular DI-Guy appearance? (Entity Di-Guy Appearance Check)
- Is a random probability true? (Random)
- Is a resource for this simulation object at a certain level? (Resource)
- Is the simulation time greater than or less than a specific elapsed time? (Simulation Time)
- Is a subordinate of a unit independently tasks? (Simulation Object Independently Tasked)
- Is the simulation date and time a certain value? (Simulation Date/Time)
- Is a tactical graphic active? (Tactical Graphic Active)

Conditional statements can test for True and False, and can include the logical operators AND, OR, and NOT.

For details, please see "Conditional Tests," on page 36-9.

36.2.1. The If Statement

An `If` statement tests a condition. It only tests the condition once.



Do not use an `If` statement to test for receipt of a message or some other circumstance that does not represent a simulation object's state. VR-Forces does not keep a record of events and such a test will almost always evaluate to false. Use a trigger to test for conditions that do not evaluate simulation object state.

An `If` block consists of:

- The word `If`.
- A condition to be evaluated.
- A block of statements.
- An `else` statement.
- Optionally, statements for the `else` condition.
- An `endif` statement.

If the condition is true at the moment the `If` statement is evaluated, control passes to the first statement of the `If` block. If the condition is false, control passes to the first statement in the `else` block, if you have one, or to the first statement after the complete `If` block if you do not have an `else` block. You can nest `If` statements within `If` statements. The syntax is as follows:

```
If (condition) then
   zero_one_or_more_statements
else
   zero_one_or_more_statements
endif
```

When VR-Forces reaches the end of the block of `If` or `else` statements, control passes to the first statement after the `If` statement.

The Add Conditional Expression dialog box ensures that you have the correct syntax. It does not ensure that your `If` block makes logical sense.

36.2.2. Triggers

A trigger consists of a condition and a block of statements that are executed when the condition becomes true.

When you create a trigger, it is not automatically added to the plan's execution sequence. You must register a trigger before VR-Forces will test it. (If you have a programming background, it may be helpful to think of triggers as functions or subroutines that you write separately and then call from elsewhere in a plan. Extending the analogy, a trigger is always local to its plan.)

When you register a trigger, VR-Forces checks the condition. If the condition is true, control passes to the first statement in the trigger body. If the condition is false, VR-Forces continues to the next statement in the plan. It continues to check the trigger condition periodically. If at any point, the condition evaluates as true, control passes to the block of statements associated with the trigger.

When you create a trigger, you can specify that it interrupt the current task or that it suspend and resume the current task.

If you specify that a trigger interrupt the current task, task execution proceeds as follows:

- If a trigger contains any tasks that are mutually exclusive with the currently executing task, the current task is discarded and the statements in the trigger are executed.
- If the tasks in a trigger are not mutually exclusive with the current task, or if the trigger only has set data request statements, the trigger's statements are executed and the current task continues. For more information about this case, please see "Using Triggers That Do Not Have Mutually Exclusive Tasks," on page 36-16.
- After the tasks in the trigger block are completed, control returns to the next statement in the simulation object's plan after the one that was interrupted by the trigger.

If you specify that a trigger suspend the current task, task execution proceeds as follows:

- If a trigger contains any tasks that are mutually exclusive with the currently executing task, the current task is suspended and the statements in the trigger are executed.
- If the tasks in a trigger are not mutually exclusive with the current task, or if the trigger only has set data request statements, the trigger's statements are executed and the current task continues. For more information about this case, please see "Using Triggers That Do Not Have Mutually Exclusive Tasks," on page 36-16.
- After the tasks in the trigger block are completed, the suspended task is resumed, if possible. There may be cases where the previous task is no longer meaningful due to changes in the scenario.

The location of a trigger in a plan has an important affect on its functioning. For a discussion about where to place triggers, please see "Deciding Where to Put Triggers," on page 36-16.)

By default, the name of a trigger is the syntax of its condition. However, you can change the name of a trigger to be any text you want to improve readability or shorten it.

Figure 36-2 illustrates the process when a trigger has a task that is mutually exclusive with the current task and you have specified that the current task be discarded. After the tasks in the trigger are completed, control returns to the next statement in the simulation object's plan. If the simulation object was in a `While` loop when the trigger fired, it returns to the next statement in the `While` block.

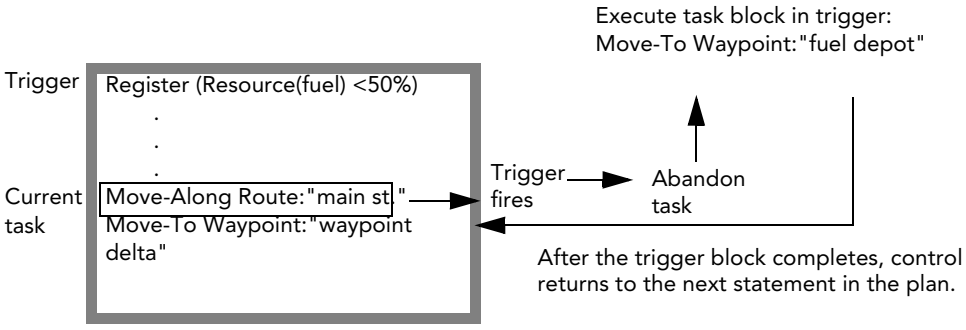


Figure 36-2. Flow of control for triggers

By default, after a trigger fires, it is removed from the list of registered triggers. If you want to continue to test for the same condition, you must reregister the trigger. You can design the plan in such a way that the trigger is registered again. Alternatively, when you create a trigger, you can specify that it reregister automatically.

An alternative to using triggers in a plan is to write a reactive task. For information about reactive tasks, please see Chapter 31, *Creating Scripted Tasks and Sets*.

36.2.3. While Statements

A `While` statement consists of:

- The word `While`.
- A condition to be evaluated.
- A block of statements.
- An `endWhile` statement.

If the condition is true at the moment the `While` statement is reached, control passes to the first statement of the `While` block. If the condition is false, control passes to the first statement after the complete `While` block. You can nest `While` statements within other conditional blocks.

When VR-Forces reaches the end of the `While` block, it evaluates the condition again. If the condition is still true, VR-Forces executes the statements again. If it is false, control passes to the first statement after the `While` block.



- VR-Forces re-evaluates a `While` condition only after it has executed all of the statements in the block. If the condition becomes false while VR-Forces is executing statements in the block, VR-Forces continues to execute all statements because it does not re-evaluate the condition until it completes the `While` block.
- If you put a continuous task such as Wait or Patrol Between in a `While` block, it is possible that the condition will never be re-evaluated, because these tasks never finish.

36.2.4. Wait Until Statements

A `Wait Until` statement consists of:

- The words `Wait Until`.
- A condition to be evaluated.

If the condition is true at the moment the `Wait Until` statement is reached, the simulation object continues to the next task in the plan. If the condition is false, the simulation object waits in place. VR-Forces continues to check the condition periodically until the condition becomes true or the simulation object is tasked by some other form of input, such as a trigger or independent task.

If a `Wait Until` state is interrupted by a trigger and the trigger's Interrupt Current Task check box is not selected, the plan returns to the `Wait Until` state after the trigger's tasks are completed. If the trigger's Interrupt Current Task check box is selected, the plan moves to the statement after the `Wait Until`.

If the condition becomes true, control passes to the next statement in the plan.

36.2.5. Conditional Tests

This section describes each of the conditions that VR-Forces can test. When you set up conditions, keep in mind the following:

- If there is no simulation object with the specified name, the result is always false.
- If the specified simulation object is destroyed, the condition might never be satisfied. Your plan should take this possibility into account.

Altitude

Altitude returns true if the simulation object (or member of a unit) is at a higher or lower altitude than the one specified.

Detect Entity

The Detect Entity condition tests to see if this simulation object has detected another simulation object at a particular identification level. The identification levels are Detected, Classified, Identified, and Full Knowledge. Identification levels are determined by the detection probability configuration files, based on the distance between the detecting simulation object and the target and for how long the simulation object has been detected. For details about the detection probability files, please see "Detection Tables," on page 75-16.



This condition is not supported in global plans.

Engineering Object Breached

The Engineering Object Breached condition returns true if the specified engineering object has been breached. (Aggregate-level scenarios only.)

Entity Destroyed

Entity Destroyed returns true if the specified simulation object is destroyed.

The definition of destruction for a unit is implementation specific. For the entity-level models provided with VR-Forces, disaggregated units are considered to be destroyed when all members of the unit are destroyed.

If you have implemented units locally using the VR-Forces Toolkit, they could have a different definition of destroyed.

Entity Di-Guy Animation Check

The Entity Di-Guy Animation Check condition tests the current DI-Guy animation for a lifeform. This condition may be useful if you are scripting a series of animations. For example, if a particular entity is using a threatening or hostile animation, you might want to have other simulation objects respond a certain way.

Entity Di-Guy Appearance Check

The Entity Di-Guy Appearance Check condition tests the current DI-Guy appearance for a lifeform. This condition may be useful if you are scripting a lifeform's interaction with another lifeform based on ethnicity or how they are dressed.

Entity Embarked

Entity Embarked returns true if the named simulation object is embarked.

Entity Has Target

Entity Has Target returns true if the simulation object specified in the condition identifies a target. For example, consider the entities in Figure 36-3. The condition in M1A2 1's plan is true when BMP 1 has a target, not when M1A2 1 targets BMP 1.

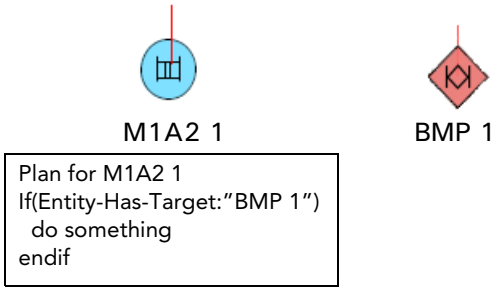


Figure 36-3. Entity Has Target condition

If a simulation object's rules of engagement prevent it from firing, this condition does not return true when the simulation object identifies a target.

Entity In Area

Entity In Area returns true if the simulation object reports that it is in the named area. You can use the `NOT` operator, to test whether a simulation object is outside the area.

If a simulation object enters an extruded area, VR-Forces does a 2D check first, to see if it is in the area. Then it checks the simulation object's altitude to see if it is within the area's volume.

If you have a plan in which you are testing for entities in an area and destroying them as they enter it, the Exclude Destroyed Simulation Objects option lets you reregister a trigger and only test for entities that have not been destroyed.

Entity Left of Line

Entity Left of Line returns true if the simulation object reports that it is to the left of the specified phase line.

A simulation object is considered to be to the left of a phase line if it is on the left side of an infinite extension of the phase line as viewed from a point on the line facing in the direction of the line. For a description of phase line direction, please see "Phase Lines," on page 38-4.

Entity Under Fire

Entity Under Fire returns true if the simulation object is under fire. An entity is considered to be under fire if it is directly targeted in a Fire PDU or if there is a hostile detonation within the **under-fire-parameter** distance specified in the `under-fire-determination-sensor` sensor in its OPE file. The default for all entities is 1000 meters.



Lifeform Surrendered

The Lifeform Surrendered condition tests to see if a lifeform entity has surrendered (using the Surrendered set data request).

Missile Approach Warning

The Missile Approach Warning condition returns true if a hostile missile is within the specified distance of this simulation object. This condition is not available in global plans.

Random

You can use the Random condition to randomly decide a simulation object's behavior at run time.

Percentage – A floating point probability percentage value between 0% (zero) and 100%. Random returns true if a random number between zero and 100 is less than or equal to the specified value.

If you use the Random condition, consider the following issues:

- Random is not very useful in triggers, since triggers may evaluate their condition expressions as often as once per tick. This means that a trigger expression of "Random Probability:50%" is likely to fire within the first couple of times it is evaluated. Therefore, we recommend that you not create triggers that are based on a random condition.
- If you use a random condition in a cascading sequence of `If` statements, and you want to ensure the correct probability distribution, be careful how you specify the percentage value. For example, randomly selecting one of three paths with equal probability would be done like this:

```
If (Random Probability:33.33%) then
  Move-Along Route:"1"
Else
  If (Random Probability:50%) then
    Move-Along Route:"2"
  Else
    Move-Along Route:"3"
  Endif
Endif
```

Resource

The Resource condition tests the amount of a specified resource for this simulation object using comparison operators. This condition is not available in global plans.

Receive Text Message

The Receive Text Message condition tests to see if this simulation object has received a text message that contains the text specified in the Receive Text Message dialog box. This condition is not available in global plans. The text message can be received as the result of a Send Text Message task or the Send Text Message command in a plan.

Scenario Event

The Scenario Event condition returns true if the specified scenario event is in process or has completed.

Simulation Object Independently Tasked

The Simulation Object Independently Tasked condition tests to see if an object that is a member of a unit is executing a task independently of the unit. If a unit member is independently tasked, you might not want to issue a new task to the unit until the subordinate finishes its task. This condition lets you test the independent task status of all members of a unit or particular members of a unit to help plan the timing of unit tasks. For example in the Parachuting example scenario, the plan for the helicopter CH 47 1 waits until all members of Inf SQD 1 are not independently tasked before it sends a message to the squad.

Simulation Time

The Sim Time condition returns true if the elapsed simulation time is greater than (Sim Time >) or less than (Sim Time <) the time specified in the statement. Sim Time is the elapsed simulation time in days, hours, minutes, and seconds.

Simulation Date/Time

The Simulation Date/Time condition returns true if the simulation date and time matches the specified value.

Tactical Graphic Active

The Tactical Graphic Active condition returns true if a published tactical graphic is in the active state. For details about setting a tactical graphic's state, please see "Active and Inactive Tactical Graphics," on page 40-24 and "Activation," on page 41-2.

Boolean Operators

You can use the following boolean operators in conditional expressions:

- `True`.
- `False`.
- `AND (`*`expression, expression`*`)` – evaluates as true if both conditional expressions are true.
- `OR (`*`expression, expression`*`)` – evaluates as true if either of the expressions are true.
- `NOT (`*`expression`*`)` - evaluates as true if the conditional expression is false.

36.3. Plan Variables

Many tasks require you to specify a simulation object to act on, such as Fire on Target. Conditional tests, such as Entity Destroyed, require you to identify a simulation object to evaluate. However, you will not always know the name of the simulation object that you need to specify. Variables let you specify a simulation object without knowing its name. Plans can reference two variables:

- Simulation Object. The Simulation Object variable references a simulation object that has been identified by a conditional test, such as Entity in Area or Entity Left of Line. For example, suppose that if an opposing force entity enters an area you want to fire at it. You can create a trigger using an Entity in Area condition. Then in the body of the trigger, you can insert a Fire at Target task that specifies the Simulation Object variable rather than the name of a specific entity.

  In an AND condition, the variable is set to the last condition satisfied. In an OR condition, the variable is set to the first condition satisfied.

- Created Object. The Created Object variable references the most recently created object. For example suppose you create an object from a plan and do not give it a name. You can issue it a plan or refer to it using the Create Object variable. This variable refers to the most recently created object.

Plan variables are specific to each simulation object plan. A variable value in one plan does not affect the variable value in a different plan. Therefore, if you want to use the Created Object variable, you must create the object in the plan in which you want to use it. You could not, for example, create an object in a global plan and then reference it using the Created Object variable from a simulation object's plan.

Once a variable has a value, you can use that variable in a plan as many times as you want to. However, you will need to keep track of plan logic that might change the value as the simulation object progresses through the plan.

For details about how to use variables in plans, please see "Using Plan Variables," on page 37-26.

36.4. Viewing Plans

You can view a simulation object's plan and observe its progress through the plan. Statements are numbered to show the order in which they will execute unless interrupted by a trigger, independent task, or other external input. If a plan is executing the statements in a conditional block, VR-Forces draws a box around the block and the numbering is shown for the statements in the block.

If the Plan window is collapsed (that is, the Trigger pane is not visible) and control passes to a trigger, the window automatically expands to show the tasks in the trigger that is being executed.

The task that a simulation object is executing is highlighted in the Plan window. When the simulation object completes its plan, a message is displayed in the status area at the bottom of the Plan window.

You can view the plans for as many simulation objects at a time as you want. In addition to viewing plan status in the Plan window, you can view the current task on the Task Status page of the Information dialog box. The Last Selected Object Panel also lists the currently executing task.

To view a simulation object's plan:

1. Select the simulation object whose plan you want to view.
2. Open the Plan window using one of the following methods:
   - Choose **Objects** → **Plan**.
   - Press **p**.
   - On the Last Selected Object Panel, click the Plan button ().

36.5. Considerations for Creating Plans

This section has tips for constructing plans including how to avoid problems that are hard to debug and how statements affect each other, with implications for how to organize the statements in a plan.

36.5.1. Considerations for Using Triggers

Triggers are a versatile part of plans and you should have a good understanding of their intricacies before you use them.

Deciding Where to Put Triggers

VR-Forces does not evaluate a trigger condition until it registers the trigger. If you want to ensure that a trigger always gets evaluated by VR-Forces, register it at the beginning of a plan before any non-trigger statements. If you register a trigger later in a plan, it is possible that it will never be evaluated. This could occur if a plan is abandoned. Of course, you might decide that you do not want a trigger to be evaluated unless a set of preceding tasks have been completed. In that case, placing it later in the plan may be exactly what you want to do.

You can also register a trigger inside a conditional block so that it gets registered only if a particular condition is true at a particular time.

For example, if you always want entity 1 to respond to entry of hostile forces into an area, put a trigger at the beginning of the plan. If you want entity 1 to respond to a hostile simulation object in an area only after the simulation has progressed for 15 minutes, nest the trigger inside another trigger that fires when Sim Time > 15 minutes.

Using Triggers That Do Not Have Mutually Exclusive Tasks

A trigger does not have to have any task statements in its statement block. It could just have Set statements. Or, it could have sets and concurrent tasks. If a trigger block just has Set statements, it does not interrupt the task that the simulation object is executing when the trigger gets executed. The Set statements get executed while the simulation object is performing its task. This has the same effect as if you manually set the simulation object's state while it was performing a task.

Using a trigger that just has Set statements can be useful if you want to change a simulation object's state under certain conditions, but do not know in advance when those conditions might exist.

36.5.2. Using Concurrent Tasks in Plans

If you are assigning tasks to simulation objects manually, you can give a simulation object a concurrent task while it is executing some other task. You cannot do this directly in a plan, because you cannot assign two tasks at the same time. A plan always waits for its current task to complete before it starts the next task, even if the next task does not conflict with the current one. If you want to assign a concurrent task in a plan, you must use a trigger. If a trigger that just has a concurrent task fires, the simulation object executes the concurrent task and continues its existing task. If the trigger has other tasks, the behavior will be as expected for triggers.

For more information about concurrent tasks, please see "Concurrent Task Execution," on page 25-4.

As an alternative to using plan triggers to automatically assign concurrent tasks, you can create reactive tasks. For information about reactive tasks, please see "Reactive Tasks," on page 25-11.

36.5.3. Moving In Formation

If you want a unit to move in a certain formation, you can set the formation (Formation set data request) or you can have the unit move into a formation (Move Into Formation task). The default formation is a column.

36.5.4. Using the Tasked-By-Superior Request in a Plan

If you put the Tasked-By-Superior request in the body of a plan, it has no real effect. This is because, as soon as it is set, the plan goes to the next task and overrides the superior. The only way that the Tasked-By-Superior command gets implemented as part of a plan is if it is the last statement in a plan or the last statement in a trigger that happens to get executed after all the statements in a simulation object's plan have been executed.

36.5.5. Following Simulation Objects

If you tell two or more simulation objects to follow another simulation object, do not give them the same offset values. If the followed simulation object stops, one follower will arrive at the specified offset position and stop, while the others will circle the location because they cannot all occupy the same space.

36.5.6. Using Non-VR-Forces Simulation Objects in Plans

You can use a non-VR-Forces simulation object as a parameter in a plan, for example, following a remote simulation object, or testing to see if a remote simulation object is in an area.

To identify non-VR-Forces entities, VR-Forces uses their marking text. If a non-VR-Forces simulation object does not have marking text, VR-Forces uses its simulation object identifier as its name.

VR-Forces does not recognize remote units as units, so it cannot check their children. Remote units are ignored by the simulation engine.



- Referring to remote entities by their marking text may have unpredictable results if there is more than one remote simulation object with the same marking text.
- If you include a non-VR-Forces simulation object in a plan you must ensure that the remote simulator uses the same name for the simulation object in future runs.



37. Writing Plans

This chapter explains how to combine tasks, set data requests, global commands, and conditions into plans.

Writing Plans for Simulation Objects 37-3
- Adding a Task or Set Data Request to a Plan 37-4
- Adding an If Block to a Plan 37-5
- Adding a While Block to a Plan 37-6
- Adding a Wait Until Statement to a Plan 37-6
- Adding a Trigger to a Plan 37-7
- Registering Triggers 37-9
- Cleaning Up Unused Triggers 37-10
- Editing a Statement 37-11
- Deleting Statements from a Plan 37-12

Printing Plans 37-12

Saving Changes to a Plan 37-12

Building Condition Statements 37-13
- Building a Condition Statement that Has One Condition 37-13
- Building a Complex Condition 37-15
- Editing Condition Statements 37-22
- Specifying Names or Patterns in Condition Statements 37-23

Using Plan Variables 37-26

Creating Global Plans 37-27
- Controlling When Global Plans Run (Quick Launch) 37-29
- Opening an Information Dialog Box for a Global Plan 37-29

Adding Commands to a Plan 37-30
- Adding Tasks and Sets from the Commands Menu 37-31
- Sending Text Messages from the Commands Menu 37-34
- Sending Console Messages 37-35

Creating Objects from Within a Plan 37-35
Deleting Objects from Within a Plan 37-35
Issuing a Plan 37-36
Sending View Control Messages 37-37
Writing Plans for Units 37-41
Writing a Plan for Multiple Simulation Objects 37-41
Writing Plans for Remote Entities 37-41
Copying Plans and Plan Statements 37-42
Restarting a Plan 37-43
Abandoning a Plan 37-44

37.1. Writing Plans for Simulation Objects

This section explains how to write individual plans. For conceptual information about plans, please see "Introduction to Plans," on page 35-1, and the previous sections in this chapter.

When you write a plan, keep the following considerations in mind:

- If you edit a plan during a simulation, when you apply the changes, the simulation object whose plan has been changed begins executing it from the beginning. This might mean its activities are no longer synchronized with those of the other simulation objects, unless the updated plan only includes statements that you want to execute starting at the current point of the simulation. Therefore, we recommend that you edit plans when you first load a scenario, before you start running the simulation.
- If you have two or more Plan windows open at the same time, and you open a dialog box, such as Move To, but do not complete the command, you cannot open that same dialog box from another Plan window until you complete the one that is open.

To edit a simulation object's plan:

1. Select a simulation object.
2. Choose **Objects** → **Plan**, or press **p**. The Plan window opens. The simulation object's plan is displayed.

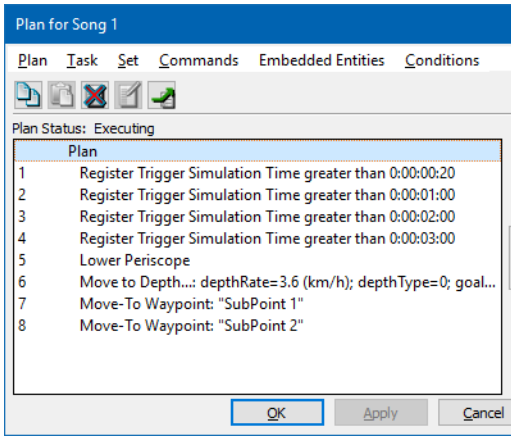


Figure 37-1. Plan window



You can also open the Plan window by clicking the Plan button () on the Last Selected Object Panel.

3. Add new statements, edit statements, or delete statements as described in the following sections.

37.1.1. Adding a Task or Set Data Request to a Plan

When you add statements to a plan, you add them after the currently selected statement.

To add a task to a plan:

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert a new statement. If the plan does not have any statements yet, the plan root – `Plan` is already selected.
3. Choose **Task** → *category* → *Task*, where *category* is one of the categories on the Task menu and *Task* is a task listed for the selected category. A dialog box appropriate to the task opens. The procedures for filling out each type of task dialog box are described in the chapters in Section V, *Tasks*.
4. Complete the required parameter information.
5. Click OK in the task dialog box to add the statement. You can click Cancel at any time to exit a task dialog box.

➤ To add a set data request to a plan, follow the same procedure as for adding a task, except choose **Set** → *category* → *Set Data Request*.

The procedures for filling out Set statement dialog boxes are in the chapters in Section VI, *Set Data Requests*.

---

i Some task and set statements, such as Wait, or Reorganize, do not require you to specify any values. They are added immediately to the plan without any intermediate dialog boxes.

37.1.2. Adding an If Block to a Plan

To add an If condition to a plan:

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert the `If` block. If the plan does not have any statements yet, the plan root – `Plan` is already selected.
3. Choose **Conditions** → **If**. The If/Else Expression dialog box opens (Figure 37-2).

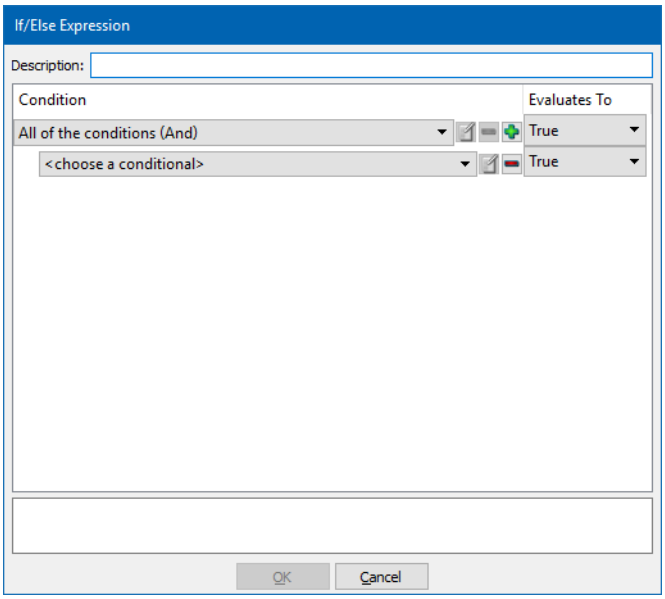


Figure 37-2. If/Else Expression dialog box

4. Optionally, in the Description box, type a description to use in the If statement instead of the condition syntax.
5. Create the conditional expression as described in "Building Condition Statements," on page 37-13.
6. Click OK. An `If` statement block is added to the plan.
7. Select the first line in the block (the `If` statement).
8. Add the tasks, sets, or other commands that you want to be executed if the condition is true.
9. Optionally, select the `else` line and add the commands you want to be executed if the condition is false.

37.1.3. Adding a While Block to a Plan

To add a While block to a plan:

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert the `While` block. If the plan does not have any statements yet, the plan root – `Plan` is already selected.
3. Choose **Conditions** → **While**. The While Expression dialog box opens. (It is identical to the If/Else Expression dialog box (Figure 37-2).)
4. Optionally, in the Description box, type a description to use in the While statement instead of the condition syntax.
5. Create the conditional expression as described in "Building Condition Statements," on page 37-13.
6. Click OK. A `While` statement block is added to the plan.
7. Select the first line in the block (the `While` statement).
8. Add the tasks, sets, or other commands that you want to be executed if the condition is true.

37.1.4. Adding a Wait Until Statement to a Plan

To add a Wait Until statement to a plan:

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert the `Wait Until` statement. If the plan does not have any statements yet, the plan root – `Plan` is already selected.
3. Choose **Conditions** → **Wait Until**. The Wait Until Expression dialog box opens. (It is identical to the If/Else Expression dialog box (Figure 37-2).)
4. Optionally, in the Description box, type a description to use in the `Wait Until` statement instead of the condition syntax.
5. Create the conditional expression as described in "Building Condition Statements," on page 37-13.
6. Click OK. A `Wait Until` statement is added to the plan.

37.1.5. Adding a Trigger to a Plan

Adding a trigger to a plan is a three-step process. First, you specify the trigger condition. Then you add the commands to be executed to the trigger body. Finally, you register the trigger in the body of the plan.

To add a Trigger to a plan:

1. Open the Plan window for a simulation object.
2. Choose **Conditions** → **Add Trigger**. The Edit Trigger Condition dialog box opens (Figure 37-3). (You can also add a trigger by choosing **Conditions** → **Register Trigger** and selecting New in the trigger list.)

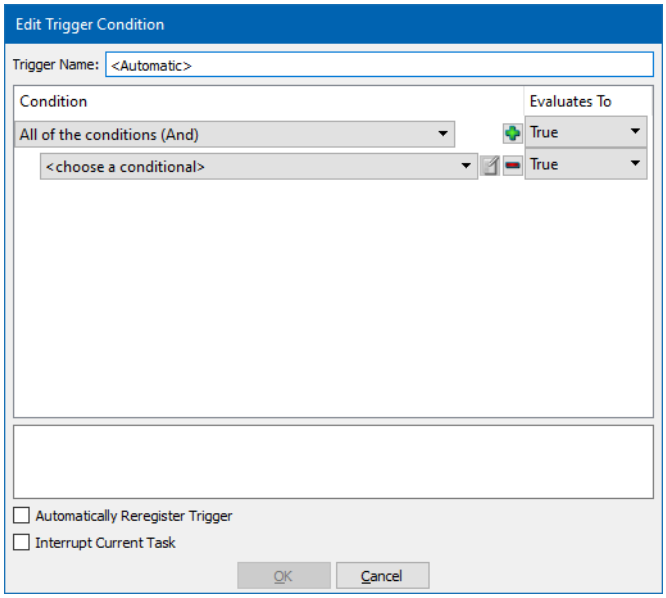


Figure 37-3. Trigger dialog box

3. Optionally, in the Trigger Name box, type a name for the trigger. If you specify a name, it is displayed in the Register command when you register the trigger. If you do not specify a name, the text of the condition is displayed in the Register command.
4. Optionally, select the Automatically Reregister Trigger check box to specify that the trigger should be automatically reregistered after it runs.
5. Optionally, select the Interrupt Current Task check box to specify that the trigger should interrupt the currently running task rather than suspending and resuming it.
6. Create the conditional expression as described in "Building Condition Statements," on page 37-13.
7. Click OK. If the Trigger Body pane is not already visible, the Plan window expands to show the it (Figure 37-4).

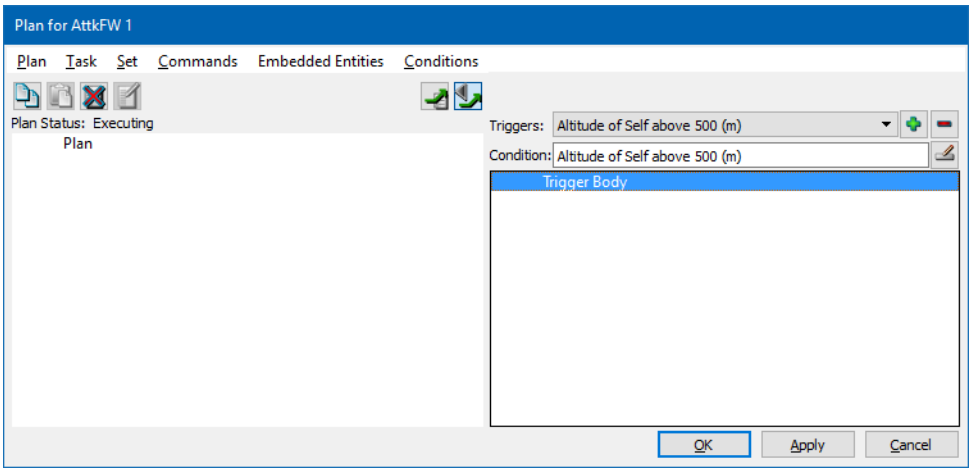


Figure 37-4. Trigger pane

8. Select the first line in the Trigger pane (`Trigger Body`).
9. Add the tasks, sets, or other commands that you want to be executed when the condition becomes true.
10. Register the trigger. For details, please see "Registering Triggers," on page 37-9.

37.1.6. Registering Triggers

Writing a trigger does not necessarily include it in a plan. To use a triger, you must register it. If you create a new trigger using **Conditions** → **Add Trigger**, you must register it manually. If you create a new trigger by choosing **Conditions** → **Register Trigger** and selecting New, it automatically gets registered at the currently selected location in the plan.

To register a trigger:

1. In the Plan window, select the statement below which you want to register the trigger.
2. Choose **Conditions** → **Register Trigger**. The Register Trigger dialog box opens (Figure 37-5).

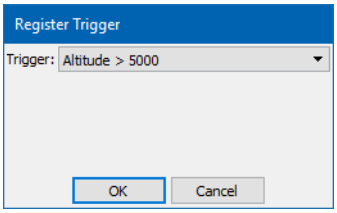


Figure 37-5. Register Trigger dialog box

3. Select the trigger you want to register from the list. (If there are no triggers in the plan, select New.)
4. Click OK.

Reregistering Triggers

By default, once the statements in a trigger get executed, the trigger is unregistered and VR-Forces no longer tests the condition. However there may be cases where you want to test for a recurring condition. You could register the trigger again when the trigger block competes. Or, when you create or edit a trigger, you can specify that it automatically reregister. For details. please see "Adding a Trigger to a Plan," on page 37-7.

As an alternative to using triggers, you can write reactive tasks, which work similarly to triggers or you can write a regular scripted task that is designed to repeatedly test a condition.

Unregistering Triggers

You can unregister triggers from within a plan. Unregistering a trigger does not remove it from a plan. It causes VR-Forces to stop testing the condition. You might want to unregister a trigger in response to a change in scenario circumstances that makes the actions in the trigger undesirable.

To unregister a trigger:

1. Select the plan statement above where you want to add an Unregister Trigger statement.
2. In the Plan window, choose **Conditions** → **Unregister Trigger**. The Unregister Trigger dialog box opens.
3. Select the trigger that you want to unregister.
4. Click OK.

37.1.7. Cleaning Up Unused Triggers

If you add triggers to a plan and then decide not to use (register) them, you can quickly remove them from the plan. (However, there is no penalty for leaving unused triggers in a plan.)

➤ To clean up unused triggers, choose **Plan** → **Clean Up Unused Triggers**.

Automatically Checking for Unused Triggers

VR-Forces can automatically check for unused triggers when you save or apply changes to a plan. If it finds any, it displays a dialog box offering you options for handling them.

To specify that VR-Forces should check for unused triggers when it updates a plan:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 37-6).

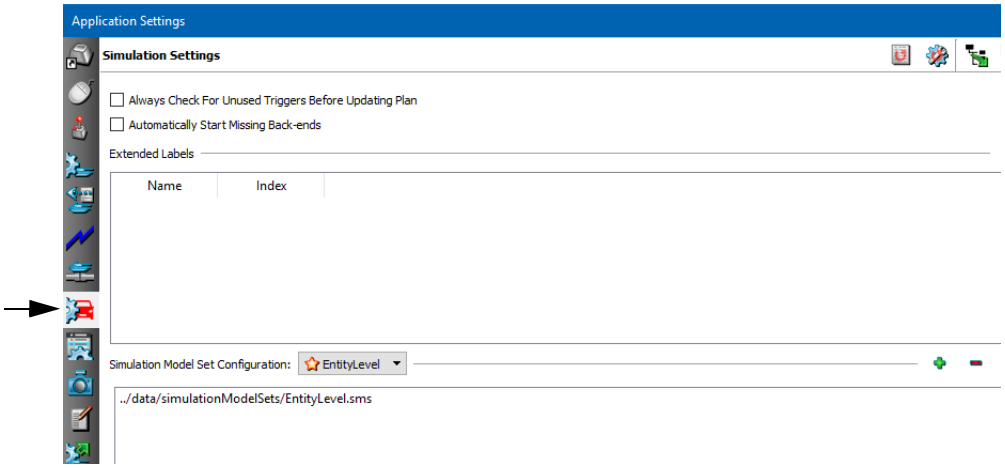


Figure 37-6. Simulation Settings page

3. Select the Always Check For Unused Triggers Before Updating Plan check box.
4. Click Close.

37.1.8. Editing a Statement

You can edit simple statements in the Plan window.

To edit a plan statement:

1. Double-click the statement or select the statement and click the Edit button (). An edit dialog box for the particular statement type opens.
2. Enter the changes you want to make.
3. Click OK in the edit dialog box.
4. Click OK or Apply in the Plan window.

37.1.9. Deleting Statements from a Plan

You can delete statements from a plan individually, or all at once. You can delete the individual statements in the statement block of a conditional statement. If you delete the conditional statement itself, all substatements are also deleted.

To delete a statement from a plan:

1. Select the statement you want to delete.
2. Click the Delete button () or press **`Delete`**.

Deleting All Statements From a Plan

To delete all statements in a plan:

1. Select the top line in the plan.
2. Click Delete or press **`Delete`**.

37.2. Printing Plans

You can print plans. The printout lists the name of the scenario, the simulation object's name, and the plan statements.

To print a plan:

1. Open a global plan or the Plan window for the simulation object whose plan you want to print.
2. Choose **Plan** → **Print**. A standard print dialog box for your operating system opens.
3. Complete the dialog box as you normally would.

37.3. Saving Changes to a Plan

You can save changes to a plan and continue editing, or save changes and exit the Plan window. The plan is saved to the scenario plan file. You do not specify a file to save it to.

➤ To save changes to a plan, in the Plan window click Apply or OK.

37.4. Building Condition Statements

If you want to add an `If` block, a `While` loop, a `Wait Until`, or a trigger, you must specify a condition to be tested. The conditional expression dialog box is a dynamic dialog box that lets you build a condition. It contains lists that have the permissible options for each step of a condition. As you select options, the dialog box redisplays, adding additional fields. You can build condition statements using comparison operators, logical operators, and tests of simulation object status.

A condition statement can have one condition or many. They can be nested and grouped. In the following sections we explain how to build a simple condition and how to build a complex condition.

37.4.1. Building a Condition Statement that Has One Condition

Although the If/Else Expression, While Expression, Wait Until Expression, and Trigger dialog boxes differ slightly, the process of building a condition is the same for all of them.

To build a condition statement that has one condition:

1. On the Conditions menu, choose the type of condition you want to create. The applicable dialog box opens (Figure 37-7). The top line in the condition specifies that all of the conditions are True. Since we are creating a condition statement that just has one condition, this line is largely irrelevant except for the Evaluates To column.

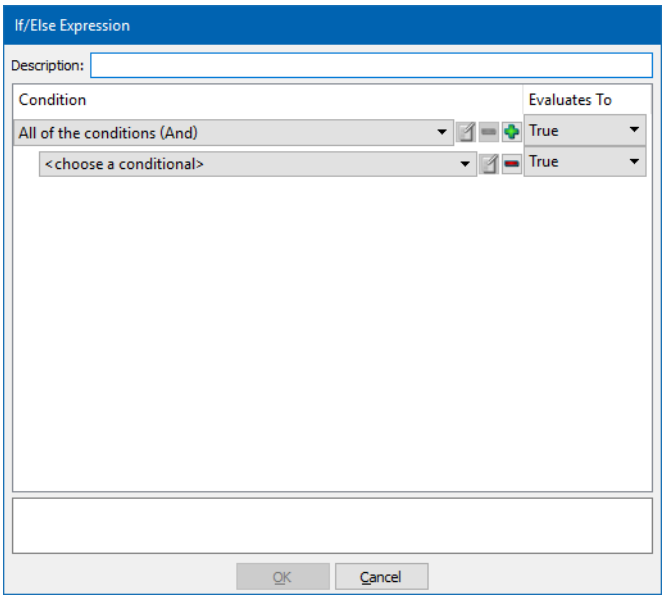


Figure 37-7. If/Else Expression dialog box

2. Optionally, in the Description box, type a short summary of the condition.
3. Optionally, in the Evaluates To list, change the entire condition from True to False.
4. In the <choose a conditional> list, select the condition that you want to test. If applicable, a dialog box opens for you to specify the parameters of the condition.
5. Specify the parameters of the condition.
6. Click OK. The condition is displayed in the display box at the bottom of the dialog box (Figure 37-8).
7. Optionally, in the Evaluates To list, change the condition from True to False. Figure 37-8 shows a simple condition.

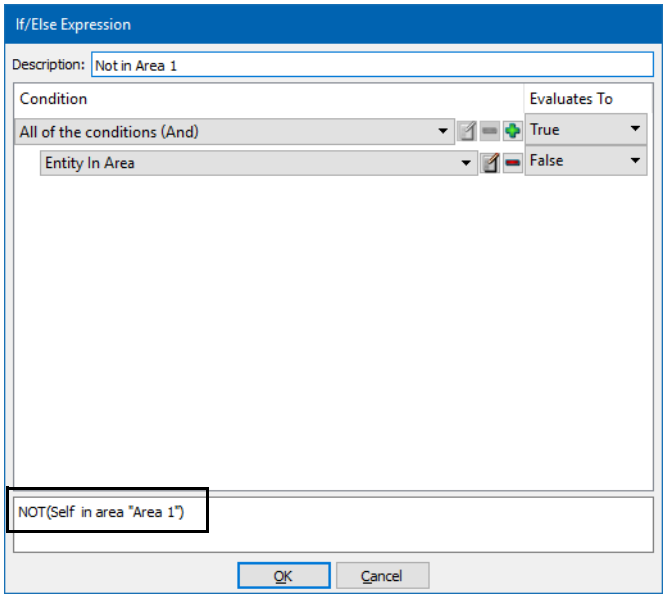


Figure 37-8. Simple conditional expression

8. Click OK. The condition statement is added to the plan or the Trigger pane. In Figure 37-9, note that the If statement uses the text from the description (in Figure 37-8) instead of the full syntax of the condition shown at the bottom of the If/Else Expression dialog box.

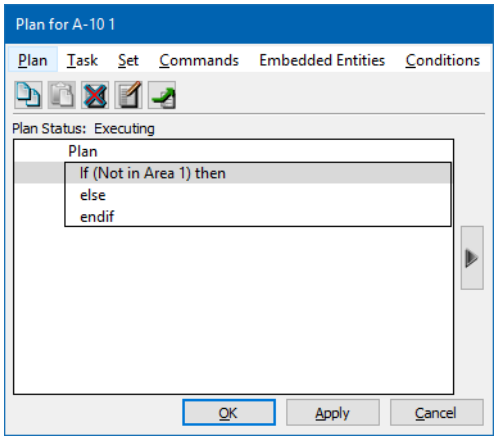


Figure 37-9. Simple conditional expression in a plan

37.4.2. Building a Complex Condition

You can build condition statements with multiple conditions, nested conditions, and logical operators. The condition expression dialog box displays the syntax of the condition as you build it. The layout of the window shows which conditions are grouped and which are subordinate to others. For an example, of building a complex condition, please see "Complex Condition Tutorial," on page 37-16.

To build a complex condition:

1. On the Conditions menu, choose the type of condition you want to create. The applicable condition expression dialog box opens. The top line in the condition specifies that all of the conditions are True.
2. Optionally change the top level logical operator to Or.
3. Optionally change the top level evaluator to False.
4. In the <choose a conditional> list, select the condition that you want to test. If applicable, a dialog box opens for you to specify the parameters of the condition.
5. Specify the parameters of the condition.
6. Click OK. The condition is displayed in the display box at the bottom of the dialog box.
7. Click the Add button (+) next to the top line of the condition. A new condition line is added.
8. Select a condition to test and specify its parameters.
9. Continue to add conditions as desired. The condition syntax updates as you add conditions.
10. Optionally, select a new logical operator in the conditions list (And or Or). A subordinate condition line is added under the logical operator and indented.

11. Add conditions for the logical operator group as you did for the top level logical operator. To simplify the view, you can expand and collapse the logical grouping by clicking the arrow to the left of the logical operator line.
12. Continue adding conditions as desired.
13. Click OK.

Complex Condition Tutorial

This section walks you through the process of creating a complex condition. It creates the following conditional expression:



```
If("M1A2 4" in area "Area 1" AND ("M1A2 2" left of "Phase Line 1" OR
  Entity "T80 1" destroyed))then
else
endif
```

The procedure assumes a scenario with four M1A2 entities, a T-80, a phase line, and an area.

To build the condition statement:

1. Open the Plan window for M1A2 1.
2. Choose **Conditions** → **If**. The If/Else Expression dialog box opens (Figure 37-10).

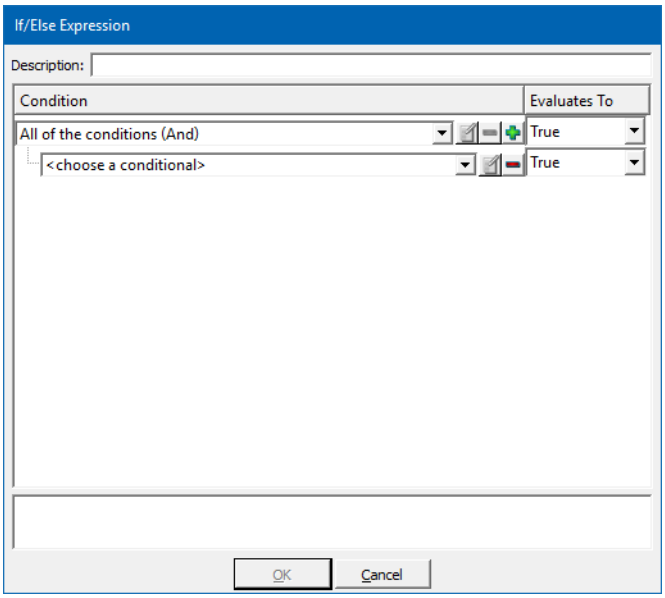


Figure 37-10. If/Else Expression dialog box

3. Type a description for the expression you want to build (Figure 37-11).

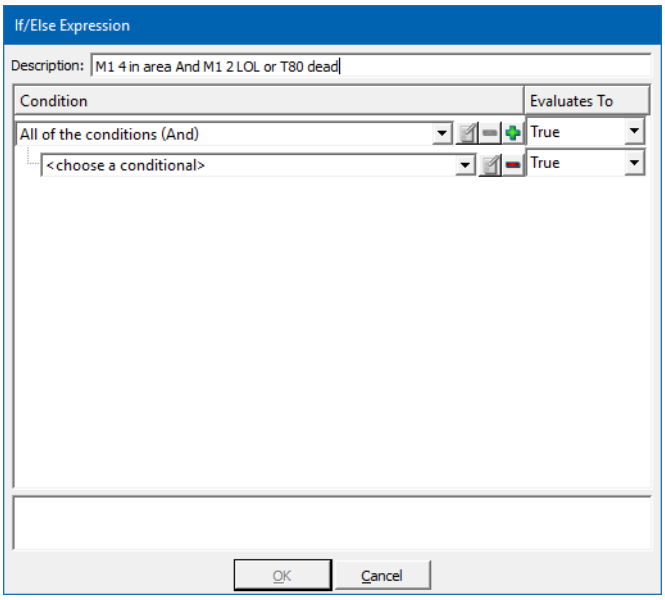


Figure 37-11. Description added

4. In the <choose a conditional> list, select `Entity in Area`. The Entity in Area dialog box opens.
5. In the Entity in Area dialog box, select M1A2 4 and Area 1.
6. Click OK. The condition is added and the If expression is updated (Figure 37-12).

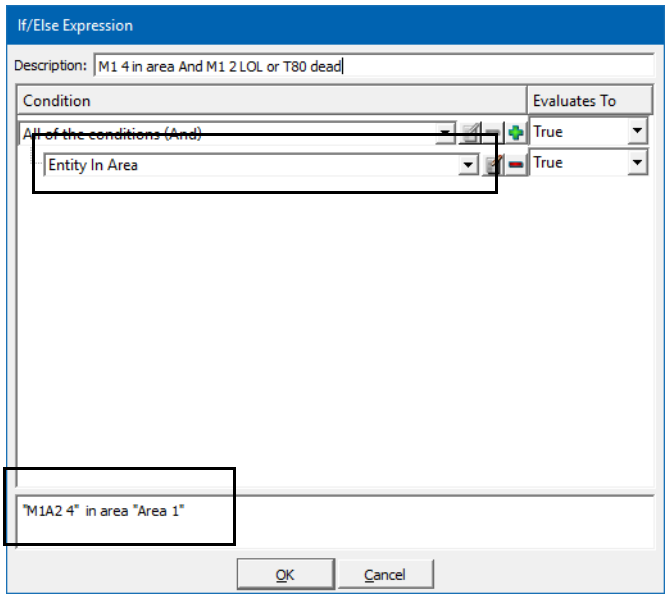


Figure 37-12. Entity in Area condition added

7. On the All of the conditions line, click the Add button (+). A new line is added to the window (Figure 37-13).

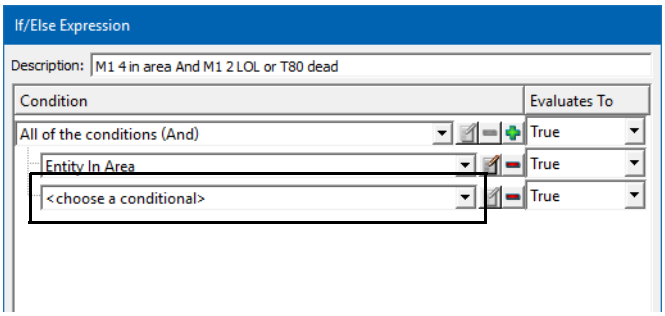


Figure 37-13. Condition added

8. In the <choose a conditional> list, select `Any of the conditions (Or)`. A new line is added to the window (Figure 37-14).

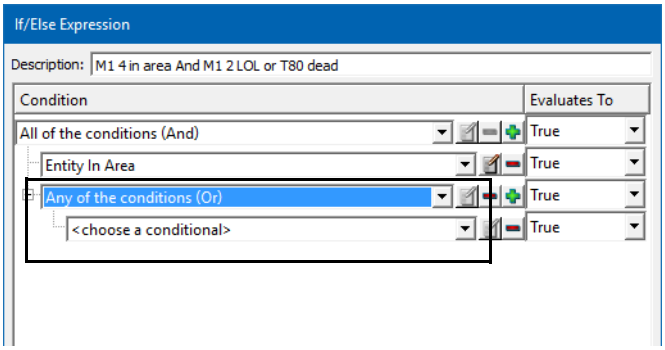


Figure 37-14. Or operator and condition added

9. In the <choose a conditional> list, select `Entity Left of Phase Line`. The Entity Left of Phase Line dialog box opens.
10. Select M1A2 2 and Phase Line 1.
11. Click OK. The condition is added and the If expression is updated (Figure 37-15).

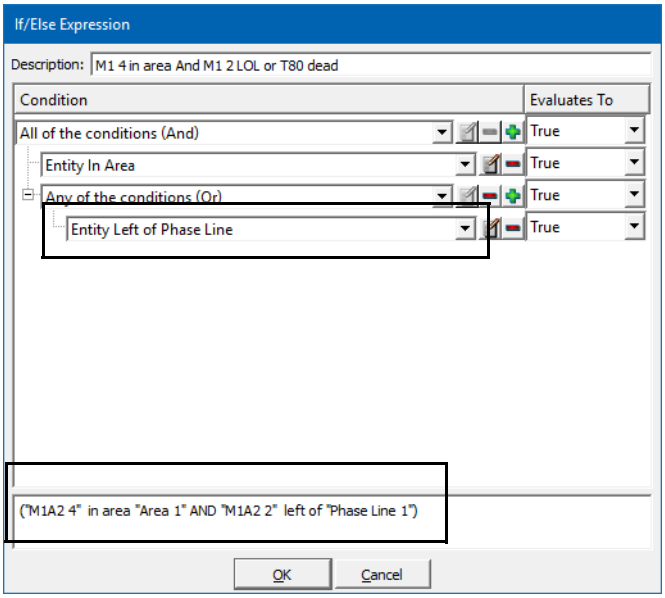


Figure 37-15. Entity Left of Phase Line condition specified

12. On the Any of the conditions (Or) line, click the Add button. A new line is added to the window (Figure 37-16).

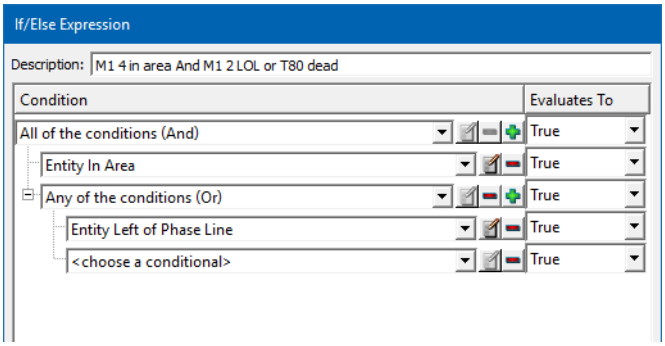


Figure 37-16. Last condition line added

13. In the <choose a conditional> list, select `Entity Destroyed`. The Entity Destroyed dialog box opens.
14. Select T80 1.
15. Click OK. The condition is added and the If expression is updated (Figure 37-17).

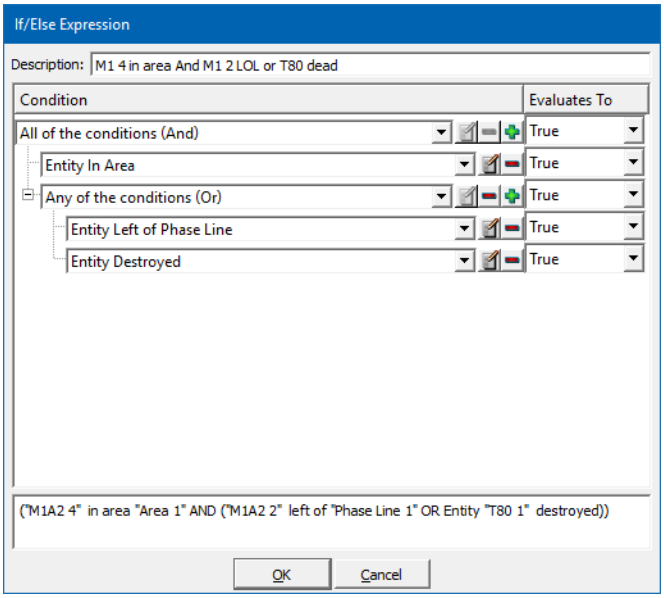


Figure 37-17. Completed expression

16. In the If/Else Expression dialog box, click OK. The If expression is added to the plan window. Notice that the expression uses the text from the Description box (Figure 37-18).

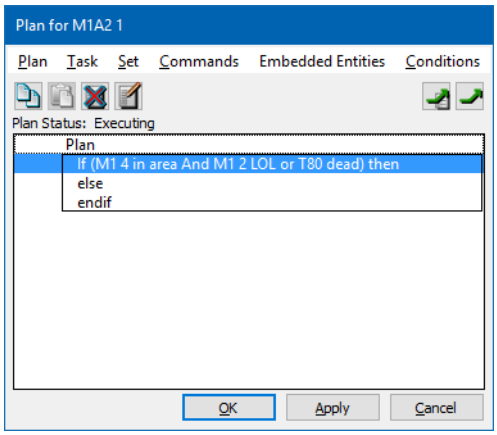


Figure 37-18. If expression using condition description

17. If you did not include a description, the entire syntax of the expression is displayed (Figure 37-19).

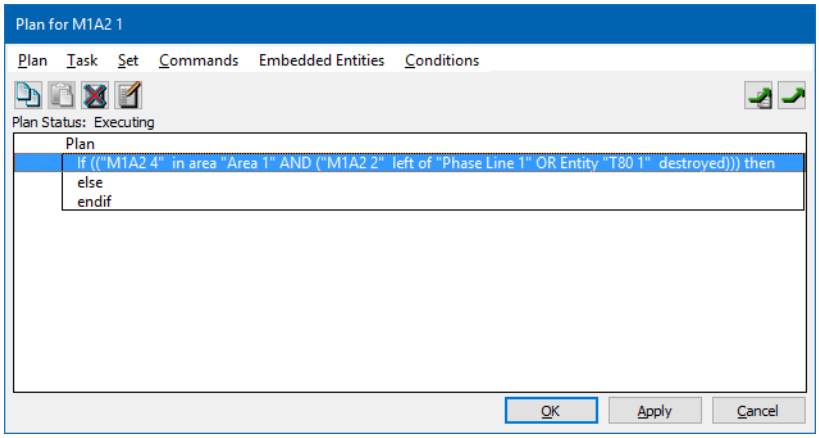


Figure 37-19. If expression using expression syntax

37.4.3. Editing Condition Statements

To edit an If, While, or Wait Until condition statement:

1. In the Plan window, double click the statement. The applicable expression building dialog box opens.
2. To edit the parameters of a condition, click the Edit button ( ) for that condition and change the parameters as desired.
3. To change the condition being tested, select a different condition in the list and specify its parameters.
4. To delete a condition, click the Delete button ( ) for that condition.
5. To change how the condition is evaluated, choose a different option in the Evaluates To list.
6. Click OK.

To edit a Trigger:

1. If the Trigger pane is not displayed, click the Expand button ( ) to show it.
2. In the Triggers list, select the trigger that you want to edit.
3. Click the Edit button ( ). The Trigger dialog box opens.
4. To edit the parameters of a condition, click the Edit button ( ) for that condition and change the parameters as desired.
5. To change the condition being tested, select a different condition in the list and specify its parameters.
6. To delete a condition, click the Delete button ( ) for that condition.
7. To change how the condition is evaluated, choose a different option in the Evaluates To list.
8. Click OK.

37.4.4. Specifying Names or Patterns in Condition Statements

Many of the conditions that you can use to build a condition statement require you to specify simulation objects for which the condition will be tested. You can specify:

- A specific named simulation object, for example, simulation object M1A2 1 is in Area 1.
- That the condition applies to "Self", for example, "I" am in Area 1. Specifying a condition for one's self makes it easy to copy plans or plan statements and paste them into the plans of other simulation objects without doing any editing.
- An enumeration, for example, any simulation object with enumeration 1:1:225:1:1:3:0 (any M1A2).
- The inverse of the selection, for example, any simulation object but myself, or any simulation object except an M1A2.
- That for units, the entire unit must satisfy the condition (the default), or that any subordinate member of a unit satisfies the condition. For example, the default behavior when Entity Is Destroyed is applied to a unit is that all members of the unit must be destroyed for the condition to be true. The alternative is that the condition is true if any member of the unit is destroyed.

The Name and Pattern specifications are mutually exclusive. You cannot specify a name and a pattern. The tab in the Condition dialog box that is visible when you click OK is the specification that gets applied.

Specifying a Name in a Condition Statement

To specify a named simulation object in a condition statement:

1. In a condition dialog box, select the Name tab (Figure 37-20).

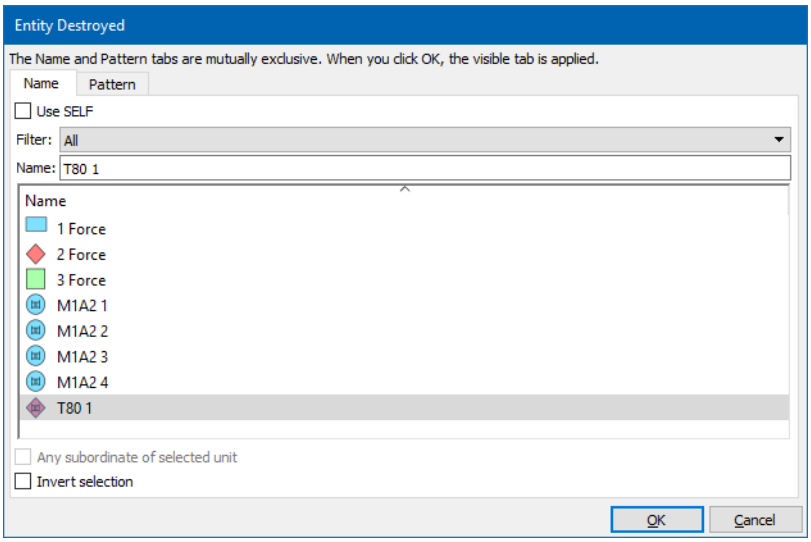


Figure 37-20. Name tab in condition dialog box

2. Do one of the following:
   - To specify the simulation object whose plan this is as the simulation object in the condition, select the Use SELF check box.
   - To specify a simulation object by name
     a. Optionally filter the list. If you want to use a plan variable, you must select the Plan Variables filter.
     b. Select the simulation object or variable in the list or type its name in the Name text box.
3. If you select a unit in the list, the Any Subordinate of Selected Unit check box is enabled. If you want the condition to be satisfied when any member of the unit meets the condition, select the check box.
4. To specify the inverse of the name, that is, any member except the selected one, select the Invert Selection check box.
5. Click OK.

Specifying a Pattern in a Condition Statement

The enumerations used in the lists on the Pattern tab are loaded from the file *./appData/settings/vrfGui/condition-entity-types.csv*. They represent a subset of the enumerations typically used for simulation objects in simulations. However, you can enter any numeric value you want in the enumeration value box.

If you expect that you will consistently need to specify enumeration values that are not part of *condition-entity-types.csv*, such as a particular country code, you can edit it to add the patterns you need.

To specify a pattern in a condition statement:



1. In a conditional statement dialog box, select the Pattern tab (Figure 37-21).

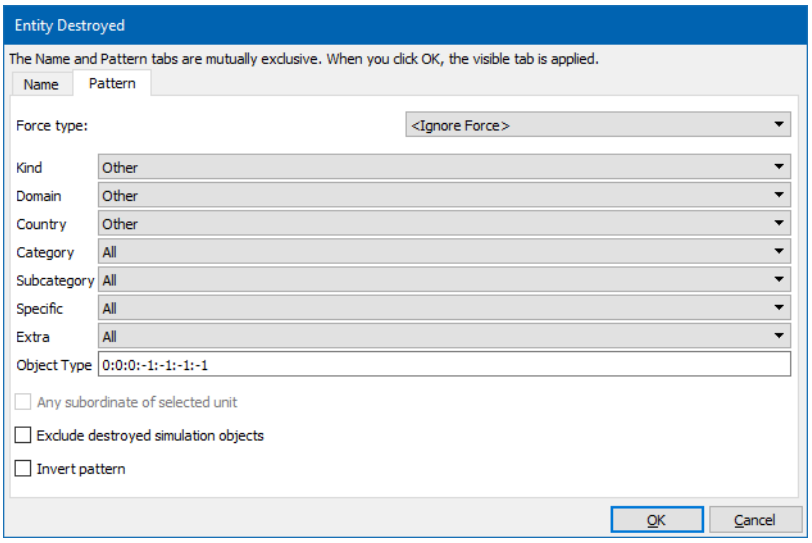


Figure 37-21. Pattern tab in condition dialog box

2. Optionally, specify a force to test for.
3. For each element of the enumeration, select an option from the list, or type an enumeration in the text box.
4. If the enumeration you enter is for a unit, the Any Subordinate Of Selected Unit check box is enabled. If you want the condition to be satisfied when any member of the unit meets the condition, select the check box.
5. To specify the inverse of the name, that is, any member except the selected one, select the Invert Pattern check box.
6. Click OK.

37.5. Using Plan Variables

You can use plan variables in most places where you would select a specific simulation object in a task, set, or conditional test. Plan variables use the names Simulation Object and Created Object. You cannot add your own variables or change the names.

The variables get assigned values only after an object has been created from within a plan (Created Object) or a conditional test has been met that identifies a simulation object (Simulation Object). If you use a variable in a plan that does not have a value, it is ignored.

A Simulation Object variable does not get assigned a value until a condition has been completely evaluated. Therefore, you cannot do something like If Entity In Area AND Simulation Object is Destroyed and expect that the Simulation Object variable refers to the entity identified by Entity in Area. In this case, the Simulation Object variable would refer to a previously identified simulation object or it would just be empty.



Remember that the values assigned to plan variables can change as a simulation object progresses through its plan.

For conceptual information about plan variables, please see "Plan Variables," on page 36-14.

To use a plan variable in plan statement:

1. In the statement creation dialog box, select Plan Variables in the Filter list. The dialog box lists Created Object and Simulation Object (Figure 37-22).

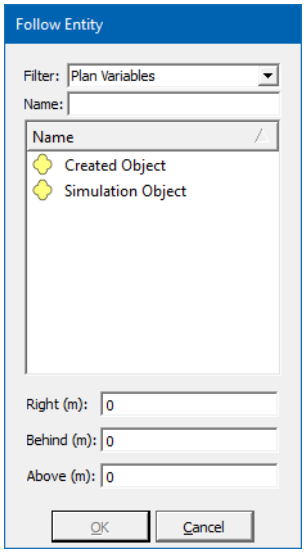


Figure 37-22. Plan variables

2. Select the variable that you want to use.
3. Complete the rest of the dialog box normally.

37.6. Creating Global Plans

Global plans are not tied to a specific simulation object. However, since they must be executed by a simulation engine, they are assigned to a specific back-end.

Global plans have a limited set of Task and Set commands and the same set of Commands menu options and conditions as individual plans. (You can add additional tasks using the VR-Forces API or by writing scripted tasks.) You can open an Information dialog box for a global plan to see its task status and console messages.

To create a global plan:

1. Choose **Simulation** → **Global Plans**. The Global Plans window opens (Figure 37-23). It lists each global plan and the simulation engine on which it is executed.

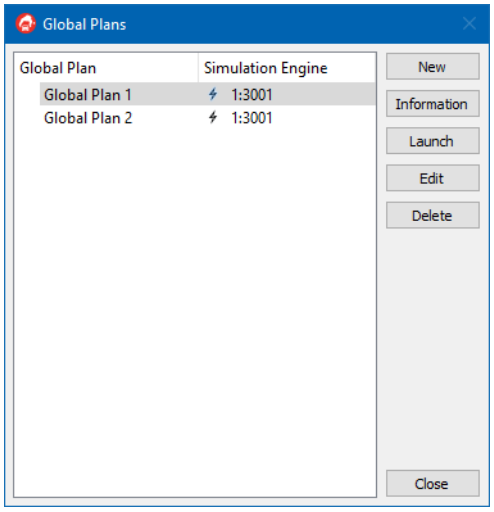


Figure 37-23. Global Plans window

2. Click New. The New Global Plan Information dialog box opens.
3. In the Plan Name box, type a name for the plan.
4. If your scenario has more than one back-end, select the back-end on which you want to run the global plan.
5. Click OK. The Global Plan window opens (Figure 37-24).

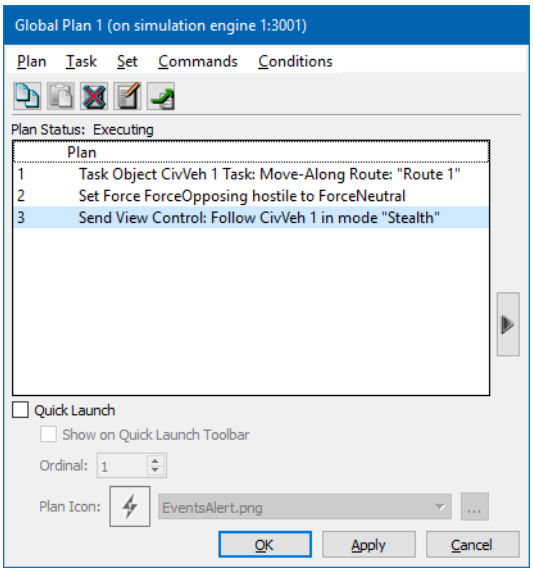


Figure 37-24. Edit Global Plan window

6. Add commands from the Task, Set, or Conditions menu, as described in "Adding a Task or Set Data Request to a Plan," on page 37-4 and "Adding an If Block to a Plan," on page 37-5. Add commands from the Commands menu as described in "Adding Commands to a Plan," on page 37-30.
7. Optionally, select the Quick Launch check box to enable quick launching of this plan. For information about using the Quick Launch Toolbar, please see "Controlling When Global Plans Run (Quick Launch)," on page 37-29.
8. Optionally, select the Show on Quick Launch Toolbar check box to add this global plan to the Quick Launch toolbar.
9. Optionally, in the Ordinal box, specify the position of this plan on the Quick Launch toolbar.
10. Optionally, change the icon for this plan on the Quick Launch toolbar, as follows:
    a. Click the Browse button next to the icon name. A file chooser window opens.
    b. Select the file you want to use for the icon.
    c. Click Open.
11. Click OK.

37.6.1. Controlling When Global Plans Run (Quick Launch)

If you enable quick launch for a global plan, it does not start executing until you launch it. You can launch it from the Quick Launch toolbar, from the Global Plans window, or from another plan. (You can enable quick launch without putting an icon on the Quick Launch toolbar.) When you launch a global plan, it runs to the end. When it finishes, you can launch it again.



When the Quick Launch toolbar is docked, it shows an icon for each quick launch item. To find out what the icon represents, mouse over it and read its tooltip. If you undock the toolbar, it displays the name for each quick launch item.

➤ To launch a global plan from the Quick Launch toolbar, click its icon.

To launch a global plan from the Global Plans window:

1. Choose **Simulation** → **Global Plans**. The Global Plans window opens (Figure 37-23).
2. Select the global plan you want to launch. If it is not running, the Launch button becomes active.
3. Click Launch. The global plan runs. When it completes, the Launch button becomes active again.

To launch a global plan from a plan:

1. Open the plan window for an individual plan or a global plan.
2. Choose **Commands** → **Launch Global Plan**. The Launch Global Plan window opens.
3. Select the global plan you want to launch.
4. Click OK.

37.6.2. Opening an Information Dialog Box for a Global Plan

To open an Information dialog box for a global plan:

1. Choose **Simulation** → **Global Plans**. The Global Plans window opens (Figure 37-23).
2. Select the global plan for which you want to open an Information dialog box.
3. Click Information.

37.7. Adding Commands to a Plan

The Commands menu lets you assign tasks and set data requests to simulation objects and create the objects on the Simulation Objects Palette and Tactical Graphics Palette. There are also several commands that are only available on the Commands menu. Except as noted in this section, the procedures for adding global commands and conditional statements to a global plan are the same as the procedures for adding them to individual plans.



Since a global plan has no idea what type of simulation object you might want to give a task or set data request to, when you add a task or set, the full list of possible tasks and sets is available. There is no context sensitivity like there is when you select a simulation object and view the Task or Set menu. It is up to you to be sure that the task or set you are assigning to a simulation object is supported by that object.

The best way to determine this is to create a simulation object of the type to which you want to give a task and see if the task or set data request is available on its context-sensitive Task or Set menu. You can also checkthe Simulation Object Editor to see if a simulation object has the system used for a particular task.

37.7.1. Adding Tasks and Sets from the Commands Menu

The commands on the Task and Set menus in an individual plan apply implicitly to the simulation object for whom you are writing a plan. Tasks and sets on the Commands menu must be assigned explicitly to a target simulation object.

Other than the requirement to specify the target simulation object, adding a global task or set command to a plan uses the same procedure as adding any task or set to an individual plan.

To assign a task from the Commands menu:

1. In a Plan window or Global Plan window, choose **Commands** → **Send Task**. The Send Task dialog box opens (Figure 37-25).

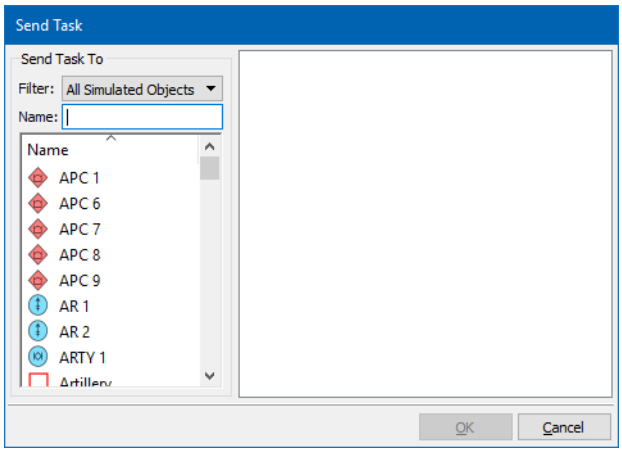


Figure 37-25. Task dialog box



The procedure for adding a set data request is the same as for a task, except that you choose **Commands** → **Send Set** and the Send Set dialog box opens.

2. In the Apply Command To group box, select a simulation object from the list or type its name in the Name box. The simulation object does not have to exist in the scenario at the time you add the command, but you should be sure that it will be created before the command is run. The window displays a list of tasks for the selected simulation object (Figure 37-26). If the simulation object already exists in the scenario, the list of tasks is context-sensitive.

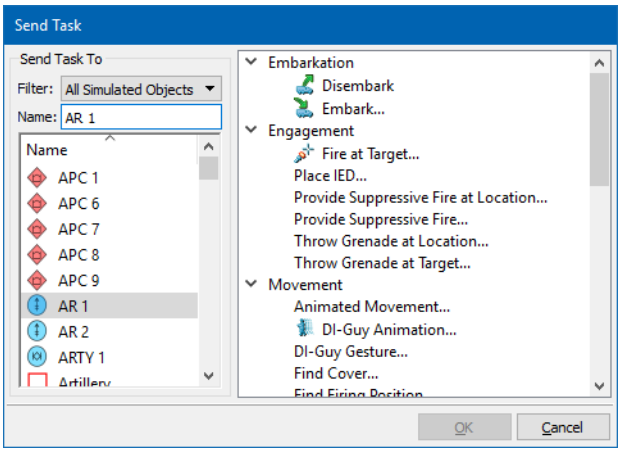


Figure 37-26. Task list

Remember that a command task affects a simulation object like an independent task. It interrupts the current task and causes the simulation object to abandon its plan. If you are adding a command to an individual plan, be sure that you are assigning it to some other simulation object and not the simulation object for whom you are writing the plan. (Unless you want the simulation object to abandon its plan.)

---

3. Select the task you want to assign.
4. Click OK. If the task takes parameters, a dialog box opens.

   If the task does not take parameters, such as Wait, the command is added to the plan.
5. If the task takes parameters, specify the parameters and click OK in the task-specific dialog box. Please see the chapters in Section V, *Tasks* and Section VI, *Set Data Requests* for the details of assigning specific tasks and sets.
6. Click OK. The command is added to the plan.

Editing Task and Set Commands in Plans

If you want to edit a task or set command in a plan, you can:

- Delete the plan statement and start over.
- Edit the existing statement. If you edit the existing statement, you can:
  - Delete the current task and choose a different one.
  - Edit the parameters of the task.

To edit a task or set command in a plan:

1. Select the plan statement you want to edit.
2. Click the Edit button, or double-click the statement. The Send Task or Send Set dialog box opens. The task list is grayed out. If you are not sure of the task that you are editing, you can read the text of the command below the window (Figure 37-27).

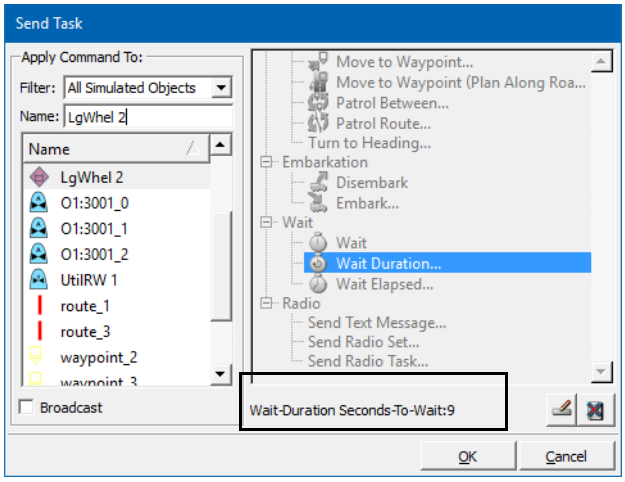


Figure 37-27. Send Radio Task dialog box for plan command

3. To clear the task and start over, click the Delete button ( ).
4. To keep the same task and change the parameters, click the Edit button ( ). The appropriate dialog box opens and you can edit the parameters.
5. Click OK.

37.7.2. Sending Text Messages from the Commands Menu

The Send Text Message command lets you send a text message to a simulation object without using the radio network. The receiving simulation object can test for receipt of the message in a conditional statement in its plan. (For details, please see "Receive Text Message," on page 36-12.)

Because the Send Text Message command does not use the radio network, simulation objects in forces other than the sending simulation object can receive the text message. Additionally, text messages sent using the Send Text Message command do not display communication lines and the sending object does not display squawk indicators.

To send a text message from the Commands menu:

1. In a Plan window or Global Plan window, choose **Commands** → **Send Text Message**. The Send Text Message dialog box opens (Figure 37-28).

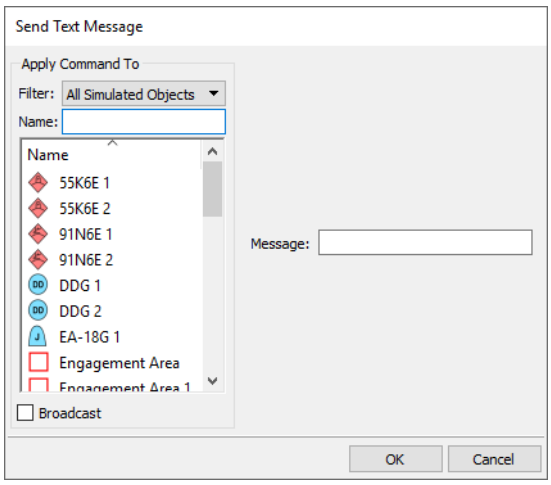


Figure 37-28. Send Text Message dialog box for plan command

2. To send the message to all simulation objects in the scenario, select the Broadcast check box.

   To send the message to a specific simulation object, in the Name text box, type the name of the simulation object that will receive the message, or select the simulation object in the list or on the terrain.

3. In the Message box, type the message.
4. Click OK. The command is added to the plan.

37.7.3. Sending Console Messages

You can send messages to be displayed in the console on an object's Information dialog box and to the back-end console. Messages sent by global plans go to the back-end console. Messages sent from a simulation object's plan go to the console in its Information dialog box.

If the notification level for a console message is lower than the notification level configured for the back-end or the target simulation object, the message is not displayed. For example, if a simulation object's notification level is set to Info, it receives Error, Warn, and Info messages, but not messages set to Verbose or Debug.

To send a console message from a plan:

1. In a plan window, choose **Commands** → **Console Message**. The Console Message dialog box opens.
2. Select the notify level from the list.
3. Type the message you want to send.
4. Click OK.

37.7.4. Creating Objects from Within a Plan

You can create any of the objects on the Simulation Objects and Tactical Graphics palettes from within a plan.

To create an object from a plan:

1. In a plan window, choose **Commands** → **Create** → *palette*, where *palette* is one of the object creation palettes. The selected palette opens.
2. Select the object you want to create.
3. Create the desired object as you would if you were creating it directly from one of the object creation palettes. When you are finished creating the object, a Create command is added to the plan and the object itself disappears. (The object disappears because you are not actually creating the object right now, you are just creating the command that will create it when the plan executes.)

37.7.5. Deleting Objects from Within a Plan

You can delete any object in a scenario from a plan.

To delete an object from a plan:

1. In a plan window, choose **Commands** → **Delete Object**. The Delete Object dialog box opens.
2. Type the name of the object you want to delete or select the object in the list or on the terrain.
3. Click OK.

Deleting One's Self from the Scenario

The Delete Object global command lets you delete any specified object. If you just want to delete the simulation object whose plan you are editing, you can use the Delete Self command. Like the Use Self option in some conditions, Delete Self provides a non-entity-specific command that is useful for plans that you want to assign to multiple simulation objects.

➤ To assign the Delete Self command, choose **Commands** → **Delete Self**.

37.7.6. Issuing a Plan

The Issue Plan global command lets you assign a plan to a simulation object. When you issue a plan, the new plan replaces whatever plan or task the simulation object is executing. Although you can issue a plan to any simulation object, this command is particularly useful when you create a simulation object after a scenario has started.

When you create a scenario, you cannot create a plan for a simulation object that does not exist. Therefore, if you create a simulation object after the scenario starts (using the Create global command), it is difficult to send it a sequence of tasks because global commands are not executed in sequence (as discussed in "Introduction to Individual Plans and Global Plans," on page 36-2). The Issue Plan command solves this problem.

Issue Plan is a concurrent task. Including it in a plan does not affect the current task that the simulation object is executing or the sequencing of tasks in its plan.

To issue a plan to a simulation object:

1. In a plan window, choose **Commands** → **Issue Plan**. The Issue Plan dialog box opens (Figure 37-29).

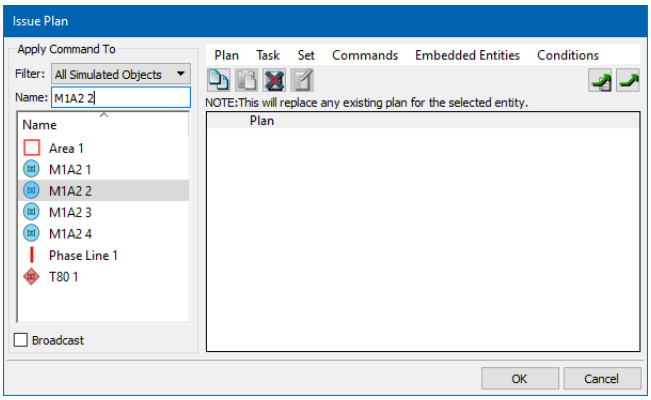


Figure 37-29. Issue Plan dialog box

2. In the Apply command To: group box, select the simulation object to which you want to assign this plan or type the name of the simulation object in the Name text box.

   If this is a simulation object that does not yet exist (because you plan to create it after the scenario starts), you must type the name and you are responsible for ensuring that the name you type matches the name of the simulation object that gets created. You must also ensure that the scenario does not have more than one entity with the same name.

   If you want to issue a plan to an entity that has just been created, you can use the Created Object plan variable to identify the object. For more information about the Created Object variable, please see "Plan Variables," on page 36-14.

3. Use the right side of the dialog box to write a plan. It has the exact same set of menus and commands as the Plan dialog box. Follow the procedures described elsewhere in this chapter to write the plan.

4. Click OK.

37.7.7. Sending View Control Messages

You can send view control messages to set the view mode for observers in the simulation. This includes the observers in VR-Forces and in any other VR-Vantage applications in the exercise. (The VR-Forces front-end is built using the VR-Vantage toolkit and is, therefore, considered a VR-Vantage application.) When you send a command you can direct it to a specific observer or to all similarly named observers, for example, all Observer 1s.

You can also send saved observer views. Observer views work only with VR-Forces. They do not affect other VR-Vantage applications. If you send an observer view, you can specify a transition time to the new view, similar to the automated transition described in "Animating Transitions Between Observer Views," on page 51-31.



- The View Control command does not support all of the view modes. However, you can create a saved view using any view mode and send the saved view (to VR-Forces only).
- If you send an observer view, the current values for the observer view are put into the plan. If you subsequently change the observer view, the changed values do not get picked up.
- If you record a scenario with the MAK Data Logger, it records view control commands. You can use them to create demonstration recordings that automatically change to different views as the recording progresses.

To send a view control message:

1. In a plan window, choose **Commands** → **Send View Control**. The Send View Control window opens (Figure 37-30).
2. Select an Observers to Control option. Observer 1 sends the view control command to Observer 1 in all VR-Vantage applications. If you want to direct the view control message to a particular observer:
   a. Select Specific. The dialog box expands to show a list of observers.
   b. Select the observers you want to control by object name (such as 1:3101 Observer 1) or by observer name (such as Observer 2).
   c. Click OK.

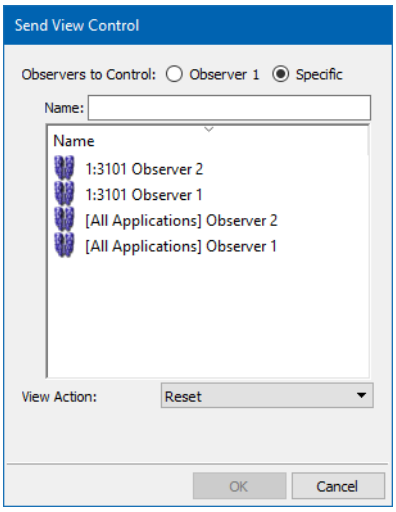


Figure 37-30. Observers to control

3. In the View Action list, select the view control you want to send, as follows:
   – Reset. Reset the observer view to the default view.
   – Detach. Detach the observer from an attached simulation object.
   – Observer View. Sends the selected observer view (from the Observer Views Panel).
   – *Attach mode*, where *Attach mode* is Absolute or one of the attach modes. Sends the selected attach mode.

   For every command except Reset and Detach, the dialog box redisplays to add a selection window appropriate to the view mode.
4. If you selected Observer View as the View Action, select the observer view that you want to send (Figure 37-31). Optionally, specify a transition time, in seconds.

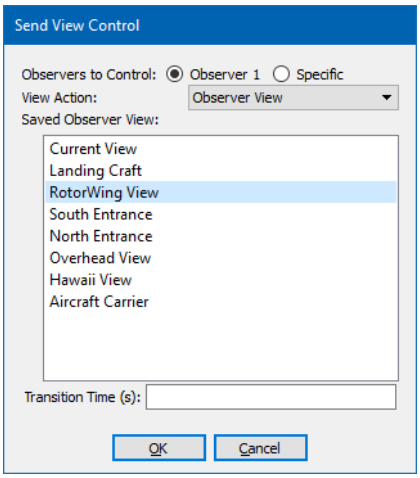


Figure 37-31. Observer view

5. If you selected Absolute mode or an attach mode as the View Action, complete the dialog box (Figure 37-32) as follows:
   a. In the Observer Mode list, select the observer mode for the view control message.
   b. If you selected Absolute mode, specify the location of the observer and, optionally, an offset orientation.
   c. If you selected an attach mode, select the simulation object to attach to.
   d. If you selected an attach mode that supports an offset vector or offset orientation, optionally select Manual and specify an offset. The default offset is 0,0,0.

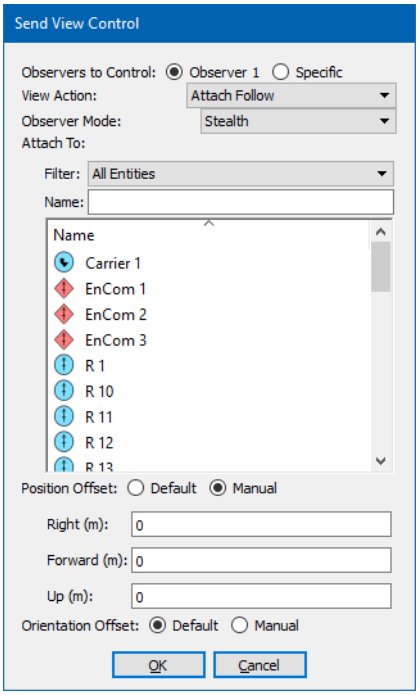


Figure 37-32. Attach mode

e. If you select Track, specify the location for the observer to track from.

f. If you select Tether Track, select a secondary simulation object and, optionally, a position offset.

6. Click OK.

37.8. Writing Plans for Units

To write a plan for a unit:

1. Select the unit.
2. Follow the procedures for writing a plan for an individual simulation object.



Please see "Independently Tasking Unit Members," on page 25-11 for notes about how simulation objects that are members of units respond to individual plans and independent tasks.

37.9. Writing a Plan for Multiple Simulation Objects

You can write a plan and assign it to multiple simulation objects at the same time. If any of the simulation objects already has a plan, it is overwritten by the new plan. After you assign a plan using this procedure, you can subsequently edit the plans for particular simulation objects and the changes do not affect the other simulation objects.



If you select several simulation objects and they all have the exact same plan, the plan window displays that plan. If any of the simulation objects have different plans, you are given the option to continue or cancel. If you continue, VR-Forces displays an empty Plan window and the new plan replaces whatever plans the simulation objects previously had.

To write a plan for multiple simulation objects:

1. Select the simulation objects to which you want to assign the same plan.
2. Choose **Objects** → **Plan**. The Plan window opens.
3. Add statements to the plan as you would if you were writing a plan for an individual simulation object.
4. Click OK. The plan is assigned to the selected simulation objects.

37.10. Writing Plans for Remote Entities

If you attach components to remote entities, you can write plans for those entities. To assign plans, the entities must exist in the exercise, must have attached components, and must be present in the exercise when VR-Forces saves the scenario. The plans must be appropriate for the attached components. Movement tasks are not appropriate, because VR-Forces cannot control the location of a remote simulation object. Appropriate plan statements might be to enable spot reporting, or to send a radio message. For details about how to attach components to remote entities, please see "Attaching Components to Remote Entities," on page 81-2.

37.11. Copying Plans and Plan Statements

You can copy plans and individual plan statements and paste them elsewhere in a plan or into another simulation object's plan. You can even paste them into another scenario if you do not close VR-Forces between the copy and paste operations.



In addition to copy and paste, you can drag statements and condition blocks from one location to another in a plan and to other plans.

To copy plans and plan statements:

1. Open the Plan window for the plan that you want to copy.
2. To copy the entire plan, select the top line in the plan (`Plan`). To copy a statement, select the statement. If you select a conditional statement, the entire statement block is selected.
3. Choose **Plan** → **Copy Plan** or click the Copy button ().
4. Open the Plan window for the simulation object you want to copy the plan to.
5. Select the line just above where you want to insert the copied plan.
6. Click the Paste button ().
7. Continue editing the plan.
8. Save the plan.

37.12. Restarting a Plan

There may be occasions when you want to restart a simulation object's plan without restarting a scenario. You can restart a plan interactively or by placing a Restart Plan command in a plan.

When you restart a plan, VR-Forces processes the statements in the plan, starting from the first statement. However, the simulation object executes tasks starting from its current location, not the location it had at the start of the scenario (unless it has not moved since the start of the scenario).

To restart a plan interactively:

1. Select the simulation object whose plan you want to restart.
2. Choose **Objects** → **Manage Plan** → **Restart Plan**.

To restart a plan from within a plan:

1. Open a simulation object's plan or a global plan.
2. Choose **Commands** → **Restart Plan**. The Restart Plan dialog box opens.
3. Select the simulation object whose plan you want to restart.
4. Click OK.
5. Save the plan.

To restart a plan from the Plan window:

1. Open a simulation object's plan.
2. Click Restart.

37.13. Abandoning a Plan

If you give a simulation object that is executing a plan an independent task, it abandons the plan. You can also order a simulation object to stop executing its plan. When you abandon a plan, the following things happen:

- The simulation object stops whatever task it is working on, even if it is an independent task.
- The plan is marked as complete.
- All registered triggers are deleted.
- The statement indicator skips to the end of the plan.
- The simulation object enters a wait state.



If you give a simulation object that is executing a plan an independent task, VR-Forces prompts you before it abandons the plan. (There is no prompt for the Abandon Plan command.) You can disable the prompt. For details, please see "Enabling and Disabling the Task Confirmation Prompt," on page 25-5.

You can abandon a plan interactively or by sending an Abandon Plan command in an individual or global plan.

To abandon a plan interactively:

1. Select the simulation object whose plan you want to abandon.
2. Choose **Objects** → **Manage Plan** → **Abandon Plan**.

To abandon a plan from within a plan:

1. Open a simulation object's plan or a global plan.
2. Choose **Commands** → **Abandon Plan**. The Abandon Plan dialog box opens.
3. Select the simulation object whose plan you want to abandon.
4. Click OK.
5. Save the plan.

To abandon a plan from the Plan window:

1. Open a simulation object's plan.
2. Click Abandon.



VIII. Tactical Graphics

Tactical graphics are objects that represent points, lines, areas, and information overlaid on the terrain. This section describes overlays and particular details about tactical graphics. For generic information about creating tactical graphics, please see Chapter 14, *Creating and Placing Objects*. For information about combat engineering objects, which are displayed similarly to tactical graphics, please see "Combat Engineering Objects," on page 22-11.



38. Introduction to Tactical Graphics

This chapter is a brief introduction to tactical graphics.

Introduction 38-2
Points 38-4
Phase Lines 38-4
Routes 38-5
Areas 38-5
Obstacles 38-5
Displaying Tactical Graphics 38-6
- Showing and Hiding Tactical Graphics 38-7
- Showing and Hiding Tactical Graphics Labels 38-8
- Displaying Height-Above-Terrain Lines for Vertices 38-8

38.1. Introduction

VR-Forces lets you add tactical graphics to scenarios. Tactical graphics are objects that represent points, lines, areas, and information overlaid on the terrain. They are automatically assigned to overlays. (You can configure the default overlay assignments and can change the overlay at or after creation.) Simulation objects have knowledge of some types of tactical graphics, which allows tasks and plans to take them into account. VR-Forces supports the following types of tactical graphics:

Table 38-1: Types of tactical graphics

Tactical Graphic	Description
Control objects	Simulation objects are aware of control objects and can respond to them. They avoid obstacles, can move to waypoints, follow routes, and so on.
Waypoint	A specific X, Y, Z coordinate.
Area	A polygonal area. Areas are considered to have volume, so that air and subsurface entities can interact with them.
Route	A sequence of points in three-dimensional space.
Phase Line	Infinitely long line defined by two points.
Obstacle	A polygonal area on the terrain that simulation objects try to avoid if obstacle avoidance is enabled.
Linear Obstacle	A line on the terrain that simulation objects try to avoid if obstacle avoidance is enabled. Linear obstacles are supported only for entity-level scenarios. They work only in navigation areas.
Pedestrian Path	When humans plan paths in navigation areas that have navigation data, if a pedestrian path is present that can take them to their destination, they will use it. You can think of a pedestrian path as a route that a human will automatically choose to follow instead of being told to follow it. Pedestrian paths are supported only for entity-level scenarios.
Disaggregation Area	An area in which an aggregated unit automatically disaggregates if automatic disaggregation is enabled. (Entity-level only)
Terrain Page-in Area	An area in which streaming or paged terrain is given priority for paging in.
Other tactical graphics	
Terrain Profile Line	A line that lets you examine the profile of the terrain along the line.
Intervisibility Line	A line that shows line-of-sight between two points.

Table 38-1: Types of tactical graphics

Tactical Graphic	Description
Lines	Generic lines and lines with particular graphics, such as main axis and fortified line. Simulation objects treat most lines as routes.
Points	Various military symbology, such as point of contact and decision point. Simulation objects treat most points as waypoints.
Areas	Minefields and other areal objects. These are not treated like area control objects.
Text	Any arbitrary informational text on the screen.
Symbols	Symbols that do not represent simulated objects. They are just graphics on the screen and do not affect the simulation.
Circles, ellipses, and boxes	These are just graphics and do not affect the simulation.
Volumetric objects	Volumetric areas, circles, boxes, ellipses, and corridors allow you to test for a simulation object's presence in a 3D volume rather than just a 2D area. They are useful for defining air corridors and airspace.
Combat engineering objects	Combat engineering objects are lines and areas that can impede the progress of simulation objects and cause damage to them. Combat engineering objects are supported only in aggregate-level scenarios. For information about how they work, please see "Combat Engineering Objects," on page 22-11.



In aggregate-level scenarios, linear obstacles and pedestrian paths are not supported.

Tactical graphics have names. New objects are assigned default names in the format `Point 1`, `Point 2`, `Route 1`, `Route 2`, and so on. You can specify an alternative name when you create the object. A graphical object's name can be up to 255 characters long.

Tactical graphics can have labels and extended labels. They do not have to be unique and do not have a length restriction.

Tactical graphics are assigned to a force. Waypoints are displayed using their force color. To determine the force of other tactical graphics, examine the Force attribute in the Environmental State page of the graphic's Information dialog box. For more information, please see "Displaying Information About an Object," on page 16-3.

The following sections provide more information about the basic control objects. For information about the other tactical graphics types mentioned in this section, please see:

- Chapter 48, *Intervisibility*.
- "Terrain Profiles," on page 56-2.
- "Manually Paging-In Terrain," on page 58-6.
- "Using Disaggregation Areas," on page 23-12.
- Chapter 14, *Creating and Placing Objects*.
- Chapter 39, *Overlays*.

38.2. Points

A point identifies an X, Y, Z coordinate in the terrain. You can command a simulation object to travel to a point and to travel back and forth between points. Figure 38-1 illustrates waypoints in the 3D and 2D views. In 3D, the bottom of the symbol points to the location of the waypoint. In 2D, it is centered. You can place several different kinds of points on overlays.

A point can have an altitude value. This allows you to send aircraft to waypoints or have them patrol between points.

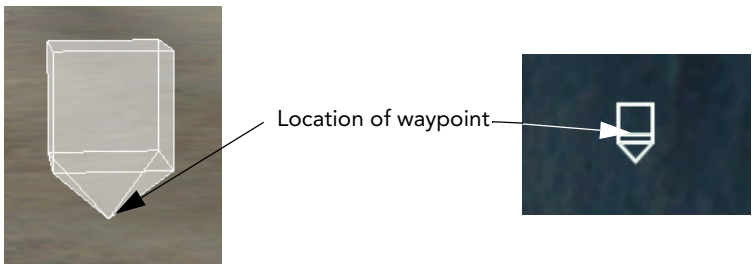


Figure 38-1. Waypoint symbol

38.3. Phase Lines

A phase line is defined by two points, but is considered to extend indefinitely in either direction and have infinite height. A phase line is used to test the location of a simulation object, for example: Is simulation object A to the left of phase line B? The left side of the line is determined relative to the direction in which the line is drawn. A phase line's label is placed at the beginning of the line. Therefore, you can always determine which side of a line is the left side. Figure 38-2 illustrates two phase lines. The arrows are placed to the left of the line and indicate the direction of the line.



Simulation objects cannot move along phase lines, so do not create a phase line if you want a two point route.

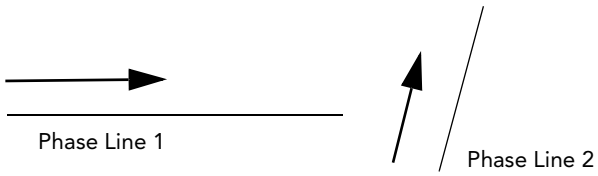


Figure 38-2. Phase line direction

38.4. Routes

A route is a sequence of points in three-dimensional space. Like phase lines, routes have a direction. A line's name is displayed at its beginning.

Simulation objects can move along routes from one end to the other, and can patrol back and forth along them.

- Do not create a two-point route if you want a phase line. VR-Forces cannot determine whether a simulation object is to the left or right of a route.
- Many "line" tactical graphics are treated as routes by simulation objects.

38.5. Areas

An area defines a polygonal area on the terrain. (This is not a terrain polygon, it is just a graphic.) An area is used to test the location of a simulation object, for example: Is simulation object A in area B?

38.6. Obstacles

Obstacles represent lines and areas of the terrain that would cause a simulation object problems, such as a ditch or minefield. In entity-level scenarios, if obstacle avoidance is enabled, if a ground entity encounters an obstacle in its path, it tries to drive around the obstacle.

In addition to obstacles, aggregate-level scenarios support combat engineering objects and contamination areas. These objects can defend, damage, or impede simulation objects.

The many combat engineering objects available in aggregate-level scenarios make simple obstacles comparatively less useful. Linear obstacles are not supported in aggregate-level scenarios. They only work in navigation areas.

38.7. Displaying Tactical Graphics

You can configure the display of tactical graphics as follows:

- Enable and disable the display of tactical graphics.
- Show or hide vertex indicators for point, linear, and areal objects.
- Show or hide height-above-terrain lines.
- Display height-above-terrain lines only for selected graphics.

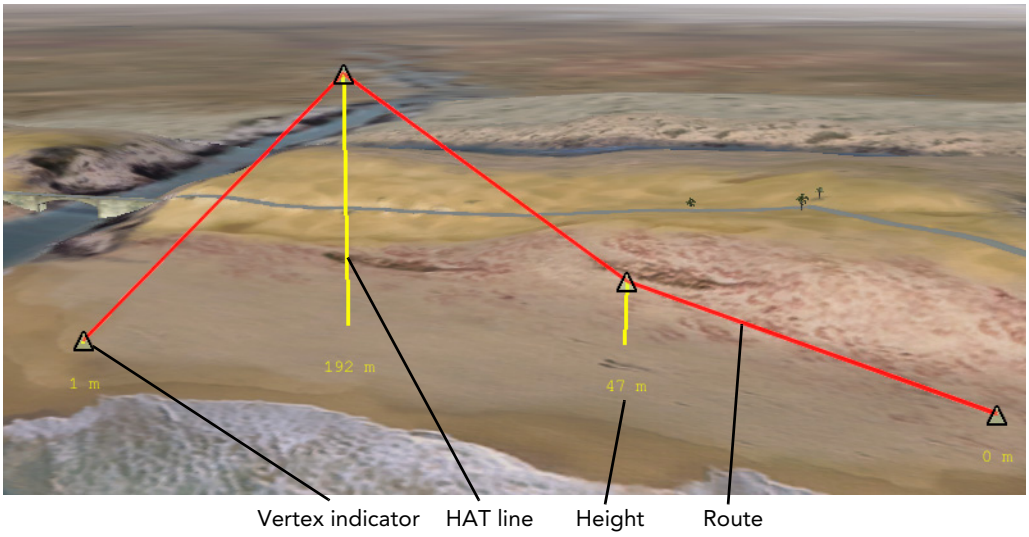


Figure 38-3. Tactical graphics with height-above-terrain lines (3D)

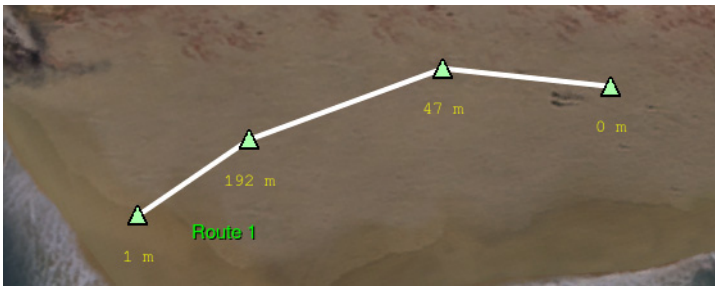


Figure 38-4. Tactical graphic (2D)

38.7.1. Showing and Hiding Tactical Graphics

You can control the display of tactical graphics from the Display Settings dialog box, Tactical Graphics Display Settings page, from menu and toolbar options, and from the Overlays tab.

To show or hide tactical graphics:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 38-5).

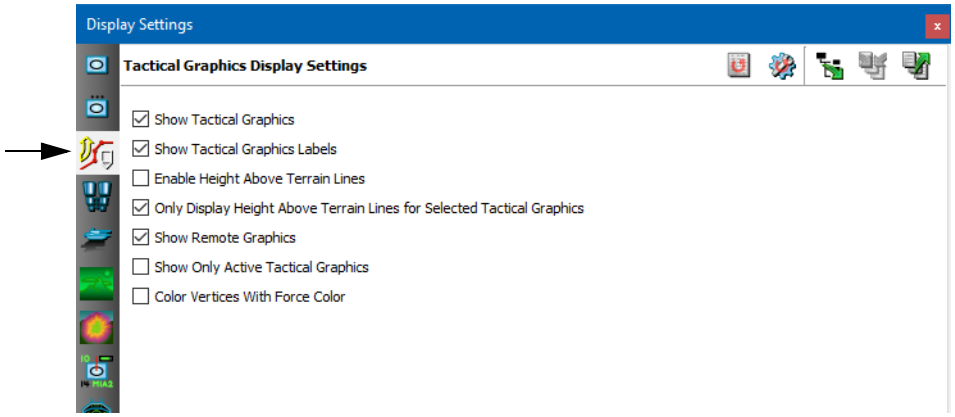


Figure 38-5. Tactical Graphics Display Settings page

3. Select or clear the Show Tactical Graphics check box.
4. To show only active tactical graphics, select Show Only Active Tactical Graphics. For more information about active tactical graphics, please see "Active and Inactive Tactical Graphics," on page 40-24.



You can also enable and disable tactical graphics by choosing **Settings** → **Tactical Graphics** or clicking the Tactical Graphics button ( ) on the Display Settings Toolbar.

Showing and Hiding Tactical Graphics on the Overlays Tab

On the Overlays tab, you can show and hide tactical graphics individually and by overlay.

To control the display of tactical graphics from the Overlays tab:

1. On the Objects List Panel, select the Overlays tab (Figure 39-2).
2. To hide all graphics on a particular overlay, clear the check box for that overlay.
3. To hide a specific tactical graphic, clear its check box.
4. To redisplay a hidden tactical graphic or overlay, select its check box.

38.7.2. Showing and Hiding Tactical Graphics Labels

In addition to displaying the outlines of tactical graphics, you can display graphics (labels) for the vertices of these objects. Vertex labels are enabled in Figure 38-3 and Figure 38-4. In Plan View mode, VR-Forces also displays a tactical graphic's name, label, and extended labels, if any. For details about displaying labels for 2D icons, please see "Displaying Labels for 2D Icons," on page 16-14.



Displaying vertex labels can affect performance.

To show or hide vertex labels:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 38-5).
3. Select or clear the Show Tactical Graphics Labels check box.



You can also enable and disable tactical graphics labels by choosing **Settings** → **Tactical Graphics Labels** or clicking the Tactical Graphics Labels button () on the Display Settings Toolbar.

38.7.3. Displaying Height-Above-Terrain Lines for Vertices

In 3D observer modes, you can display height-above-terrain (HAT) lines for the vertices of tactical graphics (Figure 38-3). These vertical lines between an object's vertices and the ground help you understand the relationship of the object to the terrain. The height for each vertex is displayed (using whatever units are configured on the Display Units Settings page).

HAT lines are shown only for tactical graphics that could be above or below the ground, such as routes, and waypoints. You can show and hide these lines and you can specify that they be shown for all objects or only for the selected objects.

Use of height above terrain lines can reduce the frame rate.

To show or hide height-above-terrain lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 38-5).
3. To display HAT lines, select the Enable Height Above Terrain Lines check box.
4. To only display lines for selected objects, select the Only Display Height Above Terrain Lines For Selected Tactical Graphics check box.



39. Overlays

This chapter explains how to create and edit overlays.

Creating and Editing Overlays 39-2
- Creating an Overlay 39-3
- Selecting Overlays 39-3
- Locking an Overlay 39-4
- Hiding the Objects on an Overlay 39-4
- Changing an Overlay's Name 39-5
- Deleting an Overlay 39-5

Saving and Loading Tactical Graphics and Overlays 39-6
- Loading Tactical Graphics and Overlays 39-6

Exporting Tactical Graphics as Shapefiles 39-7
- Importing Shapefiles into Overlays 39-7

39.1. Creating and Editing Overlays

Overlays allow you to add graphic objects to the VR-Forces window as if you were adding clear plastic overlays to a physical map and drawing on them. They provide a convenient way to organize and manage the tactical graphics and simulation objects in a simulation. You can add any number of overlays. You can rename them. You can enable or disable editing of the objects they contain.

When you create an object, if you do not explicitly assign it to an overlay, it gets placed on a default overlay. If the default overlay does not exist, it gets created. The default overlay for an object type is configured in the Simulation Object Editor.

Overlays and their contents are displayed in the Overlays View of the Objects List Panel. The Overlay Bar has a label for each overlay that lets you show and hide the objects on that overlay (Figure 39-1).

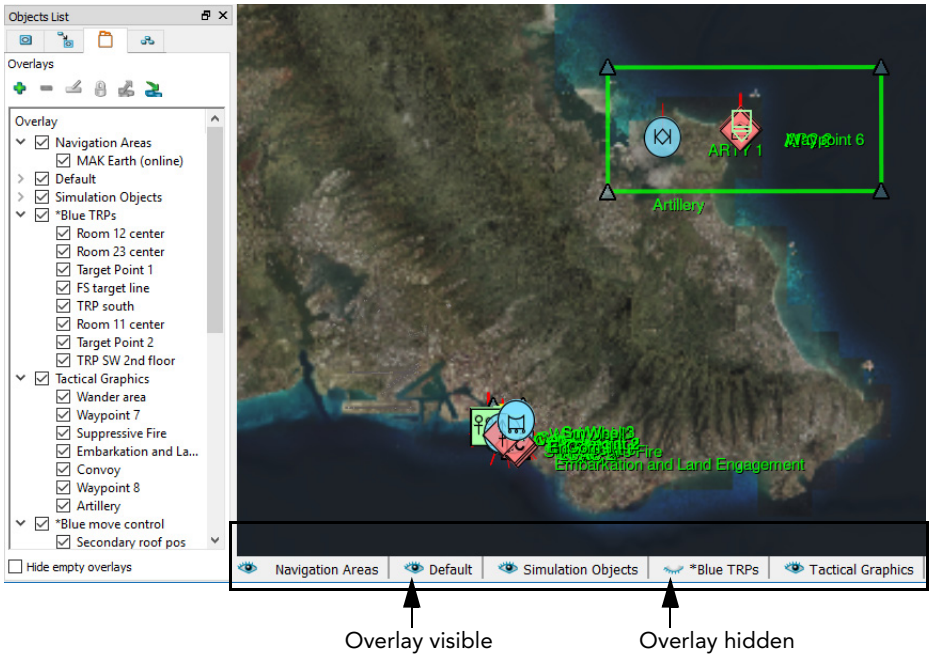


Figure 39-1. Overlays View and Overlay Bar

39.1.1. Creating an Overlay

To create an overlay:

1. On the Objects List Panel, select the Overlays View (Figure 39-2).

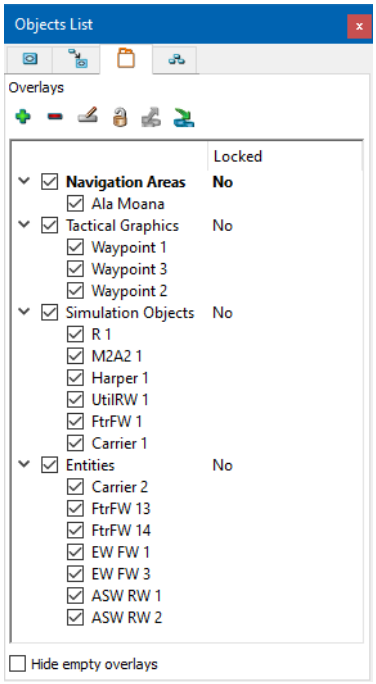


Figure 39-2. Overlays View

2. Click the Add button (). An overlay is added to the list with a default name.
3. Optionally, change the name as described in "Changing an Overlay's Name," on page 39-5.

39.1.2. Selecting Overlays

➤ To select an overlay, click its name on the Overlays View.

39.1.3. Locking an Overlay

When an overlay is locked, you cannot edit the objects on it. You can still add and delete objects.

When you save a scenario, VR-Forces saves the locked/unlocked state of each overlay. (If you lock or unlock an overlay while you are running a scenario, but do not save the scenario, the state of the overlay is not saved.

To lock or unlock an overlay:

1. On the Objects List Panel, select the Overlays View.
2. Select the overlay you want to lock or unlock.
3. Click the Lock/Unlock button (), or double-click the entry in the Locked column. It will toggle between Yes and No.

39.1.4. Hiding the Objects on an Overlay

You can hide the objects on an overlay. This may be convenient if you want to focus on a particular set of simulation objects or tactical graphics while you are editing or observing a scenario. You can hide all of the objects on an overlay or individual objects.

When an overlay is visible, the eye on the overlay tab is open. When it is hidden, the eye is closed (Figure 39-1).

When you save a scenario, VR-Forces saves the visible/hidden state of each overlay. (If you view or hide an overlay while you are running a scenario, but do not save the scenario, the state of the overlay is not saved.

To hide all of the objects on an overlay, do one of the following:

- Click the overlay's tab on the Overlay Bar.
- On the Objects List Panel, Overlays View, clear the check box for the overlay that you want to hide.

➤ To hide individual objects on an overlay, on the Objects List Panel, Overlays View, clear the check box for the object.

Hiding an Overlay by Default

You can mark overlays so that they are hidden when a scenario loads, when you add a simulation object group, or when you import a scenario.

➤ To hide an overlay by default, place an asterisk (*) at the start of the name, for example, *Tactical Graphics.

39.1.5. Changing an Overlay's Name

You can change the name of a default overlay, such as Entities or Tactical Graphics. However, the next time you create an object that gets assigned to a default overlay, unless you assign it to the renamed overlay, a new overlay with the default name gets created.

To change an overlay's name:

1. On the Objects List Panel, select the Overlays View.
2. Select the overlay that you want to rename and click the Rename button (), or slowly double-click the overlay whose name you want to change. The name becomes editable.
3. Type a new name.
4. Press **ENTER**.

39.1.6. Deleting an Overlay

To delete an overlay:

1. On the Objects List Panel, select the Overlays View.
2. Select the overlay you want to delete.

! When you delete an overlay, all of the simulation objects and tactical graphics on it are also deleted.

3. Click the Delete button ().

39.2. Saving and Loading Tactical Graphics and Overlays

If you publish a tactical graphic, it is automatically saved to the scenario objects file (*.oob*). If you do not publish a tactical graphic, it is automatically saved to the overlay file (*.ovl*). There may be cases where you want to create a set of overlays for use in multiple scenarios. You can save them to an overlay file independently of the scenario, or even save the overlays and not save the scenario at all.



When you save overlay objects, only unpublished tactical graphics get saved.

To save overlays and tactical graphics:

1. Choose **File** → **Overlay** → **Save Overlay**. The Save Overlay dialog box opens.
2. Type a name for the overlay file.
3. Click Save.

39.2.1. Loading Tactical Graphics and Overlays

You can load overlays that you have saved from a previous session.

To load overlays:

1. Choose **File** → **Overlay** → **Load Overlay**. The Load Overlay dialog box opens.
2. Select the overlay file you want to load.
3. Click Open.

When you load overlay objects, you must use the same terrain database you created them on, or a terrain database that includes the area they were created in.

39.3. Exporting Tactical Graphics as Shapefiles

You can export point, line, and area tactical graphics to shapefiles. VR-Forces exports all of the graphics on a selected overlay. This may be useful if you need feature data for a terrain, such as a road network, but there is no feature data available from external sources. You can edit the shapefile created by VR-Forces in an appropriate third-party application to add the attributes needed to fully specify the features. For example, suppose you need a road network. You would create routes in the proper locations in VR-Forces and export the overlay to a shapefile. Then you would edit the shapefile to connect the road segments, specify road width, number of lanes, and so on. You could then load the shapefile in VR-Forces or any other application that supports shapefiles.

When VR-Forces creates shapefiles from tactical graphics, it creates separate files for points, lines, and areas. It appends *P.shp*, *L.shp*, or *A.shp* to the filename you provide based on the tactical graphics types being exported. For example, if an overlay has points and lines and you export them to myFeatures, VR-Forces creates files called *myFeaturesP.shp* and *myFeaturesL.shp*.

To export tactical graphics as shapefiles:

1. On the Objects List Panel select the Overlays View.
2. Select the Overlay whose tactical graphics you want to export.
3. Choose **File** → **Overlay** → **Save Overlay as Shape Files**. The Save Overlay as Shape Files dialog box opens. (You can also right-click an Overlay on the Overlay Bar and open the context-sensitive menu.)
4. Type a name for the shapefile.
5. Click Save.

39.3.1. Importing Shapefiles into Overlays

You can import feature data into overlays. Features are placed as points, lines, or areas. This is a quick way to create tactical graphics for features such as roads, buildings, and so on. For example, if you were to load the *MAK Earth (online).mtf* terrain and import *BuildingsP.shp*, which is used in the Little Pond terrain, it places a group of waypoints at the locations of the buildings.



Importing shapefiles into an overlay is not the same thing as adding a feature layer to the terrain. For example, importing a file such as *RoadsL.shp* to an overlay creates routes that can be referenced in movement tasks. Adding it as a feature layer creates a road network that could be used for path planning.

To import feature data into an overlay:

1. Select an overlay.
2. Choose **File** → **Overlay** → **Import Shape File Onto Overlay**. The Import Shapefile Onto Overlay dialog box opens.
3. Select the shapefile you want to import.
4. Click Open.



40. Creating and Editing Tactical Graphics

This chapter explains how to create and edit tactical graphics.

Creating Tactical Graphics 40-2
- How Areas Are Drawn 40-2
- Creating Freehand Lines 40-3
- Drawing Circles and Ellipses 40-4
- Drawing Boxes 40-4
- Creating Volumetric (Extruded) Tactical Graphics 40-5
- Adding Text to an Overlay 40-5
- Assigning Tactical Graphics to an Overlay 40-6
- Displaying a Terrain Profile When You Create a Line 40-7
- Publishing Tactical Graphics 40-8

Creating a Computed Route 40-9

Creating an Intel Collection Area 40-11

Creating Nav Edges 40-12

Creating Cloud Layers 40-14

Attaching Tactical Graphics to a Simulation Object 40-16
- Attaching a Tactical Graphic Automatically 40-16

Deleting Tactical Graphics 40-17

Editing Tactical Graphics 40-17
- Editing Vertices 40-18
- Resizing Boxes and Text Objects 40-20
- Rotating an Ellipse 40-20
- Changing the Direction of a Line 40-21
- Changing the Color of a Tactical Graphic 40-21
- Using the Force Color for Vertices 40-21
- Changing Linear and Areal Style Properties 40-23

Active and Inactive Tactical Graphics 40-24

40.1. Creating Tactical Graphics

The generic procedures for creating tactical graphics are described in Chapter 14, *Creating and Placing Objects*. This section provides additional details about tactical graphic creation and placement.

40.1.1. How Areas Are Drawn

VR-Forces supports two types of areas, projected and volumetric (Figure 40-1). The points in a projected area do not have an altitude. They define an area that projects from ground level to an infinite height. Volumetric areas have a floor and a ceiling.

When you create a projected area, the points are drawn using the altitude of the first point in the area. This may result in some or all of the boundary of the area being drawn in the air. This is simply a matter of how it is drawn. The area still is projected from ground level upwards.

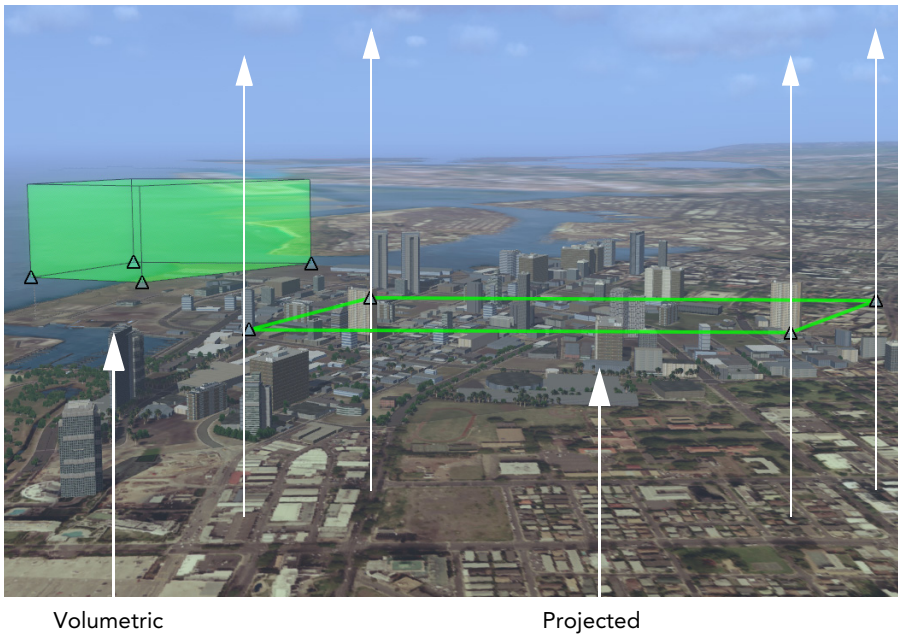


Figure 40-1. Volumetric and projected areas

40.1.2. Creating Freehand Lines

A freehand line is essentially many short line segments. After you create a freehand line you can change the sampling rate used to determine how many segments it has. The fewer the segments, the less fluid the line is.

Unlike other lines, the individual vertices of a freehand line do not have vertex markers. You cannot select them individually on the map. However, you can edit them in the Edit Freehand Line dialog box.

The Freehand Line With Arrow tactical graphic automatically puts an arrow at the end of the line. However, you can also add an arrow to a freehand line that is created without one.



If you reduce the sampling rate, resulting in fewer segments and a less fluid line, once you save the line you cannot automatically increase the number of segments to increase the fluidity of the line. You would have to add vertices by hand.

To create a freehand line:

1. Click the Tactical Graphics Palette.
2. Select Freehand line.
3. Click where you want the line to start, and while holding down the left mouse button, draw a line on the screen.
4. Release the mouse button. The Create Freehand dialog box opens.
5. Edit the properties as desired.

40.1.3. Drawing Circles and Ellipses

Simulation objects do not interact with circles and ellipses. They are strictly for use as graphics.



If you have cleared the Attach Object to Mouse check box in the Create Circle or Create Ellipse dialog box, you must still define the center point and other dimensions before the Create button is enabled. Since the object is not attached to the mouse, you will not see dynamic resizing of the object as you specify the points.

To draw a circle:

1. Click the Tactical Graphics Palette.
2. Select Circle. The cursor changes to draw mode and shows a vertex icon.
3. Click to place the center of the circle. A center point is placed on the terrain and a circle is drawn. There is now a vertex icon on the circumference of the circle.
4. Drag the mouse to set the radius of the circle.
5. Click to complete the circle.

To draw an ellipse:

1. Click the Tactical Graphics Palette.
2. Select Ellipse. The cursor changes to draw mode and shows a vertex icon.
3. Click to set the center of the ellipse. A center point is set and a small circle is displayed with a vertex indicator at the bottom of the circle.
4. Drag the mouse up or down and click to set the height of the ellipse. A vertex indicator is added to the left side of the circle.
5. Drag the mouse left and right and click to set the width of the ellipse.

40.1.4. Drawing Boxes

Boxes are constrained to be parallelograms. Simulation objects do not interact with boxes. (In other words, they are not areas.)

To draw a box:

1. Click the Tactical Graphics Palette.
2. Select Box. A box is attached to the cursor
3. Click to place the box in the approximate location that you want.
4. Double-click any vertex. The vertex changes to show it is editable.
5. Drag the vertex to reshape the box.

40.1.5. Creating Volumetric (Extruded) Tactical Graphics

You can create volumetric tactical graphics. Volumetric tactical graphics are useful for defining air corridors, airspace, and similar 3D areas. You can create the following volumetric graphics:

- Area (extruded).
- Box (extruded).
- Circle (extruded).
- Corridor (a linear object).
- Ellipse (extruded)
- Joint Engagement Zone (Altitude Limited)
- Missile Engagement Zone (Altitude Limited).

Volumetric areal objects have a floor and a ceiling above the terrain. Corridors have a width and height, which are centered on the vertices. Simulation objects can move along a corridor. (They move along the center line and ignore the height and width.)

The Entity in Area condition can check to see if an entity is within the volume of an extruded area.

To create a volumetric tactical graphic:

1. Click the Tactical Graphics Palette.
2. Select the volumetric tactical graphic you want to create.
3. Specify the vertices of the tactical graphic just like you would for a 2D tactical graphic. When you right-click to set the final vertex, the Edit dialog box should open.
4. If the Edit dialog box does not open, click its tab to open it.
5. If you are creating a route, specify the height and width. If you are creating any other extruded tactical graphic, specify the floor and ceiling (above mean sea level).
6. Click Create.

40.1.6. Adding Text to an Overlay

You can add text to an overlay. The maximum width of the text object is fixed.

To add text to an overlay:

1. Click the Tactical Graphics Palette.
2. Select Text. The text “Text” is attached to the cursor.
3. Click to set the bottom center of the text. The Create Text dialog box opens.
4. Type the text you want in the Text box.
5. Edit any other properties as desired.
6. Click Create.

40.1.7. Assigning Tactical Graphics to an Overlay

By default, when you create a tactical graphic, it gets assigned to the Tactical Graphics overlay. (You can change the default overlay in 1the Simulation Object Editor.) If no overlay exists, the overlay is created and the graphic is assigned to it. You can assign a tactical graphic to a different overlay when you create it or edit it. To assign a tactical graphic to an overlay when you create it, set this property as described in "Specifying an Object's Properties Before You Create It (Click to Locate)," on page 14-15. To move a tactical graphic to a different overlay, follow the procedure in "Moving a Tactical Graphic to a Different Overlay," on page 40-6.

Moving a Tactical Graphic to a Different Overlay

To move a tactical graphic to a different overlay:

1. Select the tactical graphic you want to move.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens.
3. If Attach Object to Mouse is selected, clear the check box.
4. In the Overlay list, select the overlay that you want the tactical graphic to move to.
5. Click Update.



You can also move a tactical graphic to a different overlay by dragging it to the new overlay on the Overlays View of the Objects List Panel.

40.1.8. Displaying a Terrain Profile When You Create a Line

You can display a terrain profile window while you create a line. You can configure VR-Forces to automatically display a terrain profile when a line is created ("Configuring Terrain Profiles," on page 56-3), or you can display the window manually.

To open a terrain profile window while you are creating a line:

1. Select the tactical graphic that you want to create as described in "Selecting the Object to Create," on page 14-4. A Create Object tab is added to the window below the palettes (Figure 14-7).
2. Click the Create Object tab. A Create Object dialog box that is appropriate for the object opens (Figure 40-2).

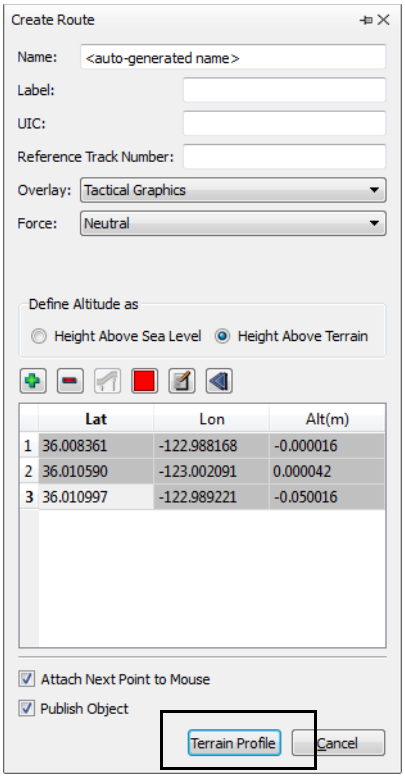


Figure 40-2. Terrain Profile button

3. Click to place the first vertex. The Terrain Profile button is activated.
4. Click Terrain Profile. A terrain profile window opens.
5. Complete the line. The terrain profile window stays open until you complete the line.

40.1.9. Publishing Tactical Graphics

By default, all tactical graphics are published. That is, they are sent out over the network and are visible to other participants in an exercise. You can choose to not publish a tactical graphic. In that case, it is visible only in the front-end on which it was created.

Published tactical graphics get saved to the scenario objects file. Unpublished tactical graphics get saved to the overlays file that gets created as part of the scenario. You can also manually save tactical graphics to an overlay file separate from the one stored with the scenario.

To publish or unpublish a tactical graphic:

1. Select the graphic you want to edit.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens.
3. To publish the object, select the Publish Object check box. To unpublish it, clear the check box.

40.2. Creating a Computed Route

VR-Forces can create routes between two points and take into account criteria such as avoiding buildings, using roads, and avoiding water. Unlike the temporary routes that VR-Forces creates for movement tasks, a computed route is a tactical graphic that is saved with the scenario and can be used by any simulation object.

When you create a computed route, you specify rules that VR-Forces should use when creating the route. VR-Forces includes preconfigured rule sets for ships and ground vehicles. You can add additional preconfigured rule sets. You can also add additional rules. The rules and rule sets are configured in *./data/simulationModelSets/<sms>/gui/computedRouteRulesTypes.xml*.

To create a computed route:

1. Choose **Create** → **Computed Route**. The Create Computed Route dialog box opens (Figure 40-3).

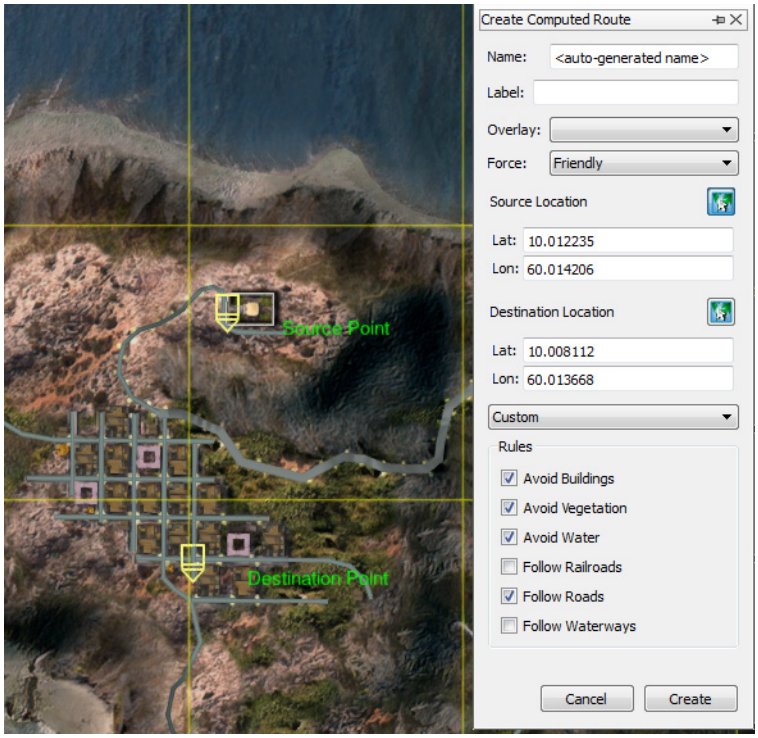


Figure 40-3. Creating a computed route

2. Optionally, specify a name, label, and overlay.
3. Select the force for the route.
4. Click to place the starting point of the route, or enter a value for the Source Location.
5. Click to place the end point of the route, or enter a value for the Destination Location.

6. Select the type of vehicle from the list. Ground vehicles and ships have a preconfigured set of rules. Select Custom to choose a different set of rules.
7. If you select Custom, select the rules that you want to be enforced for the route. Clear those you do not want to use.
8. Click Create.

40.3. Creating an Intel Collection Area

Intel collection areas modify the sensor signatures of simulation objects within their boundaries. This affects the ability of other simulation objects to detect them. Intel Collection Areas are created just like other areas. The difference is specification of area modifiers. Area modifiers are set per force. A modifier greater than 1.0 increases the sensor signature for members of that force, making them easier to detect. A modifier less than 1.0 reduces their signature, making them harder to detect.

To create an intel collection area:

1. On the Tactical Graphics Palette, select the All tag.
2. In the objects list, click Intel Collection Area.
3. Draw an area on the map as you would for any other tactical graphic area.
4. In the Create Intel Collection Area dialog box (Figure 40-4), click the Add button (+) next to the Area Modifiers label. The Choose Force dialog box opens.

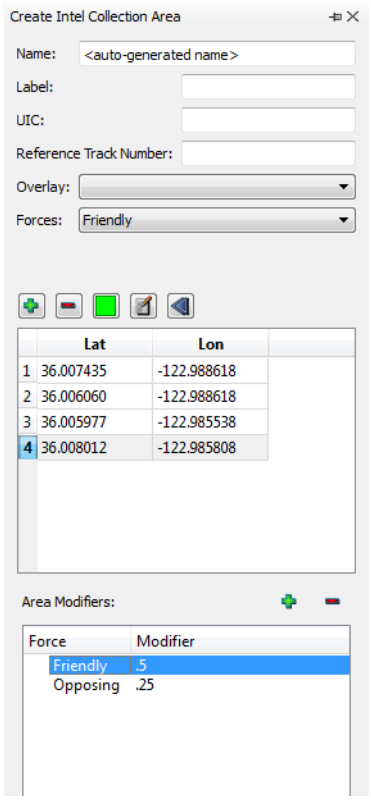


Figure 40-4. Create Intel Collection Area dialog box

5. Select the force for which you want to add a modifier.
6. Click OK. The force gets listed in the Area Modifiers window.

7. Double-click the Modifier column to make it editable.
8. Type the modifier value you want to apply to this force.
9. Optionally, add additional force entries to the list.
10. Click Create.

40.4. Creating Nav Edges

Navigation edges are tactical graphics that affect the navigation mesh in a navigation area. They permit VR-Forces to calculate paths to locations that an entity would otherwise not be able to move to.

For example, suppose that you want an entity to walk from the ground to a waypoint on top of a building that does not have any interior geometry. Normally, the task would fail with a message, "No path to destination" as in Figure 40-5.

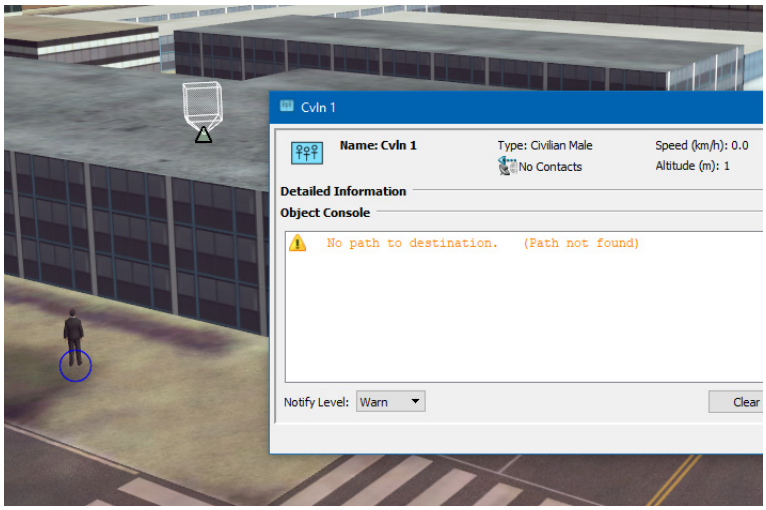


Figure 40-5. Pedestrian cannot move to waypoint

By adding a Pedestrian Nav Edge tactical graphic, the entity can plan a path to the waypoint (Figure 40-6). Use of nav edges in this way may compromise the realism of entity movement, but may be acceptable to achieve the overall goals of a simulation if the terrain is not as detailed as you would like it to be.

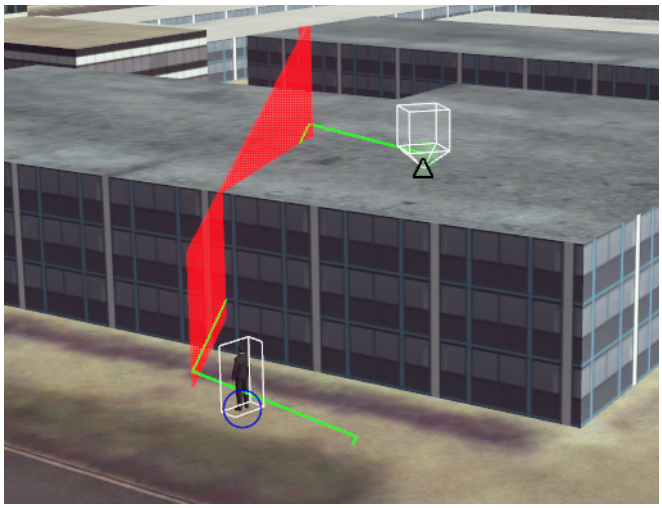


Figure 40-6. Navigation edge allows path planning

Another use for nav edges is to draw them with vertices that are underground, effectively creating tunnels that do not exist in the created terrain (Figure 40-7).

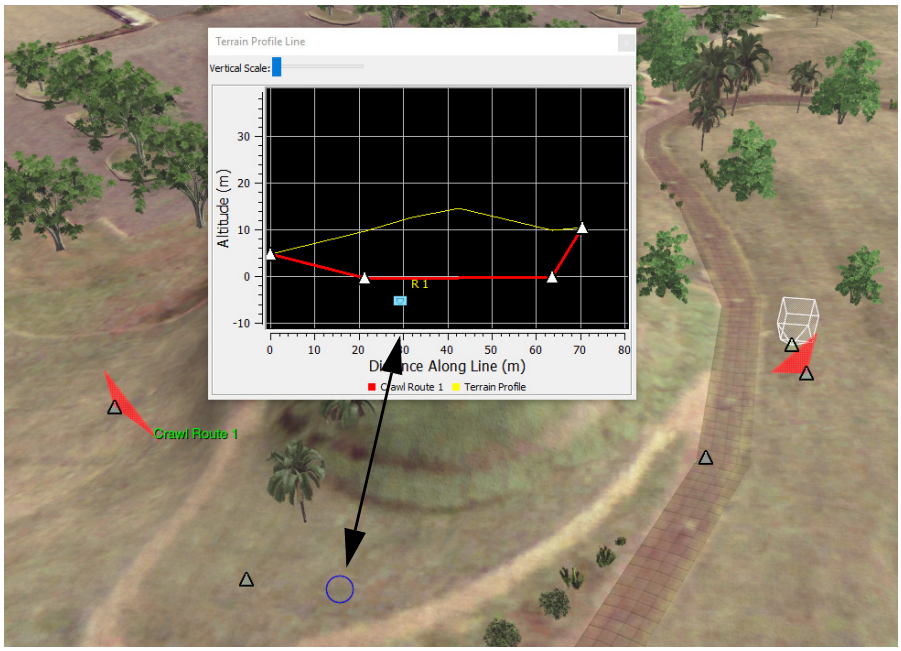


Figure 40-7. Entity follows nav edge underground

Finally, you might use a nav edge in cases where navigation data exists, but it is too restrictive. For example, the bounding volume of an entity might be slightly too large to permit it to go through a door. Adding a nav edge would allow it to go through the door.

VR-Forces has the following types of nav edge tactical graphics:

- Nav Edge. A generic nav edge.
- Pedestrian Nav Edge. A nav edge that acts like a pedestrian path. Pedestrians will prefer to use it over generic paths.
- Crawl Route (Nav Edge). When a human that has a crawl animation moves along this route, it crawls.



You can also add nav edges in a navigation profile. For details, please see "Navigation Profiles," on page 61-13.

To create a nav edge:

1. In the Tactical Graphics Palette, select a nav edge. (Filtering for "nav" will quickly list the available types of nav edges.)
2. Draw the nav edge as you would any linear tactical graphic.

40.5. Creating Cloud Layers

Cloud layers are areal tactical graphics. You can specify a floor and a ceiling, which makes them similar to volumetric areas. Cloud layers do not affect simulation objects in any way, except that you can use them in the Entity In Area condition in plans. VR-Forces publishes cloud layer details and VR-Vantage applications can use that information to display them.



Cloud layer tactical graphics are not available as cloud layers for cloud covers. Despite the similar name, they are completely separate. For information about cloud covers, please see "Creating and Editing Cloud Covers," on page 11-17.

To create a cloud layer:

1. On the Tactical Graphics Palette, select Cloud Layer. The cursor changes to show the default boundaries of the layer.
2. If it is not open, open the Create Object dialog box.
3. Specify the parameters for the object. In addition to the typical parameters, such as name and label, these objects have the following parameters:
   - Width and length. The dimensions in the X and Y planes.
   - Altitude. The height above terrain at the camera location at which the base of the object is placed. (The height of the base is automatically updated if the camera moves a large distance to recalibrate to new terrain heights).
   - Cloud Type. Select an option from the list.
   - Thickness: The height of the object above the base height. Thickness is not configurable for some cloud types.
   - Density. Density controls how much of the area of the object is filled. Specified as a number between 0.0000 and 5.0000. Density is not configurable for some cloud types.
   - Corner Radius. Lets you soften the corners of the area from the default 90 degree angle.
4. Click Create.

40.6. Attaching Tactical Graphics to a Simulation Object

You can attach tactical graphics to a simulation object so that they move with it. For example, if you draw a route on an aircraft carrier deck, you want the route to stay attached to the deck as the carrier moves. Another example is to attach a Terrain Page-in Area to an aircraft so that it pages in terrain as the aircraft flies.

You can attach a tactical graphic to an object manually or automatically (3D observer modes only).

To attach a tactical graphic to a simulation object manually:

1. Create a tactical graphics as you normally would.
2. Select the tactical graphic.
3. Choose **Objects** → **Attach Object To**. The Attach Object To dialog box opens.
4. Select the simulation object to which you want to attach the tactical graphic.
5. Click OK.

40.6.1. Attaching a Tactical Graphic Automatically

You can attach a tactical graphic to a 3D model as you create it. (This procedure uses the automatic embarkation capability. For information about embarkation, please see Chapter 17, *Embarked and Embedded Entities*.)

To attach a tactical graphic to a 3D model when you create it:

1. Select a 3D observer mode, such as Stealth mode.
2. Move the observer to view the simulation object on which you want to attach a tactical graphic.
3. On the Tactical Graphics Palette, select the tactical graphic that you want to create.
4. Move the cursor over the 3D model you want to attach to until you see a green box.
5. Click to place the first vertex (or point for point tactical graphics) on the 3D model.
6. Continue to create the tactical graphic as you normally would. The first point stays anchored to the model.

40.7. Deleting Tactical Graphics

To delete a graphical object:

1. Select the tactical graphic you want to delete.
2. Choose **Objects** → **Delete**, or press the **`Delete`** key.

40.8. Editing Tactical Graphics

You can edit a tactical graphic's properties. The properties that you can edit vary depending on the type of graphic and can include some or all of the following:

- Add, delete, and edit vertices ("Editing Vertices," on page 40-18).
- Change its name.
- Change the altitude reference ("Setting Altitude in the Create Object or Edit Object Dialog Box," on page 14-18), "Specifying the Altitude for All of the Vertices in a Route," on page 14-19.
- Change the direction of a line ("Changing the Direction of a Line," on page 40-21).
- Edit linear and areal styles:
  - Change the pen style and width.
  - Change the line style.
  - Add or remove arrows.
  - Change the fill of an area.

  ("Changing Linear and Areal Style Properties," on page 40-23.)
- Move it to a different location ("Dragging an Object to a New Location," on page 14-23).
- Move it to a different overlay ("Moving a Tactical Graphic to a Different Overlay," on page 40-6).
- Change the color ("Changing the Color of a Tactical Graphic," on page 40-21).
- Resize boxes and text objects ("Resizing Boxes and Text Objects," on page 40-20).
- Rotate an ellipse ("Rotating an Ellipse," on page 40-20).



To edit a tactical graphic, its overlay must be unlocked.

40.8.1. Editing Vertices

You can edit the vertices of routes and areas. You can edit them dynamically or in the Edit Object dialog box.

Adding a Vertex

You can add vertices to a route or area directly on the terrain or in the Edit Object dialog box.

To add a vertex to a tactical graphic dynamically:

1. Right-click the vertex next to which you want to add a new vertex. A menu is displayed.
2. Choose **Add Point After** or **Add Point Before**. A point is added midway between the selected vertex and the next or previous vertex, based on the command you chose.
3. Optionally, edit the new point as described in "Editing a Vertex," on page 40-19.

To add a vertex in the Edit Object dialog box:

1. Select the tactical graphic you want to edit.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens (Figure 40-8).

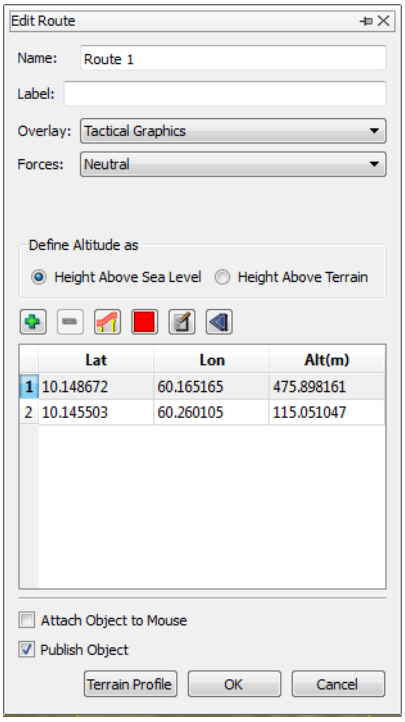


Figure 40-8. Edit Route dialog box

3. If Attach Object to Mouse is selected, clear the check box.
4. In the list of vertices, select the vertex after which you want to add the new vertex.
5. Click the Add button (+). A vertex is added midway between the selected vertex and the next vertex. If you selected the last vertex, a new vertex is added to the end of the route.
6. Optionally, edit the new point as described in "Editing a Vertex," on page 40-19.
7. Click Update.

Editing a Vertex

You can edit a vertex dynamically or in the Edit Object dialog box.

To edit a vertex dynamically:

1. Double-click the vertex.
2. Move the mouse to change the vertex's location.
3. Use **ALT**+mouse wheel or **ALT**+left mouse button+drag to change the altitude.
4. Click to place the vertex.

To edit a vertex in the Edit Object dialog box:

1. Select the tactical graphic you want to edit.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens.
3. If Attach Object to Mouse is selected, clear the check box.
4. If you are editing a waypoint, click on the terrain to specify a new location or edit the coordinates and altitude in the Edit Waypoint dialog box.

   If you are editing a phase line, route, or area, you can:

   a. Double-click the coordinate value you want to change to make it editable.
   b. Type a new value and press **ENTER** or click someplace outside of the value.

   Alternatively, you can click on the map to change the coordinates of the selected vertex. To set the altitude, you must edit the value by hand.
5. Click Update.

Deleting a Vertex

You can delete vertices directly or in the Edit Object dialog box.

To delete a vertex directly:

1. Right-click the vertex you want to delete. A menu is displayed.
2. Choose **Delete Point**. The point is deleted. (There is no confirmation prompt.)

To delete a vertex in the Edit Object dialog box:

1. Select the tactical graphic you want to edit.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens.
3. If Attach Object to Mouse is selected, clear the check box.
4. In the list of vertices, select the vertex you want to delete.
5. Click the Delete button (−).
6. Click Update.

40.8.2. Resizing Boxes and Text Objects

You cannot edit the individual vertices of a box or a text object, you can only resize them.

➤ To resize a box or text object, double-click one of its vertices to make it editable and drag it to create the new object size.

40.8.3. Rotating an Ellipse

In addition to changing the width and height of an ellipse, you can rotate it around its center point. You can undo and redo ellipse rotation.

To rotate an ellipse:

1. Select the ellipse.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens.
3. Select the Free Rotate check box.
4. Click OK.
5. Double-click one of the points on the ellipse to make it editable and drag it in a circular direction.

40.8.4. Changing the Direction of a Line

All lines have a direction. The direction affects how a simulation object moves along it and the direction that arrows face, if present. If you draw a line and then decide you want to reverse its direction, you can do so.

To change the direction of a line:

1. Select the line.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens.
3. Click the Reverse Direction button (◀).
4. Click OK.

40.8.5. Changing the Color of a Tactical Graphic

You can change the color of a tactical graphic.

To change the color of an tactical graphic:

1. Select the graphic whose color you want to change.
2. Choose **Objects** → **Edit**. The Edit Object dialog box opens.
3. Click the Color button (■). A color picker dialog box opens.
4. Click a color button or click the Custom Color button to choose a new color from the Color Picker.
5. In the Color Picker, click OK.
6. In the Edit Object dialog box, click OK.

40.8.6. Using the Force Color for Vertices

You can specify that the vertices of a tactical graphic use the force color. The color is most obvious when the tactical graphic is selected. It is muted when the graphic is not selected.

To use the force color of a tactical graphics for its vertices:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 40-9).

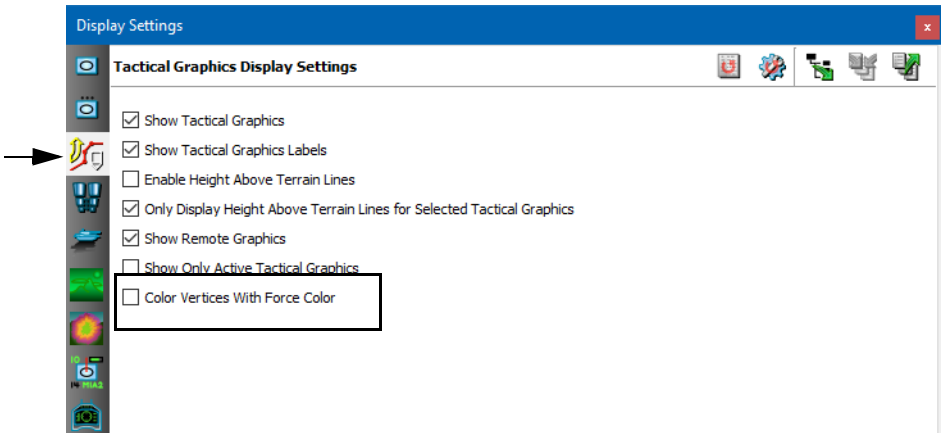


Figure 40-9. Tactical Graphics Display Settings page

3. Select the Color Vertices With Force Color check box.
4. Click OK.

40.8.7. Changing Linear and Areal Style Properties

Lines and areas have a variety of properties that you can set in the Edit Style dialog box. The procedure is essentially the same for all of these properties. The properties you might be able to change are as follows:

- Line width. The width of the lines that make up a tactical graphic, either the actual line or the boundary of an area.
- Pen style. How the line or outline is drawn – solid line, no line, or as dashed and dotted lines. This property does not apply to lines that use a texture, such as Line of Contact.
- Fill style. The fill style for areal objects. Fill styles include degrees of solid fill and various types of line and grid fills.
- Line style. The different types of special lines, such as Main Axis.
- Arrow. You can add arrows to the end of a line, at the end and beginning segment, or at the end of each segment of the line. Figure 40-10 illustrates the arrow styles.

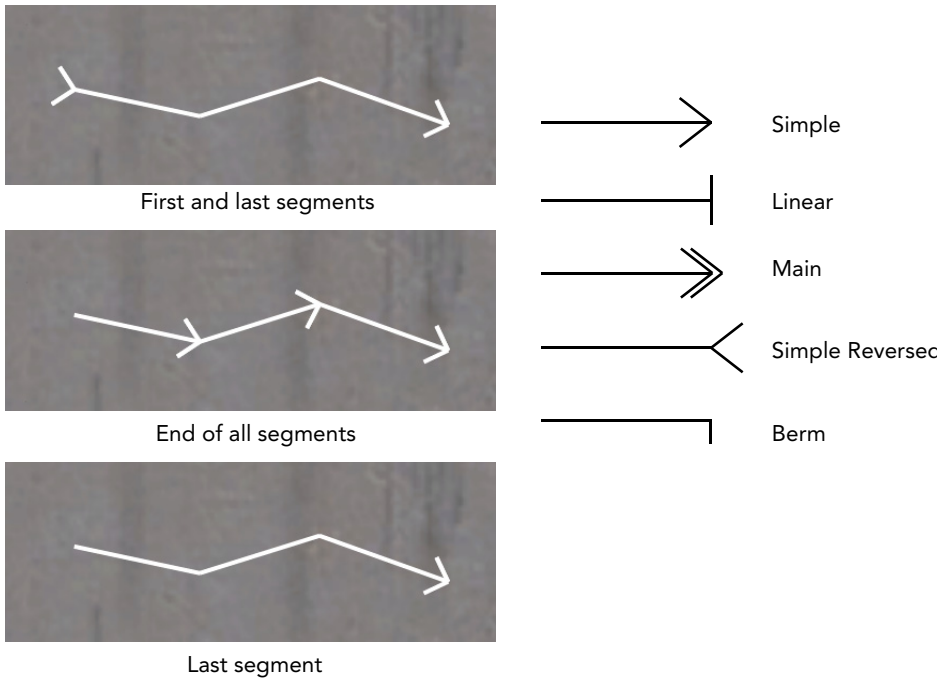


Figure 40-10. Arrow styles

To change a style property:

1. Select the graphic you want to edit.
2. Choose **Objects** → **Edit**. The Edit object dialog box opens.
3. Click the Edit Style button (). The Edit Style dialog box opens (Figure 40-11). It lists the style options that you can edit for this tactical graphic.

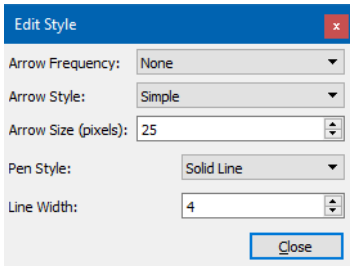


Figure 40-11. Edit Style dialog box

4. Change the properties by selecting options from the appropriate list or typing new values.
5. Click OK.

40.9. Active and Inactive Tactical Graphics

You can activate (display) and deactivate (hide) all published tactical graphics. You can set a tactical graphic's state when you create it or edit it, and using a set data request. You can test for a tactical graphic's state with the Tactical Graphic Active condition. You can set a tactical graphic's state based on simulation time or time of day.

In order for inactive tactical graphics to be hidden, you must select Show Only Active Environmentals on the Display Settings dialog box, Tactical Graphics Settings page or click the Show Only Active Tactical Graphics button ( )on the Display Settings Toolber.

In most cases, deactivating a tactical graphic is a GUI effect. The graphic continues to be simulated and published over the network. Simulation objects continue to interact with it. However, if you set an indirect fire event area to be inactive, and you specify that the detonation start time is Wait for Activation, the indirect fire event does not begin until the area is activated.

For details about the Activation set data request, please see "Activation," on page 41-2 and "Tactical Graphic Active," on page 36-13.

To set the activation state of a tactical graphic:

1. Select the tactical graphic you want to edit. (If you are creating a new tactical graphic, open the Create object dialog box and skip to step 3.)
2. Choose **Objects** → **Edit**. The Edit object dialog box opens.
3. Click the Set Activation button ( ). The Set Activation dialog box opens (Figure 40-12).

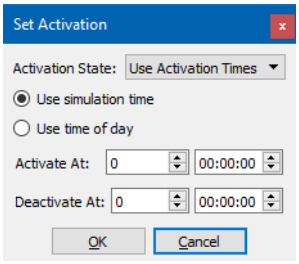


Figure 40-12. Set Activation

4. Select one of the following options from the Activation State list:
   - Active. The tactical graphic is visible.
   - Inactive. The tactical graphic is not visible if Show Only Active Environmentals is selected on the Display Settings dialog box, Tactical Graphics Display Settings page.
   - Use Activation Times. The tactical graphic is active during the times specified in the dialog box.
5. If you selected Active or Inactive, click OK. If you selected Use Activation Times:
   a. Select Use Simulation Time or Use Time of Day.
   b. In the Activate At boxes, specify the time you want the tactical graphic to be visible.
   c. In the Deactivate At boxes, specify the time you want the tactical graphic to be hidden.
6. Click OK.



41. Setting the State of Tactical Graphics

You can use set data requests to change the state of a tactical graphic.

Set Data Requests for Tactical Graphics 41-2
Activation 41-2
Allow Crossing 41-3
Arrowhead Style 41-3
Color 41-4
Fill Style 41-4
Force 34-18
Line Width 41-4
Notify Level 41-5
Pen Style 41-5

41.1. Set Data Requests for Tactical Graphics

Set data requests for tactical graphics provide an alternative to using Edit dialog boxes to change the state of a graphic. The advantage of using set data requests is that you can make changes to tactical graphics from plans and in scripted tasks or scripted sets. You can also quickly make changes from the Set menu without needing to open an Edit dialog box.

41.2. Activation

The Activation set data request lets you activate or deactivate tactical graphics. In most cases, this is strictly a visual effect; activated tactical graphics are visible; inactive tactical graphics are not visible. However, in the following cases, setting a tactical graphic to be inactive affects its functioning:

- If you set an indirect fire event area to be inactive, the indirect fire event does not start until the area is activated.
- If you set a spawn point to be inactive, it does not spawn entities.

Inactive tactical graphics continue to be published and simulation objects interact with them normally. You can use the Tactical Graphic Active condition to test the state of a tactical graphic.

For more information, please see "Active and Inactive Tactical Graphics," on page 40-24 and "Tactical Graphic Active," on page 36-13.

To activate or deactivate a tactical graphic:

1. Select the tactical graphic.
2. Choose **Set** → **Disposition** → **Activation**. The Activation dialog box opens (Figure 41-1).

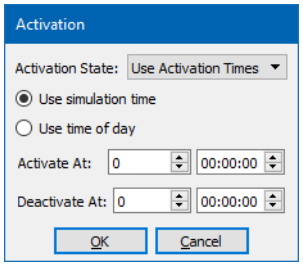


Figure 41-1. Activation

3. Select one of the following options from the Activation State list:
   - Active. The tactical graphic is visible.
   - Inactive. The tactical graphic is not visible if Show Only Active Environmentals is selected on the Display Settings dialog box, Tactical Graphics Display Settings page.
   - Use Activation Times. The tactical graphic is active during the times specified in the dialog box.

4. If you selected Active or Inactive, click OK. If you selected Use Activation Times:
   a. Select Use Simulation Time or Use Time of Day.
   b. In the Activate At boxes, specify the time you want the tactical graphic to be visible.
   c. In the Deactivate At boxes, specify the time you want the tactical graphic to be hidden.
5. Click OK.

41.3. Allow Crossing

The Allow Crossing set data request allows human entities to walk across a crosswalk. When human characters plan a path that includes a crosswalk, when they arrive at the crosswalk, if crossing is allowed, they can cross the street. If it is not allowed, they stop and wait.

To set a crosswalk to be enabled or disabled:

1. Select a crosswalk tactical graphic.
2. Choose **Set** → **Allow Crossing**. The Allow Crossing dialog box opens.
3. Select or clear the Allow Crossing check box.
4. Click OK.

41.4. Arrowhead Style

The Arrowhead Style set data request lets you set the type of arrowhead, location, and width for linear tactical graphics. For a description of arrowhead styles, please see "Changing Linear and Areal Style Properties," on page 40-23.

To set the arrowhead style:

1. Select a linear tactical graphic.
2. Choose **Set** → **Appearance** → **Arrowhead Style**. The Arrowhead Style dialog box opens.
3. Select an option from the Arrow Style list.
4. Select an option from the Arrow Placement list.
5. Specify the width, in pixels, of the arrow.
6. Click OK.

41.5. Color

The Color set data request lets you change the color of linear and areal tactical graphics.

To set the color of a linear or areal tactical graphic:

1. Select a tactical graphic.
2. Choose **Set** → **Appearance** → **Color**. The Color dialog box opens.
3. Select the color you want to use for this tactical graphic.
4. Click OK.

41.6. Fill Style

The Fill Style set data request lets you specify the fill for areal tactical graphics. The fill can be horizontal, vertical, or diagonal lines, or various degrees of opacity.

To set the fill style of an areal tactical graphic:

1. Select a tactical graphic.
2. Choose **Set** → **Appearance** → **Fill Style**. The Fill Style dialog box opens.
3. Select an option from the Fill Style list.
4. Click OK.

41.7. Force

You can change the force of a tactical graphic. For details, please see "Force," on page 34-18.

41.8. Line Width

The Line Width set data request lets you set the width of linear and areal tactical graphics.

To set the color of a linear or areal tactical graphic:

1. Select a tactical graphic.
2. Choose **Set** → **Appearance** → **Line Width**. The Line Width dialog box opens.
3. Type a value, in pixels, in the Line Width box.
4. Click OK.

41.9. Notify Level

You can set the notification level for console messages without opening the Information dialog box. For details, please see "Notify Level," on page 34-28.

41.10. Pen Style

The Pen Style set data request lets you specify the type of line used to draw linear and areal tactical graphics. This includes solid lines and varies combinations of dashes and dots.

To set the pen style of a linear or areal tactical graphics:

1. Select a tactical graphic.
2. Choose **Set** → **Appearance** → **Pen Style**. The Pen Style dialog box opens.
3. Select an option from the Pen Style list.
4. Click OK.



42. Remote Graphics

Remote graphics are graphics that are drawn onto the scene by remote programs that use the VR-Vantage Remote Draw API. This chapter explains how to display them in VR-Forces.

Displaying Remote Graphics 42-2
Viewing a List of Remote Graphics..... 42-4

42.1. Displaying Remote Graphics

VR-Forces can display graphics sent from other applications using the VR-Vantage Remote Draw API (Figure 42-1).

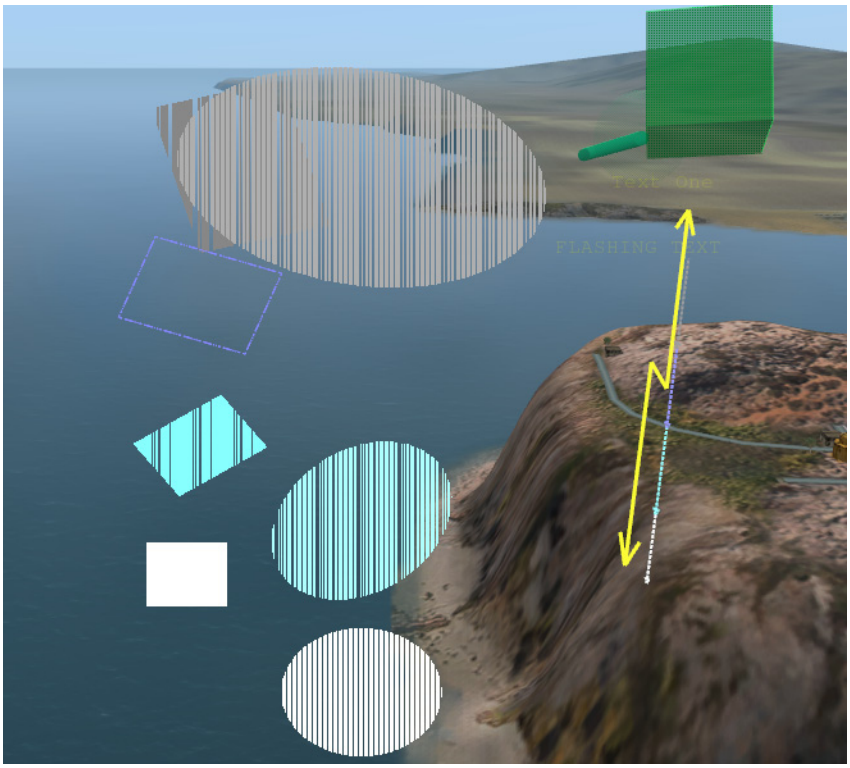


Figure 42-1. Remote graphics

To enable or disable the display of remote graphics:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 42-2).

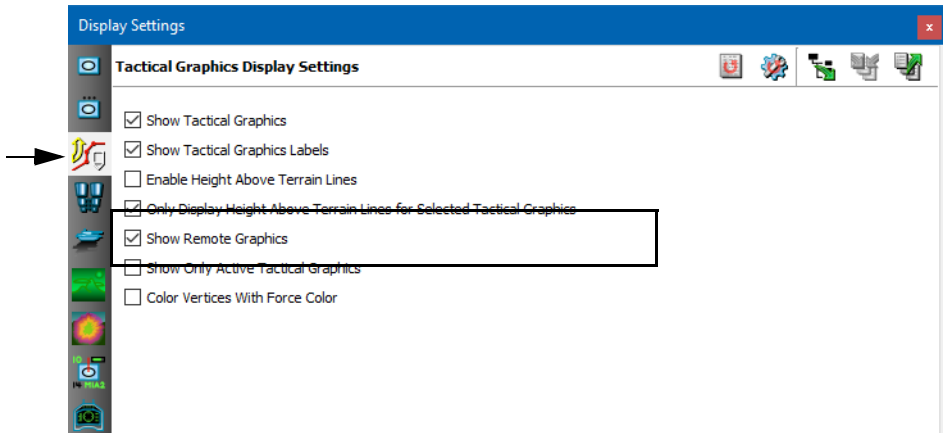


Figure 42-2. Tactical Graphics Display Settings page

3. Select or clear the Show Remote Graphics check box.



You can also enable and disable remote graphics by choosing **Settings** → **Remote Graphics**.

42.1.1. Viewing a List of Remote Graphics

Remote graphics can be organized into sets. There is no fixed organization. Labeling of sets is entirely at the discretion of the programmer who develops the application that generates the graphics. The Remote Graphics Panel displays remote graphics in a hierarchy by set.

The VR-Vantage Remote Draw API lets developers send data about objects as attribute:value pairs, for example Speed:30 KM. The attributes are organized by type using attribute group templates. You can enable and disable the display of attribute information by group and by individual attribute. Figure 42-3 illustrates the attributes for the Entity and Packet attribute group templates displayed in Figure 42-4.

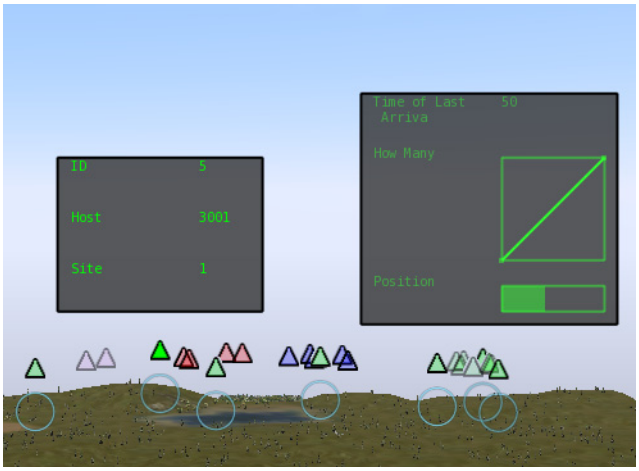


Figure 42-3. Remote graphics attribute group display

To view a list of remote graphics:

1. Choose **View** → **Remote Graphics Panel**. The Remote Graphics Panel is displayed (Figure 42-4). The Remote Graphics Sets tab lists remote graphics by set.

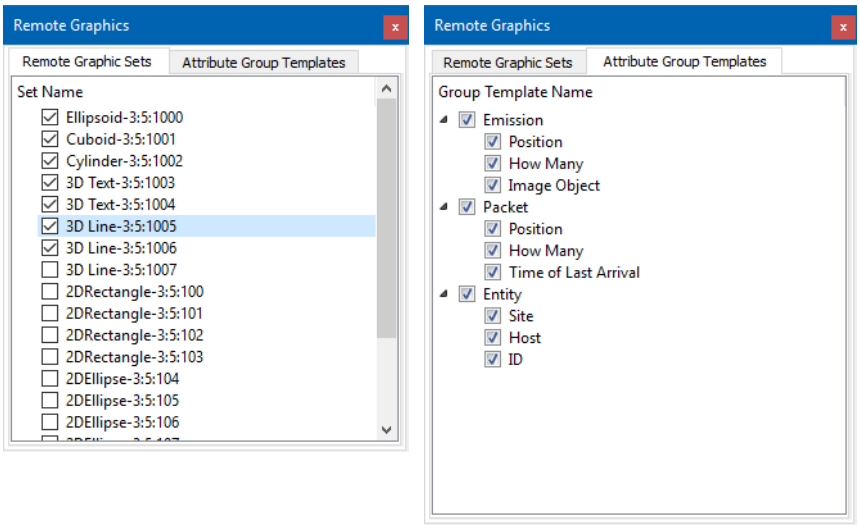


Figure 42-4. Remote Graphics Panel

2. To hide or display a remote graphic set or individual graphic, check or clear its check box in the Remote Graphics Sets list.
3. To view Attribute Group Templates, select the Attribute Group Templates tab. To see the attribute display, mouse over an object for which display data is available.



IX. Visual and Audio Effects

VR-Forces has many visual effects that add to your understanding of the interactions between simulation objects. These include fire and detonation lines, sensor contact lines, communications lines, and emitter volumes. You can test for line-of-sight with the intervisibility feature. You can also display many different lighting effects.



43. Displaying Tactical Information and Effects

This chapter describes how to manage special visual effects that can be applied to simulation objects.



Many of the effects described in this chapter apply to individual entities in Stealth mode. By default, simulation objects in aggregate-level scenarios do not use 3D models. Therefore, effects such as trailing effects, smoke clouds, and fire and detonation effects are not supported.

Displaying Trailing and Decal Effects (3D Only) 43-3
Specifying Models for Trailing Effects and Decal Effects 43-4
Displaying Track Histories 43-5
Configuring Track Histories 43-6
Displaying a Terrain Profile for a Track History 43-7
Viewing Fire and Detonation Effects 43-8
Viewing Fire and Detonation Lines 43-9
Viewing Unit Combat Graphics 43-10
Visualizing Tasks 43-12
Displaying Task Visualizations for an Entity 43-13
Pinning Task Visualizations for an Entity 43-13
Displaying Sensor Contact Lines 43-14
Enabling the Display of Sensor Contact Lines 43-14
Specifying which Simulation Objects Display Sensor Contact Lines 43-15
Configuring Sensor Contact Lines 43-16
Displaying Threat and Sensor Range Rings 43-16
Enabling and Disabling the Display of Range Rings 43-18
Pinning Range Rings for a Simulation Object 43-19

Configuring the Display of Range Rings 43-19
Displaying Laser Designators 43-22
Displaying Supply Lines 43-23
Displaying Tactical Smoke 43-23
Displaying Electromagnetic Emissions 43-24
Displaying Radio Communication Lines 43-25
Enabling Radio Communications Graphics 43-26
Configuring the Lifetime and Display Mode 43-27
Configuring the Color of Radio Communication Lines 43-28

43.1. Displaying Trailing and Decal Effects (3D Only)

VR-Forces supports trailing effects such as contrails for missiles and aircraft and dust clouds for vehicles. It also supports decal effects, such as tire tracks, footprints, and wakes. In essence, trailing effects are effects that are dispersed through the air, while decal effects are effects that are printed on the ground.

Trailing effects and decal effects are properties of an observer mode. You can enable and disable them on a per-observer basis.

Figure 43-1 illustrates some trailing effects.



Figure 43-1. Tank tracks, dust cloud

To enable or disable trailing effects or decal effects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).

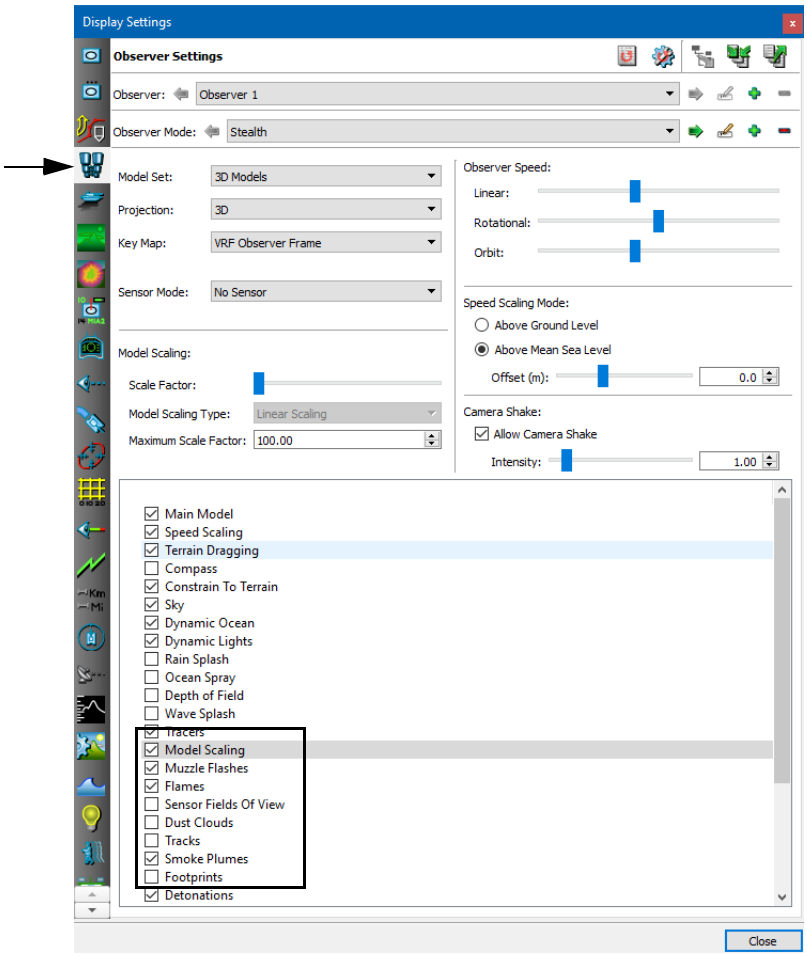


Figure 43-2. Display Settings dialog box, Observer Settings

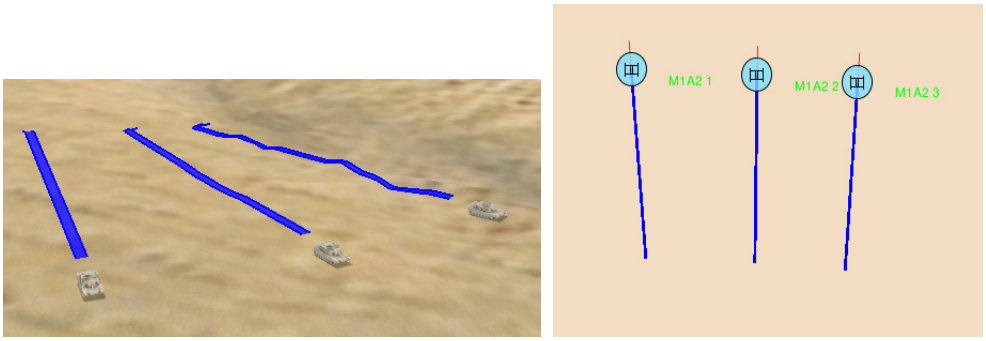
3. Select or clear the check box for each effect that you want to change.

43.1.1. Specifying Models for Trailing Effects and Decal Effects

Trailing effects and decal effects are configured through model definitions. (For the purposes of configuration, trailing effects are categorized as decals.) Each effect must have a model definition that uses a decal schema. Then, in an entity's entity definition, you configure a visualizer for the effect. For details about creating model definitions, please see "Creating and Editing Model Definitions," on page 87-2.

43.2. Displaying Track Histories

Track histories allow you to track the paths of simulation objects during an exercise, including the path of munitions as they travel from the shooter to the target. When track histories are enabled, each simulation object leaves a track ribbon. The ribbon is colored based on the simulation object's force. Figure 43-3 illustrates track histories.



3D 2D

Figure 43-3. Track history

Track histories are a property of an observer mode. You can enable and disable them on a per-observer basis.

Track histories have a significant performance and memory cost. For simulations with large simulation object counts, turning track histories off can improve performance and memory usage.

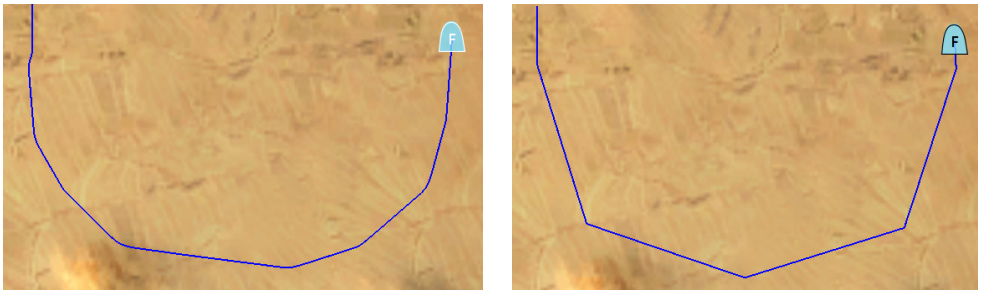
To enable or disable display of all track histories:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
3. Select the observer mode for which you want to change the Track Histories setting.
4. Select or clear the Track Histories check box.

You can also enable or disable track histories by choosing **Observer** → **Track Histories** or clicking the Track Histories button ( ) on the Observer Settings Toolbar.

43.2.1. Configuring Track Histories

You can configure how long track history ribbons are drawn. A track history is composed of a series of segments, joined together to form a line. You can configure how long each segment is and how many segments at a time can be displayed. Drawing shorter segments improves the accuracy of the track history, but can degrade performance. Increasing the maximum number of segments provides a longer history, but can clutter up the screen if there are many simulation objects and can also affect performance. Figure 43-4 illustrates the visual result of drastically changing the segment length. The left track history uses the default segment length of 0.5 meters. The right track history has a segment length of 5000 meters.



0.5 M segment length | 5000 M segment length

Figure 43-4. Track history segment length

When a track history reaches the maximum number of segments configured, it starts dropping segments from the beginning of the history (the oldest segments).

You can specify that track histories have an infinite length. In this case they never drop segments.

! If you specify infinite length for track histories, it is possible that you could eventually use up all the memory in your computer and the program would crash.

To configure track histories:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Track History Settings page (Figure 43-5).

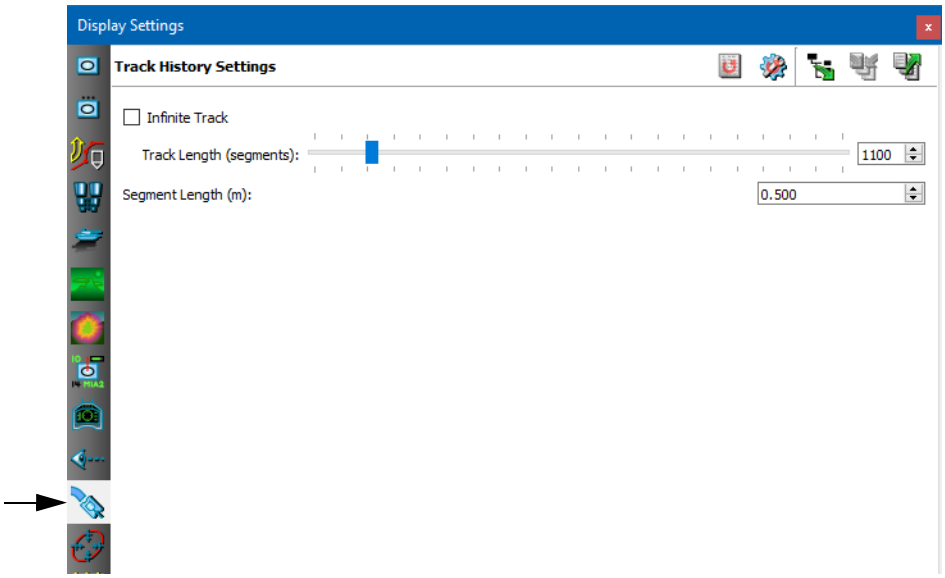


Figure 43-5. Track History Settings page

3. If you want track histories to have an indefinite length, select the Infinite Track check box.
4. To change the number of segments that can be displayed for a track history, adjust the Track Length slider, or enter a value in the box.
5. To specify the length of each track history segment, enter a value in the Segment Length box.

43.2.2. Displaying a Terrain Profile for a Track History

You can display a terrain profile for a track history. (For complete information about terrain profiles, please see Chapter 56, *Using Terrain Profiles*.)



The track history terrain profile does not display the track history that gets drawn in the window when you enable track histories. It creates an independent track that starts when you open the terrain profile window.

To display a terrain profile for a track history:

1. Select the simulation object whose track history terrain profile you want to see.
2. Choose **Objects** → **Terrain Profile** → **Track History Profile**. The Terrain Profile dialog box opens. It updates as the simulation object moves.

43.3. Viewing Fire and Detonation Effects

VR-Forces can display the following graphic effects when munitions are fired and detonated:

- Flaming effects.
- Smoke plumes.
- Detonations.
- Muzzle flashes.
- Tracers. Tracers are displayed if the munition in a fire interaction is a tracer munition and the munition is mapped to a tracer model. Tracers are not supported in 2D observer modes. (Entities that have weapons that support tracer munitions must enable tracer use for tracers to be displayed. For details about enabling tracer use, please see "Tracer Use," on page 34-44.)

To enable fire and detonation effects:

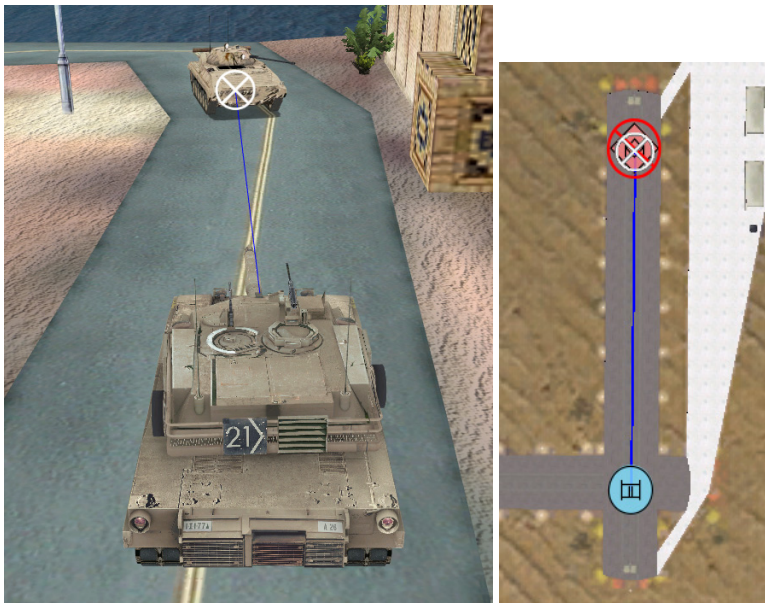
1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
3. Select or clear the appropriate check boxes.



You can also enable or disable display of a fire or detonation effect by choosing **Observer** → *effect*, where *effect* is Flames, Smoke Plumes, Detonations, Muzzle Flashes, or Tracers.

43.4. Viewing Fire and Detonation Lines

If fire and detonation lines are enabled, when an entity fires a munition, a line is drawn between the entity and the location of the detonation. The line is colored for the force of the shooter (Figure 43-6).



3D 2D

Figure 43-6. Fire line and detonation

A detonation is identified by a circle with an X through it. The detonation color indicates the result of the detonation, as follows:

- Entity impact – white.
- Ground impact – gray.
- Airburst – black.
- Other – white.

➤ To enable or disable display of fire and detonation lines, choose **Settings** → **Fire / Detonate Lines**.

43.5. Viewing Unit Combat Graphics

When simulation objects in aggregate-level scenarios engage in combat, they display two types of graphics, attack indicators, and attack animations. Attack indicators are triangular areas that show the direction a unit is attacking. They are colored with the force of the attacker. Attack animations show where a simulation object is being attacked. The footprint sector where the attack is taking place blinks from green to white.

The graphics persist until the attack is over.

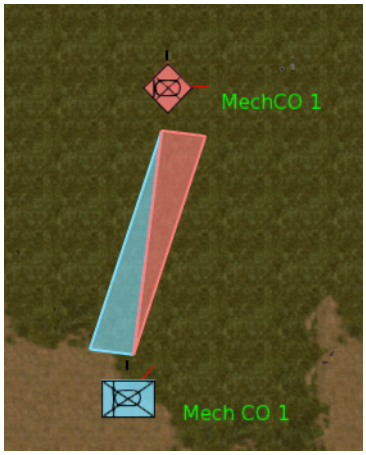


Figure 43-7. Attack indicators

To enable or disable unit combat graphics:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 43-8).

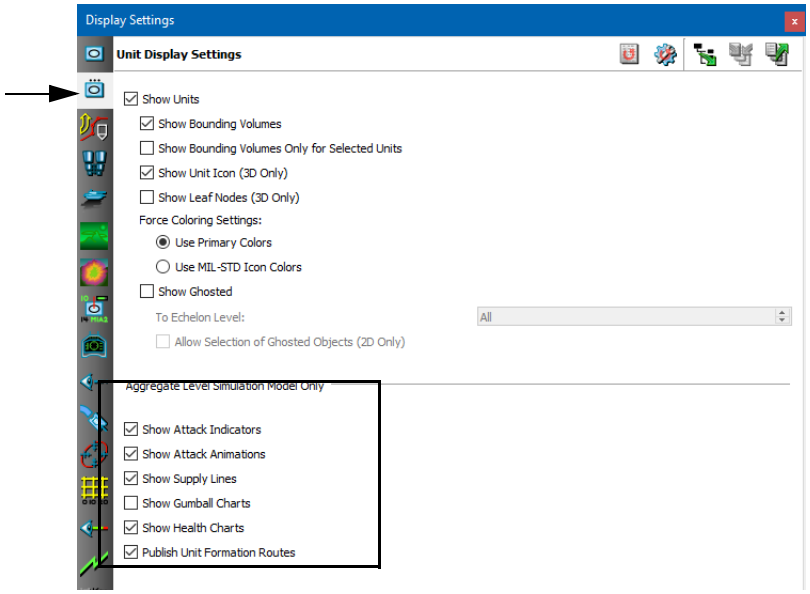


Figure 43-8. Unit Display Settings page

3. To enable or disable attack indicators, select or clear the Show Attack Indicators check box.
4. To enable or disable attack animations, select or clear the Show Attack Animations check box.

43.6. Visualizing Tasks

VR-Forces can display graphics that help to show the task a simulation object is executing. Figure 43-9 illustrates task visualization lines in Plan View mode and Stealth mode that show an entity moving to a location.

Task visualizations are not available for some tasks.

Task visualizations are supported only for individual entities, not for units.

To maintain adequate front-end performance, task visualization have the following limits:

- No more than 10 simulation objects can show task visualizations at the same time.
- The maximum number of segments in a task visualization line is 80. If a simulation object is following a route or series of points that requires more than 80 segments to visualize, VR-Forces samples the points in the line. The resulting visualization line will not be as precise as the actual path the object is following, but still provides a general sense of its intent.

If either of these conditions is met, VR-Forces prints messages if the notification level is set to verbose.

You can specify that tasks be visualized for the selected simulation object. You can also specify that tasks be visualized on a per simulation object basis. The per-simulation object setting is saved as part of the scenario.

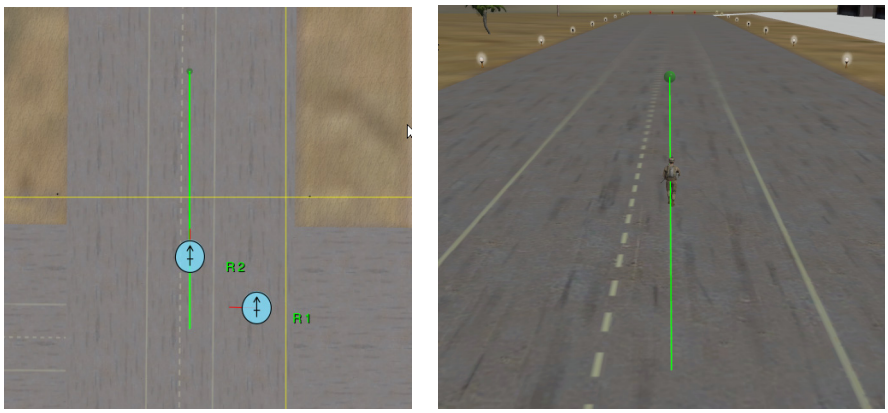


Figure 43-9. Task visualization

To enable or disable task visualization lines:

➤ Choose **Observer** → **Task Visualizations** or click the Task Visualizations button ( ) on the Observer Toolbar.

43.6.1. Displaying Task Visualizations for an Entity

If task visualization is enabled, by default they are displayed for the selected entity. You can disable automatic display of task visualizations for the selected entity. If this setting is disabled, you can still show a task visualization by pinning it to an entity, as described in "Pinning Task Visualizations for an Entity," on page 43-13.

To specify that the selected simulation object show task visualization lines:

1. Enable task visualizations.
2. Select an entity.
3. Choose **Settings** → **Display**. The Display Settings dialog box opens.
4. Select the Entity Display Settings page (Figure 16-10).
5. Select the Show Task Visualizations for Selected Entity check box. Clear the check box to disable them.

43.6.2. Pinning Task Visualizations for an Entity

By default, when task visualizations are enabled, they are only displayed for the selected entity. You can pin task visualization to an entity so that they are displayed even if it is not selected. This is the only way to have multiple entities visualize their tasks at the same time. You can pin a task visualization and display it even if the Show Task Visualization for Selected Entity check box on the Entity Display Settings page is cleared.

To pin task visualizations for a simulation object:

1. Select the simulation object for which you want to visualize tasks.
2. Choose **Objects** → **Decorations** → **Pin Task Visualizations**.

43.7. Displaying Sensor Contact Lines

You can display lines between a simulation object and the objects it detects with its sensors (Figure 43-10). Display of sensor contact lines requires two levels of control. Display of sensor contact lines is an observer-specific setting. They must be enabled in the observer mode to be displayed.

When display of sensor contact lines is enabled, you can specify that sensor contact lines be shown for specific simulation objects, for the selected simulation object, or for both. If you enable display of sensor contact lines for a specific simulation object, the setting is saved with the scenario.

You can also configure the width and color of sensor contact lines.

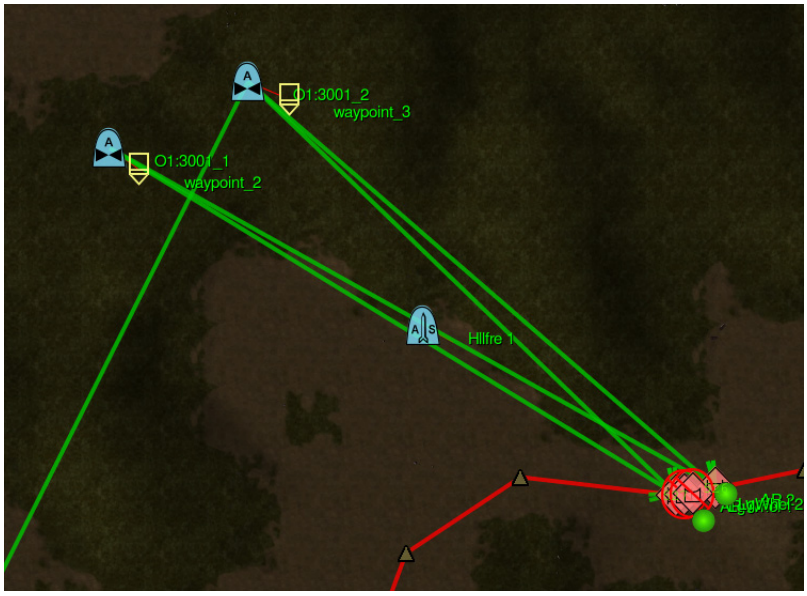


Figure 43-10. Sensor contact lines

43.7.1. Enabling the Display of Sensor Contact Lines

Regardless of other settings, you must enable the display of sensor contact lines in the observer to see them.

To enable the display of sensor contact lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
3. In the settings list, select Sensor Contact Lines.



You can also enable or disable sensor contact lines by choosing **Observer** → **Sensor Contact Lines** or clicking the Sensor Contact Lines button () on the Observer Settings Toolbar.

43.7.2. Specifying which Simulation Objects Display Sensor Contact Lines

Assuming that the display of sensor contact lines is enabled, you can specify that particular simulation objects always display them and you can specify that simulation objects display sensor contact lines if they are selected.

To display sensor contact lines for a simulation object whether it is selected or not:

1. Select the simulation objects for which you want to enable sensor contact lines.
2. Choose **Objects** → **Decorations** → **Pin Sensor Contacts**.



This setting is saved with the scenario.

To enable display of sensor contact lines by the selected simulation objects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Sensor Contact Line Settings page (Figure 43-11).

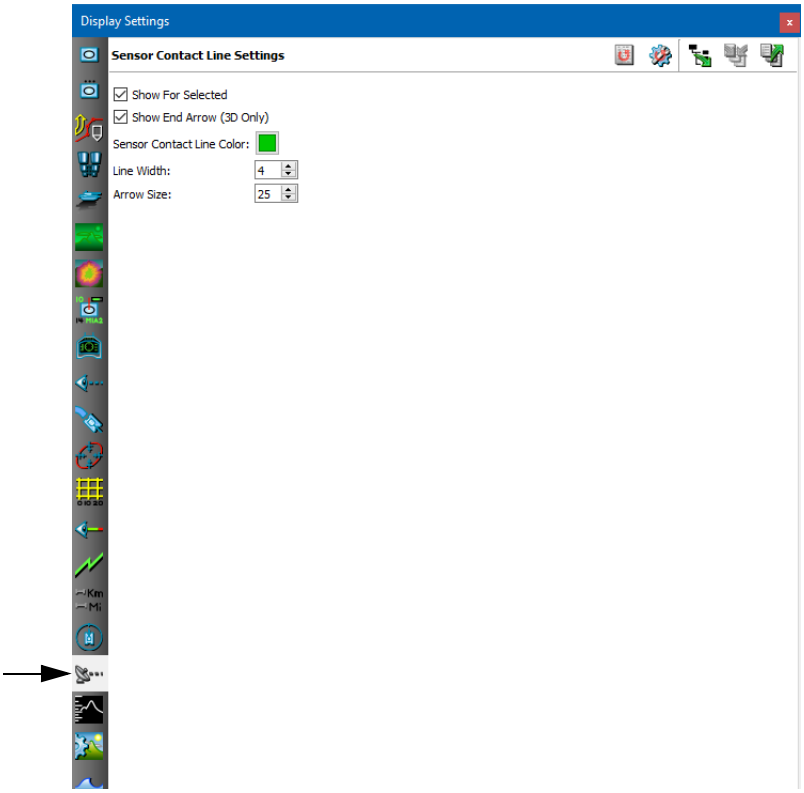


Figure 43-11. Sensor Contact Settings page

3. Select or clear Show For Selected.



This setting is saved with the GUI settings.

43.7.3. Configuring Sensor Contact Lines

You can set the color, line width, and arrow size of sensor contact lines.

To configure sensor contact lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Sensor Contact Settings page (Figure 43-11).
3. Change the values as desired.

43.8. Displaying Threat and Sensor Range Rings

You can display a simulation object's threat range – the area within which its armaments are effective. The threat range is displayed as a circle, with a line indicating the armament and its direction of fire (Figure 43-12). A ring is displayed for each weapon system that the entity has. For example, in Figure 43-12, the M1A2 entity has an inner ring for its 62mm machine gun and an outer ring for its 120mm main gun.

You can also display range rings for a simulation object's sensors.



2D 3D

Figure 43-12. Range rings (entity-level scenario)

Simulation objects in aggregate-level scenarios display range rings that show their combat effectiveness and a footprint that represents the physical area that the unit occupies (Figure 43-13). Unit range rings have breaks to show the front, rear, left flank, and right flank sectors.



On small databases, a simulation object's range ring may extend beyond the edge of the terrain display.

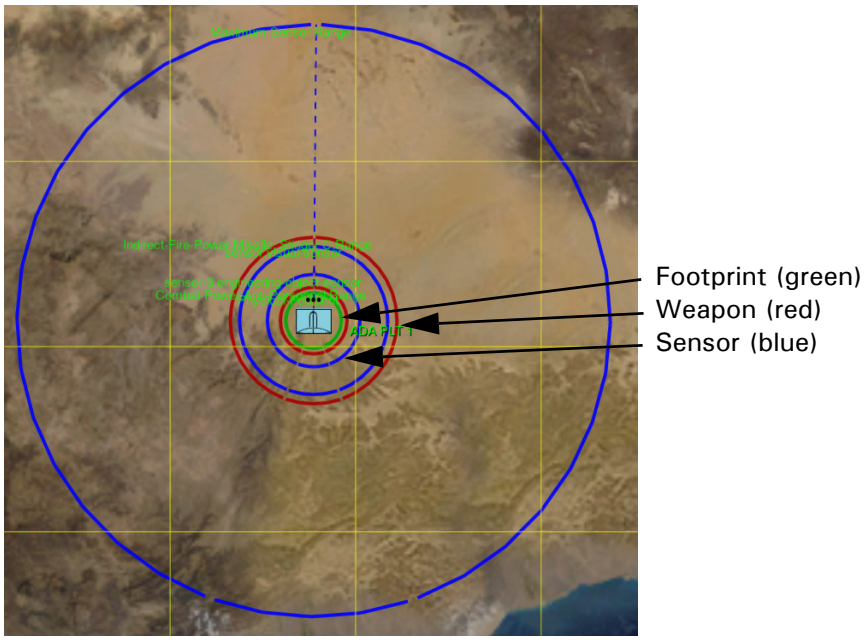


Figure 43-13. Range rings (aggregate-level scenario)

You enable range rings globally. They are an observer-specific option. You display range rings on a per-entity basis. You can globally configure the color of rings, their transparency, and the display of labels. In 3D views, you can specify whether or not to display a range sphere (Figure 43-14).

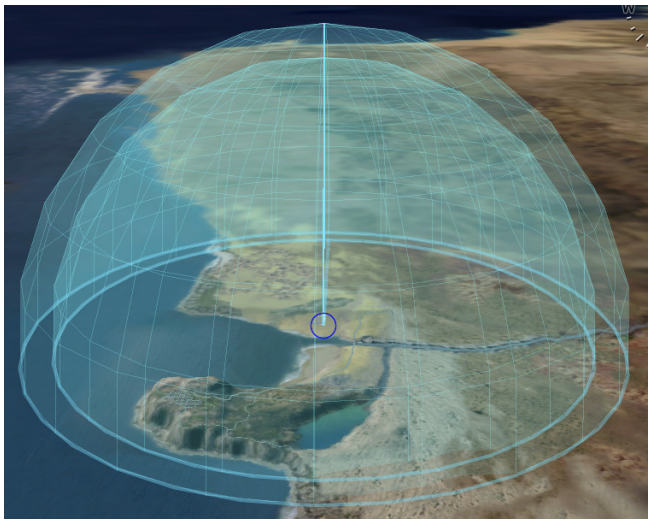


Figure 43-14. Range ring sphere

43.8.1. Enabling and Disabling the Display of Range Rings

To enable or disable the display of range rings:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-15).

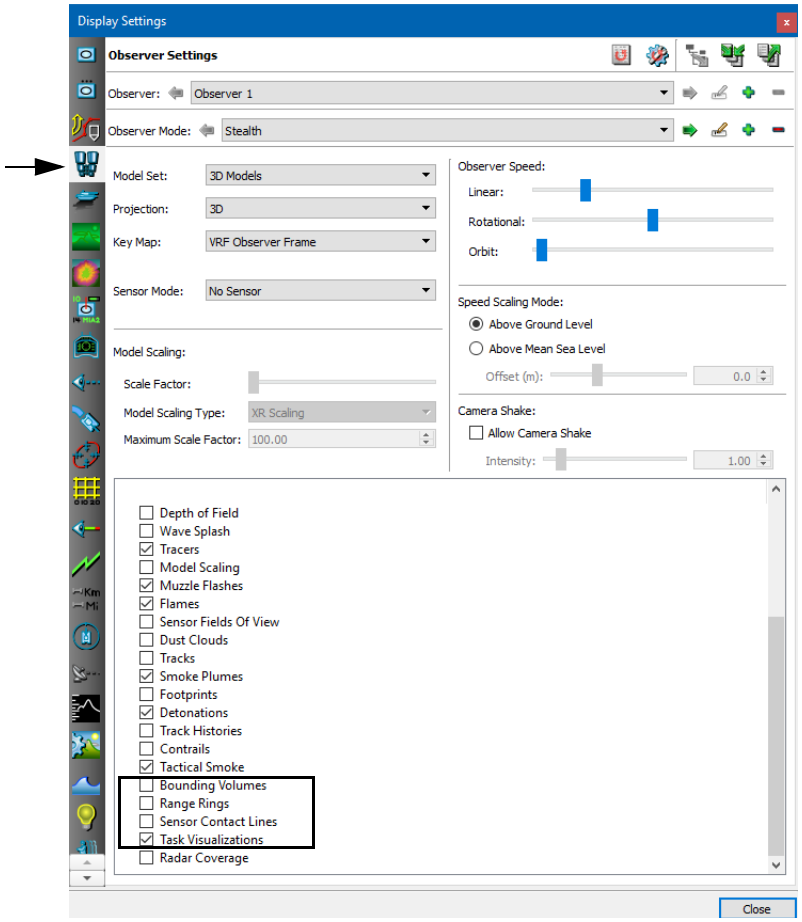


Figure 43-15. Observer Settings page

3. In the list of options, select or clear the Range Rings check box.

> You can also enable and disable range rings by choosing **Observer** → **Range Rings** or by clicking the Range Rings button ( ) on the Observer Settings Toolbar.

43.8.2. Pinning Range Rings for a Simulation Object

You display range rings on a per-entity basis. By default, if range rings are enabled, when you select an object, its range rings are displayed as long as the object is selected. You can pin range rings to an object so that they are displayed even if the object is not selected.

To pin range rings for a simulation object:

1. Enable range rings, as described in "Enabling and Disabling the Display of Range Rings," on page 43-18.
2. Select a simulation object.
3. Choose **Objects** → **Decorations** → **Pin Range Rings**.

43.8.3. Configuring the Display of Range Rings

You can configure the color of range rings and their transparency by weapon type and sensor type. You can also configure the display of range ring labels (Figure 43-16). A label can display the entity name, weapon type, and range of the weapon.

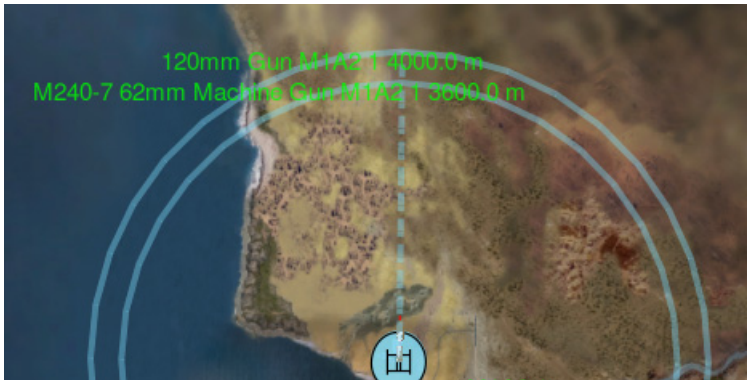


Figure 43-16. Range ring labels

To configure the display of range rings:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Range Rings Settings tab (Figure 43-17).

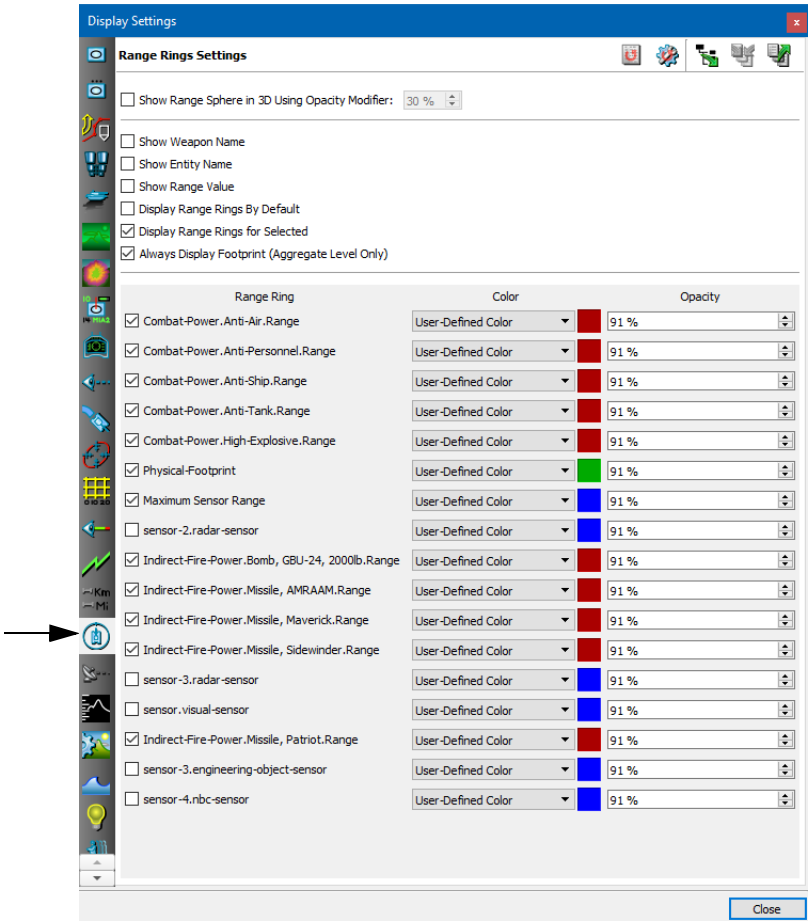


Figure 43-17. Range Rings Settings page (*AggregateLevel.sms*)

3. To display a sphere in 3D views, select the Show Range Sphere in 3D Using Opacity Modifier check box. Optionally, change the opacity modifier value.
4. To display the type of sensor or weapon system, select the Show Weapon Name check box.
5. To display the name of the simulation object, select the Show Entity Name check box.
6. To show the range of the sensor or weapon system, select the Show Range Value check box.
7. To automatically display range rings when a simulation object is created or discovered on the network, select Display Range Rings by Default.

If this setting is selected, when you load a scenario, range rings are displayed for all simulation objects.

---

8. To display range rings for a selected simulation object that has not enabled range rings, select Display Range Rings for Selected.
9. To always show the footprint of a simulation object, select Always Display Footprint.
10. To enable or disable display of range rings for a particular weapon system, select or clear the check box for that system.
11. To change the color used to display a range ring:
    a. For the type of weapon system or sensor whose color you want to change, select User-Defined Color in the Color column list. The color swatch next to the list changes.
    b. Click the color swatch. A color picker dialog box opens.
    c. Select the color you want to use.
    d. Click OK.
12. To change the opacity for a range ring, for the weapon system whose range ring opacity you want to change, change the value in the Opacity column.

43.9. Displaying Laser Designators

You can display laser designation information if it is available. VR-Forces displays a green circle and a range (on top of the entity icon or model) (Figure 43-18).

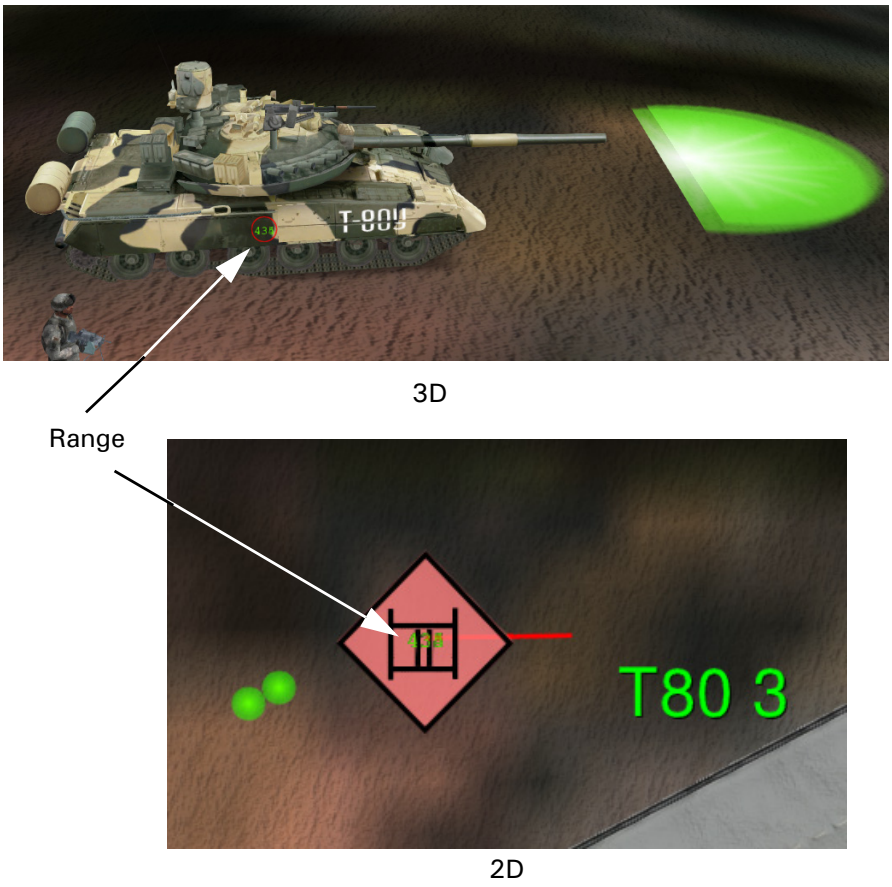


Figure 43-18. Laser designators

➤ To display laser designators, choose **Settings** → **Laser Designators**, or on the Display Settings Toolbar, click the Laser Designator button ( ).

43.10. Displaying Supply Lines

Simulation objects in aggregate-level scenarios can provide supplies to and receive supplies from other simulation objects. When a simulation object is supplying another entity you can display supply lines that show resupply is taking place.

To display supply lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 43-8).
3. Select or clear Show Supply Lines.

43.11. Displaying Tactical Smoke

VR-Forces can display tactical smoke munitions fired by VR-Forces entities. You may want to disable tactical smoke if it obscures the observer's view. Also, if many entities are generating smoke, it could affect performance.

- Do not confuse tactical smoke and smoke plumes. Smoke plumes are a visual effect displayed when an entity is on fire. Tactical smoke visualizes a smoke munition that affects line-of-sight.
- If you disable display of tactical smoke, VR-Forces continues to simulate its existence in the back-end and it continues to affect line-of-sight.

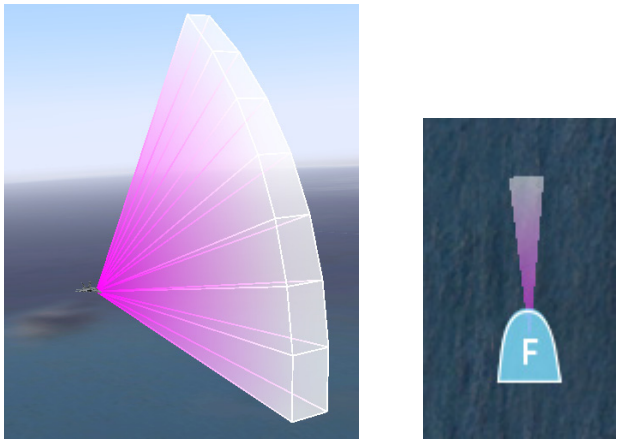
To enable tactical smoke:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
3. Select or clear the Tactical Smoke check box.

You can also enable or disable display of a fire or detonation effect by choosing **Observer** → **Tactical Smoke**.

43.12. Displaying Electromagnetic Emissions

VR-Forces can display electromagnetic emissions (emitter volumes) for entities that have emitter systems (Figure 43-19). You can configure the colors used to represent emitter frequencies and you can configure the size of the emitter volumes. (For details about configuring emitter volumes, please see "Configuring Emitter Volumes," on page 91-2.)



3D 2D

Figure 43-19. Emitter volumes

Display of emitter volumes only affects entities that have emitter systems.

To enable or disable display of electromagnetic emissions:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 43-20).

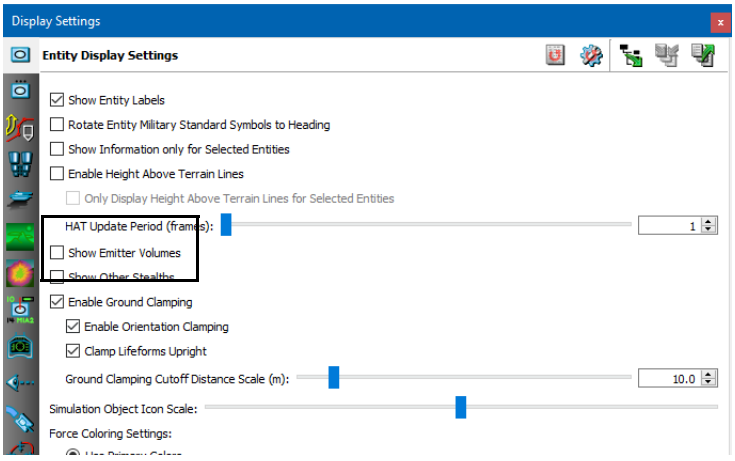


Figure 43-20. Entity Display Settings page

3. Select or clear the Show Emitter Volumes check box.

> You can also enable or disable emitter volumes by choosing **Settings** → **Emitter Volumes**, or clicking the Emitter Volumes button ( ) on the Display Settings Toolbar.

43.13. Displaying Radio Communication Lines

VR-Forces can display squawk indicators when a simulation object sends a signal interaction. It can also display communication lines between simulation objects when a VR-Forces simulation object (or a simulation object in an application built with the appropriate MAK libraries) sends a radio message to another simulation object. You can configure a communication line's color and lifetime. Figure 43-21 displays squawk indicators and a line-of-sight communication line.



Figure 43-21. Radio communication line and squawk indicators

For communications over large distances, a line-of-sight communication line would travel through the earth. Therefore, you can also display communication lines drawn with the curvature of the earth (Figure 43-22).

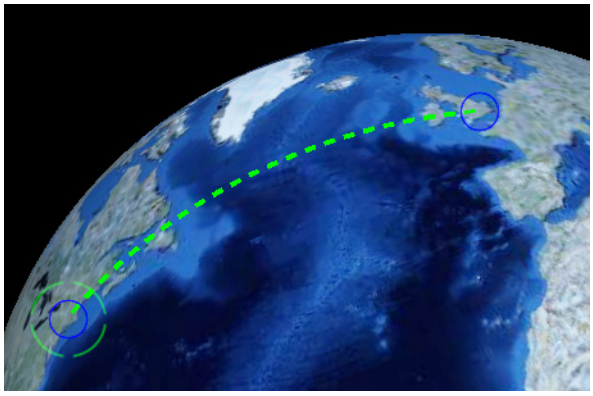


Figure 43-22. Curvature of the earth communication line

43.13.1. Enabling Radio Communications Graphics

You can globally enable or disable the ability to display radio communication lines. When they are enabled, you can enable and disable communication lines and squawk indicators separately.

To enable or disable radio communications graphics:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Radio Communications Settings page (Figure 43-23).

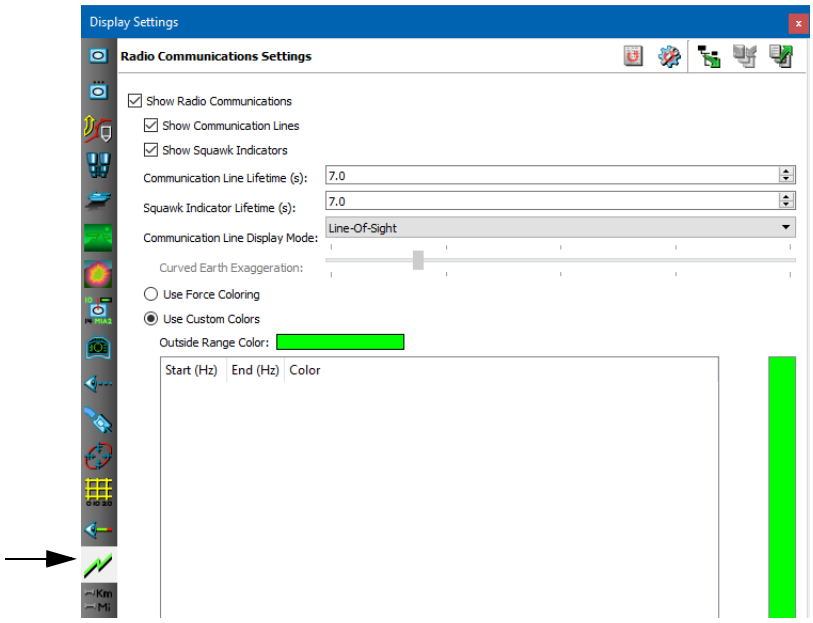


Figure 43-23. Radio Communications Settings page

3. Select or clear the Show Radio Communications check box.

> You can also enable and disable communication lines by choosing **Settings** → **Radio Communication Display** or by clicking the Radio Communications button (⁄⁄) on the Display Settings Toolbar.

To enable or disable communication lines:

1. Enable radio communication lines.
2. Select or clear the Show Communication Lines check box.

To enable or disable squawk indicators:

1. Enable radio communication lines.
2. Select or clear the Show Squawk Indicators check box.

43.13.2. Configuring the Lifetime and Display Mode

You can configure how long a radio communication line is displayed. You can also configure whether it is displayed using line-of-sight or curvature of the earth.

To configure the lifetime and display mode for radio communication lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Radio Communications Settings page (Figure 43-23).
3. To specify how long a communication line or squawk indicator should be displayed, enter a value, in seconds, in the Communication Line Lifetime and Squawk Indicator Lifetime fields.
4. To specify the display mode, select an option in the Communication Line Display Mode list.
5. If you select Curvature of the Earth, adjust the Curved Earth Exaggeration slider to change the degree to which the curvature is exaggerated.

43.13.3. Configuring the Color of Radio Communication Lines

By default, radio communication lines are green. You can configure communication lines to use force coloring or to be colored based on the frequency of the message. When you color by frequency, you specify colors for the frequency ranges you are interested in and a color to use for all messages outside the specified frequencies.

To configure the color of radio communication lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Radio Communications Settings page (Figure 43-23).
3. To color radio communication lines using the force color, select the Use Force Coloring option.
4. To color radio communication lines by frequency:
   a. Select the Use Custom Colors option.
   b. Click the Add button (). A range is added to the frequency list with a default color (Figure 43-24).

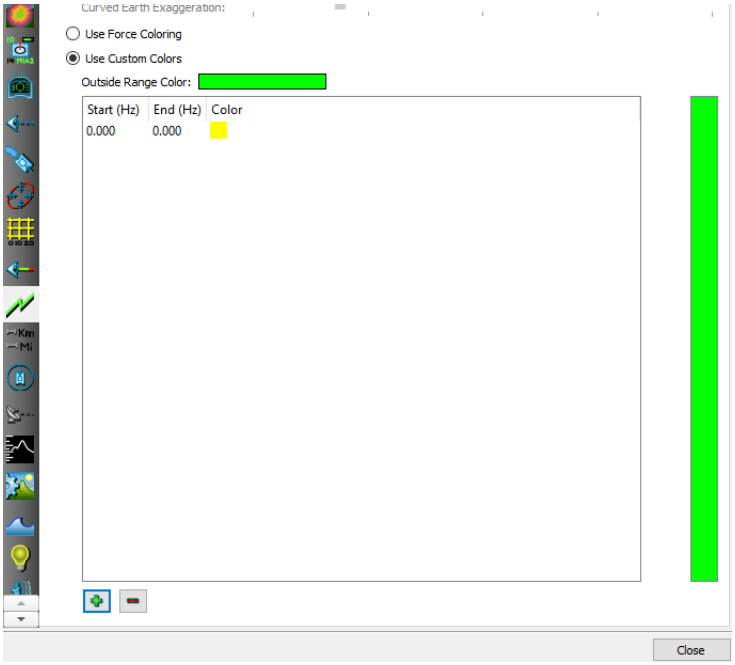


Figure 43-24. New frequency point

   c. Double-click the starting value of the range to make it editable (Figure 43-25).

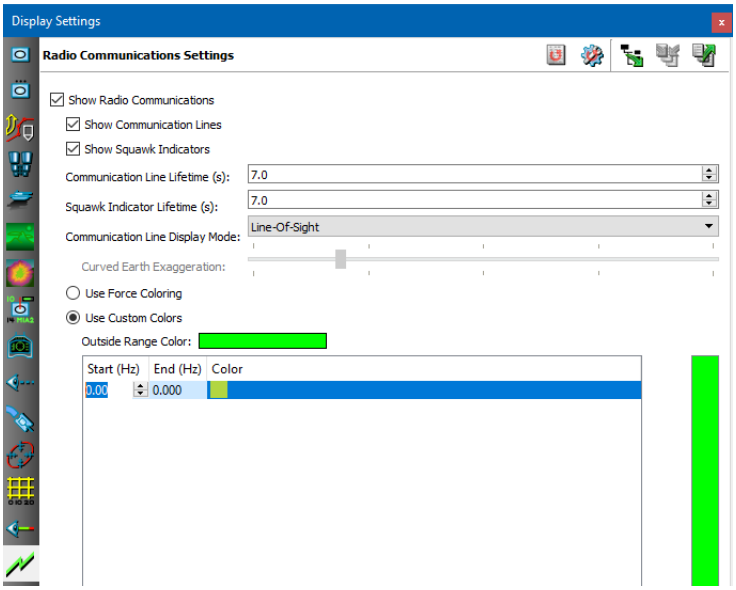


Figure 43-25. Editable frequency point

d. Type the starting frequency you want to use.
e. Edit the end frequency.
f. Double-click the color swatch and select a color for this range.
g. Optionally, add points for different frequency ranges.
h. Optionally, click the Outside Range Color color swatch and specify a color for all frequencies not specified in the frequency list.



If you configure multiple frequency ranges, you cannot add a range whose values are within those of an existing range.



44. Visualizing 2D Icons and 3D Models

This chapter describes how to change how 2D icons and 3D models look.

Simulation Object Icons 44-2
Changing the Size of 2D Icons 44-3
Rotating 2D Icons to a Simulation Object's Heading 44-4
Exaggerating the Scale of 3D Entity Models 44-5
Hiding 3D Models 44-8
Displaying DI-Guys in Vehicles 44-8

44.1. Simulation Object Icons

The 2D Icons model set uses MIL-STD 2525B icons. Figure 44-1 illustrates basic icon shapes for friendly forces.

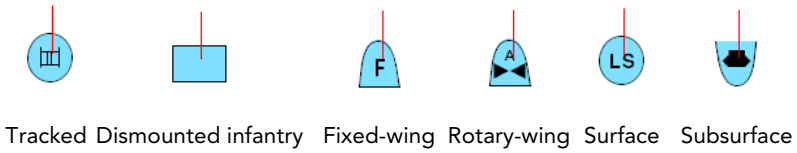


Figure 44-1. MILSTD-2525B icons

The color and shape code for 2D simulation object icons is:

- Light blue – friendly.
- Salmon – hostile.
- Light green – neutral.

Symbols have a black border except as follows:

- If there is no border on the top, the entity domain is subsurface, for example, submarines.
- Selected simulation objects have a white border.

Symbols that represent units have unit identifiers above the symbol, as illustrated in Figure 44-2.

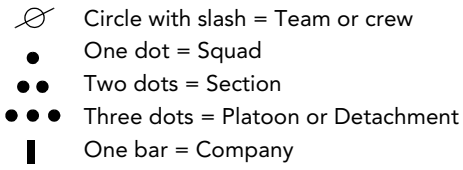


Figure 44-2. Unit echelon indicators

44.1.1. Changing the Size of 2D Icons

The Plan View observer mode uses the 2D Icons model set for simulation objects. You can change the size of 2D icons. When many simulation objects are close together, reducing the size of the icons can reduce overlap.

To change the size of simulation object icons:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 44-3).

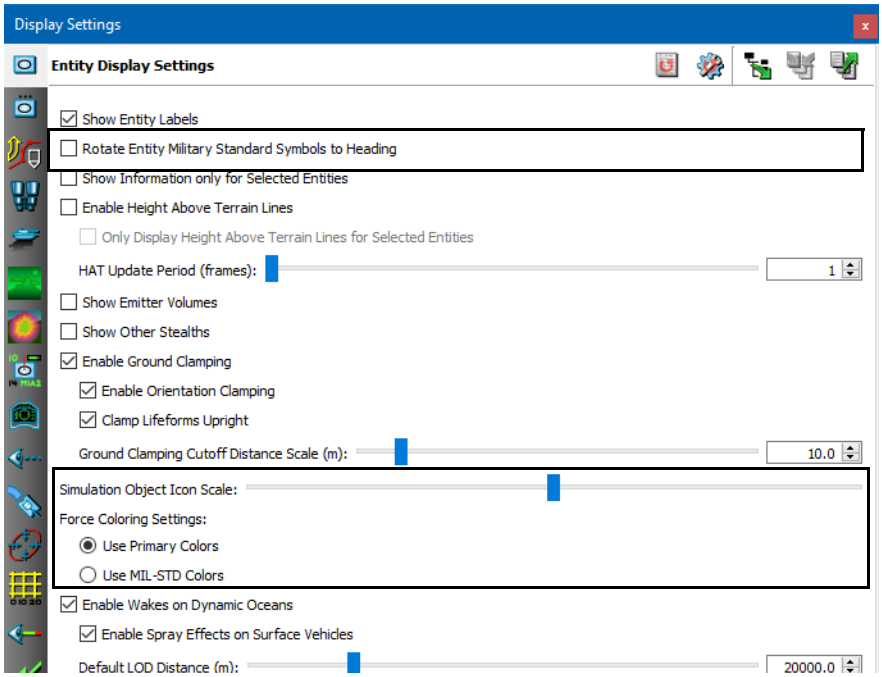


Figure 44-3. Entity Display Settings page

3. Adjust the Simulation Object Icon Scale slider.

You can also change the size of 2D icons from the Display Settings Toolbar.

To change the size of simulation object icons from the Display Settings toolbar:

1. On the Display Settings Toolbar, click the Icon Scaling button (). The Simulation Object Icon Scale Factor slider opens.
2. Adjust the slider to change the size of the icons.



The Simulation Object Icon Scale Factor slider is a dockable widget that you can undock and leave visible as long as icon scaling is enabled.

44.1.2. Rotating 2D Icons to a Simulation Object's Heading

By default, when a simulation object's heading changes, the heading indicator points to the new heading, but the icon remains oriented towards zero degrees (North). You can specify that icons rotate to the heading of the simulation object.

Rotating 2D icons to match an entity's heading works only when Orient North is enabled. If Orient North is disabled, the rotated 2D icons may not match the object's heading. Please see Section 51.9, "Forcing Orientation North" for details about controlling the observer's heading.



Figure 44-4. 2D icon rotated to heading

To rotate icons to a simulation object's heading:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 44-3).
3. Select the Rotate Entity Military Standard Symbols to Heading check box.

44.2. Exaggerating the Scale of 3D Entity Models

If you view the terrain in Stealth observer mode, beyond a certain distance most 3D entity models are too small to see. Model scaling lets you exaggerate the scale of entity models to see them at a great distance. Figure 44-5 shows the same scene with model scaling off and on. When model scaling is off, only the largest entities are visible. When models are scaled, they are all visible.

VR-Forces supports the following types of model scaling:

- XR scaling. XR scaling tries to give all entities approximately equal visibility. It does not preserve size relationships between entities. The visual effect of model scaling depends on an entity's size and the observer's distance from the entity. When the observer is close to an entity, scaling models may not make any noticeable difference in its size.
- Linear scaling. Linear scaling scales all entities by the current model scale factor. The distance from the observer is not taken into account. A model scale factor of 10 scales all entities to be 10 times normal size.

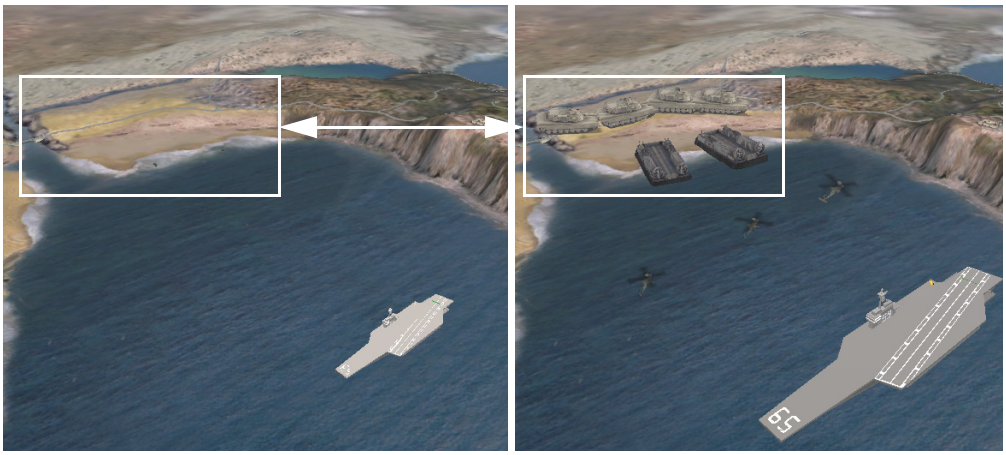


Figure 44-5. Model scaling off and on (Stealth observer mode)

Another way to quickly change the visibility of entities is to switch to XR observer mode. In XR mode, entities are immediately scaled and are represented by colorized 3D models to enhance their visibility.



Figure 44-6. XR mode

To scale entity models:

1. Choose **Observer** → **Advanced Settings**. The Display Settings dialog box opens to the Observer Settings page (Figure 44-7).

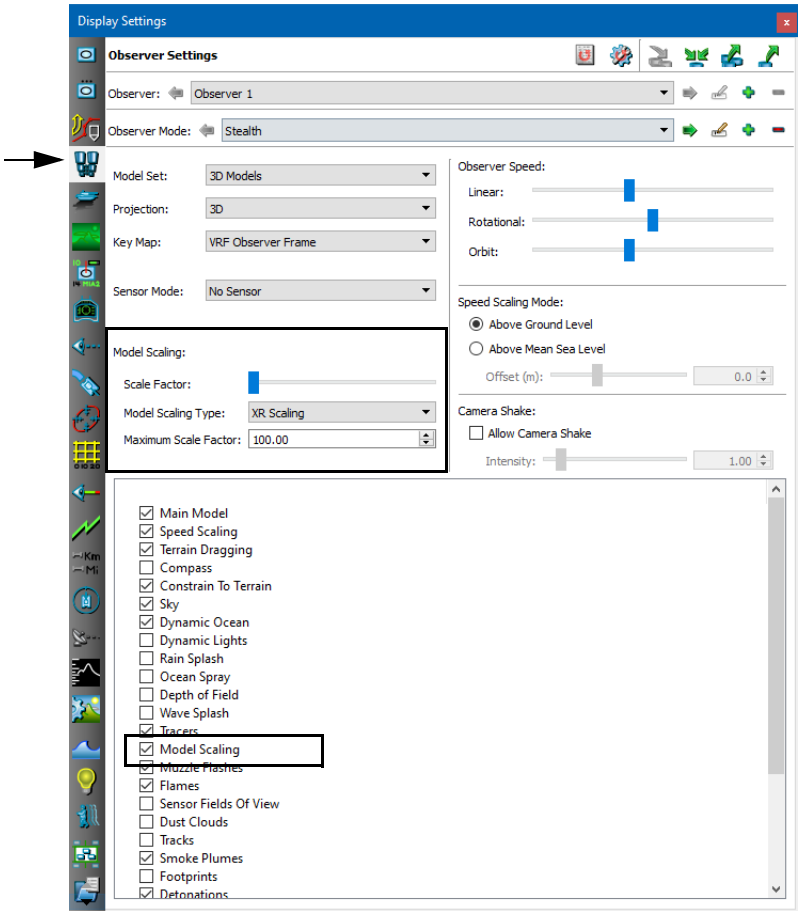


Figure 44-7. Observer Settings page

2. In the list of observer attributes, select Model Scaling.
3. In the Model Scaling Type list, select the type of scaling you want to use – XR or linear.
4. Optionally, change the Maximum Scale Factor.
5. Adjust the Scale Factor slider to the amount of scaling that you want.

44.3. Hiding 3D Models

You can hide 3D models and 3D colorized models.

To hide 3D models:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 44-7).
3. In the list of observer settings, clear the Main Model check box.

44.4. Displaying DI-Guys in Vehicles

If a vehicle has interior geometry, you can configure it to display DI-Guy characters inside the vehicle. If a vehicle is configured to display a DI-Guy, the model is added automatically. You can enable and disable display of DI-Guys. This is a global setting.

The characters displayed in vehicles are for display purposes only. They do not exist in the simulation.

You can show and hide occupants on a per-entity basis with the Hide Occupants set data request. For details, please see "Hide Occupants," on page 34-20.

To enable or disable display of DI-Guys in vehicles:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 44-3).
3. Select or clear the Show Occupants check box.



45. Lighting Effects

This chapter describes the lighting effects supported by VR-Forces.

Displaying Lighting Effects 45-2
Enabling and Disabling Lighting 45-3
Configuring Ambient and Diffuse Lighting 45-4
Specifying the Emissive Lighting Scale 45-5
Displaying Dynamic Lighting 45-6
Daylight Control of Lights 45-7
Displaying Ocean Planar Reflections 45-8
Configuring Ocean Planar Reflections 45-10
Lens Flare 45-11
Displaying Crepuscular (Sun) Rays 45-13
Configuring the Skybox Cube Map 45-15
Flashlight Mode 45-16
Using High Dynamic Range Lighting 45-17
Disabling Lighting and Clouds 45-19
Displaying Shadows (3D Only) 45-20
Enabling Shadows on Trees 45-22
Configuring Shadows 45-23

45.1. Displaying Lighting Effects

VR-Forces can simulate a variety of lighting effects, including:

- Dynamic lighting
- Ocean planar reflection
- Lens flare
- Crepuscular (sun) rays
- Shadows

Use of lighting effects affects performance. VR-Forces lets you enable and disable some effects so you can adjust the mixture of performance and effects that best meets your needs.

45.2. Enabling and Disabling Lighting

When lighting is enabled, VR-Forces takes into account direct and ambient lighting when it renders a scene. During the daytime, objects are brightly lit. At night, you cannot see them unless an object is casting light on them or you use a sensor mode that provides night vision.

If lighting is disabled, you can see objects regardless of the time of day. Lighting may not be realistic.

To enable or disable lighting:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 45-1).

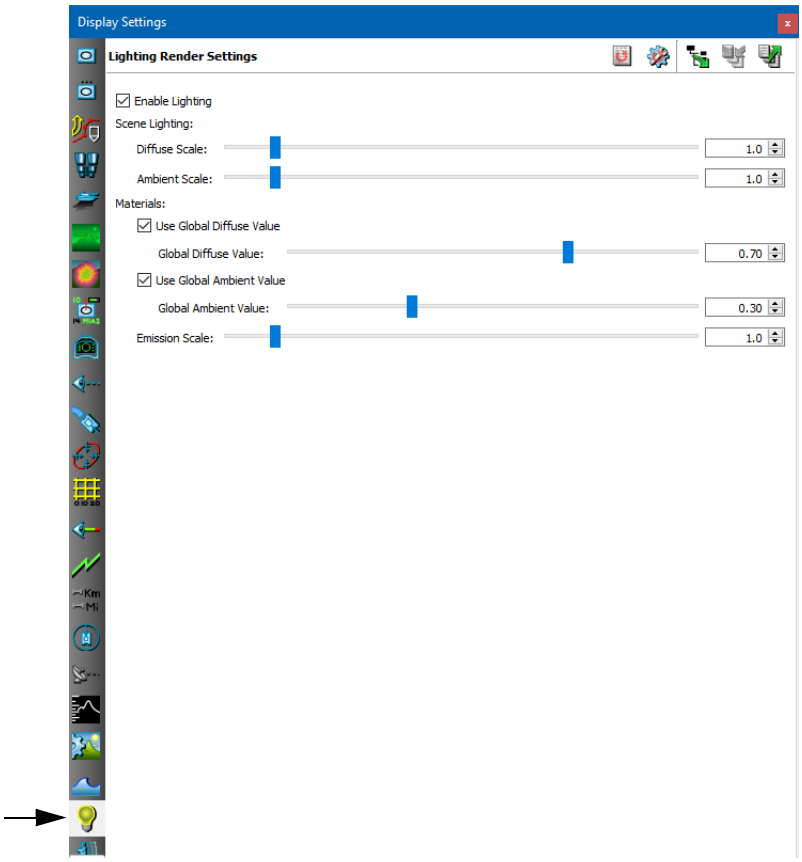


Figure 45-1. Lighting Render Settings

3. Select or clear the Enable Lighting check box.

45.2.1. Configuring Ambient and Diffuse Lighting

Ambient lighting is the lighting available to objects that are not in a direct form of light, such as sunlight. Diffuse lighting is light that is reflected off an object. You can adjust the overall amount of ambient and diffuse lighting in the scene and you can configure whether objects use the material ambient and diffuse values specified in their models or use values provided by VR-Forces.

Configuring the Ambient and Diffuse Scale Values for Scene Lighting

You can set a scale factor for ambient and diffuse light that is present in the scene. Lower values make objects darker. Higher values make them lighter. The ambient or diffuse scale factor is independent of how material ambient and diffuse values are set.

To specify the ambient and diffuse lighting scale values:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 45-1).
3. Adjust the Diffuse Scale slider or change the value in the box.
4. Adjust the Ambient Scale slider or change the value in the box.

Configuring the Material Ambient and Diffuse Values for Materials

Polygons in a 3D scene have a material and a texture. Polygon materials may specify specular, diffuse, and ambient colors that define the way they respond to incoming light.

The ambient color specifies the color used when it is in indirect light. Sometimes the ambient values for different objects do not work well together. You can specify that objects use the ambient material value, or let VR-Forces globally specify ambient material values.

In addition to specifying which material ambient and diffuse values to use, you can set a scale factor. Lower values make objects darker. Higher values make them lighter. The material ambient or diffuse scale factor is independent of how ambient and diffuse values are set for scene lighting.

To configure the material ambient value:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 45-1).
3. To use the ambient values specified by objects, clear the Use Global Ambient Value check box. To have VR-Forces use a global value, select the Use Global Ambient Value check box.
4. If you selected the Use Global Ambient Value check box, optionally, adjust the Global Ambient Value slider or change the value in the box.

To configure the material diffuse value:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 45-1).
3. To use the diffuse values specified by objects, clear the Use Global Diffuse Value check box. To have VR-Forces use a global value, select the Use Global Diffuse Value check box.
4. If you selected the Use Global Diffuse Value check box, optionally, adjust the Global Diffuse Value slider or change the value in the box.

45.2.2. Specifying the Emissive Lighting Scale

You can change a scale value that controls the brightness of emissive textures. This allows you to make windows, street lights, and other emissive light sources darker or brighter. Increasing the scale value increases the brightness.

To specify the emissive lighting scale value:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 45-1).
3. Adjust the Emission Scale slider or change the value in the box.

45.3. Displaying Dynamic Lighting

VR-Forces can display light cast by light sources such as headlights, street lamps, and spotlights. Display of dynamic lighting is an observer-specific feature. It is disabled by default.

Dynamic lights are either part of the terrain or part of an entity model. If they are part of an entity model, they can be set permanently on or they can be turned on and off by the appearance bit. VR-Forces entities that have lights can enable them using the Set Appearance set data request.

Figure 45-2 shows a night-time scene with dynamic lighting enabled. The HMMWV's taillights cast light to the rear. The headlights cast light forward. Street lamps and windows are also casting light.



Figure 45-2. Dynamic Lighting enabled

- Light points can be configured to automatically turn on at dusk and turn off at dawn. For details, please see "Daylight Control of Lights," on page 45-7.
- Emissive lights, such as automobile headlights, are always on.

You cannot change these lighting settings.

To enable dynamic lighting,

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page.
3. In the list of settings, select Dynamic Lights.



You can also enable or disable dynamic lighting by choosing **Observer** → **Dynamic Lights**.

45.3.1. Daylight Control of Lights

Lights that have a light point type specified can be placed under daylight control.

- Light points that do not have a light point type defined are not automatically placed under daylight control.
- Light points in an OpenFlight file that are loaded as an articulated model use the daylightControlledLights model definition parameter.
- Light points with a type defined use the Daylight Controlled parameter for their type in *./appData/importConfig/Lights_DefaultValues_3_2.csv*.

For more information about configuring light points, please see "Configuring Light Points," on page 90-14.

Light sources (lights that cast light onto other objects) are discovered on the cull pass of the scene graph. Sources found under the terrain root or props root of the scene graph are automatically put under daylight control.

45.4. Displaying Ocean Planar Reflections

VR-Forces can show the reflection of terrain and objects on water (Figure 45-3). To see planar reflections, dynamic ocean must be enabled. (For details about enabling dynamic ocean, please see "Enabling Marine Effects," on page 11-23.)

Displaying ocean planar reflections degrades performance.



Figure 45-3. Ocean planar reflection

To enable or disable planar reflections:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 45-4).

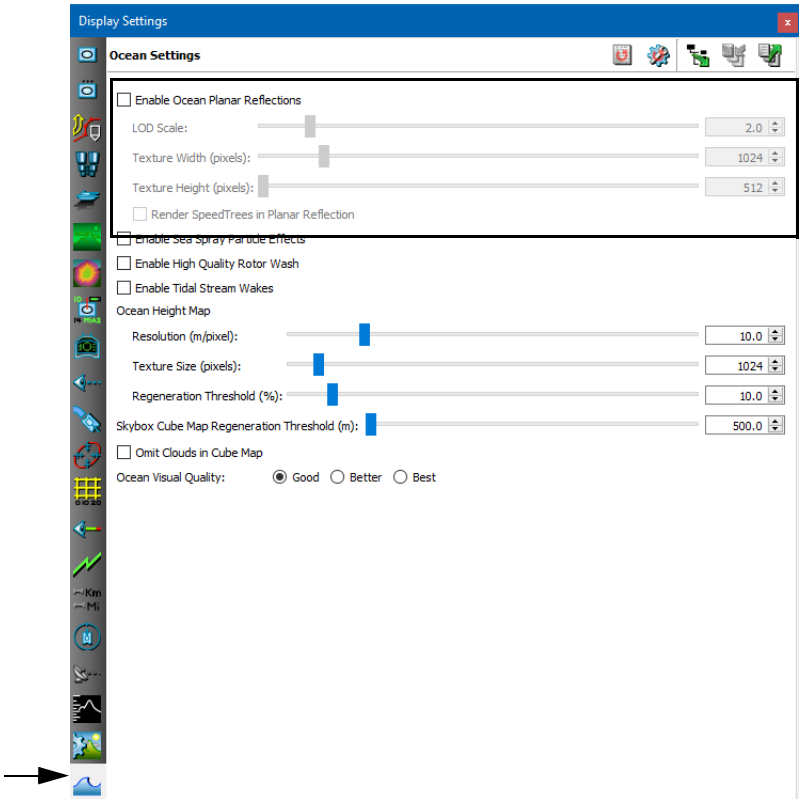


Figure 45-4. Ocean Settings

3. Select or clear the Enable Ocean Planar Reflections check box.

45.4.1. Configuring Ocean Planar Reflections

To reduce the performance impact of ocean planar reflections, you can configure the LOD scale. The LOD scale affects the distance from the observer at which VR-Forces switches LODs. The higher the value, the more quickly LODs are switched.

The Texture Height and Texture Width parameters affect the resolution of the ocean planar reflection texture, in pixels. Higher resolution decreases performance.

You can also enable and disable display of SpeedTrees as planar reflections.

To configure ocean planar reflections:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 45-4).
3. To change the ocean planar reflection LOD scale, adjust the LOD Scale slider under Enable Ocean Planar Reflections, or change the value in the box.
4. To change the width or height of the ocean planar reflection texture, adjust the Texture Width or Texture Height slider under Enable Ocean Planar Reflections, or change the values in the boxes.
5. To include SpeedTrees in ocean planar reflections, select the Render SpeedTrees in Planar Reflections check box.

45.5. Lens Flare

Lens flare is a visual artifact often seen when looking at light sources such as the sun through a camera lens (Figure 45-5). VR-Forces can simulate lens flare when the observer looks towards the sun.



Figure 45-5. Lens flare



Lens flare is supported only when high dynamic range lighting is enabled.

To enable or disable lens flare:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the High Dynamic Range Settings page (Figure 45-6).

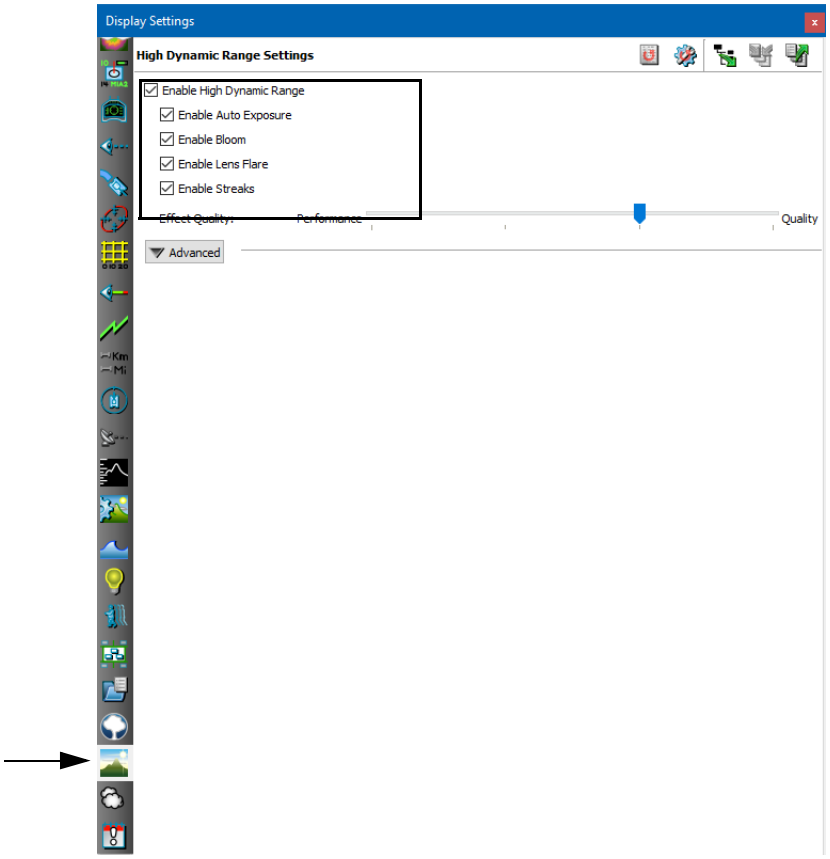


Figure 45-6. High Dynamic Range Settings page

3. Select or clear the Enable Lens Flare check box.

45.6. Displaying Crepuscular (Sun) Rays

VR-Forces can simulate the effect of crepuscular rays (sunbeams) beaming through clouds (Figure 45-7).

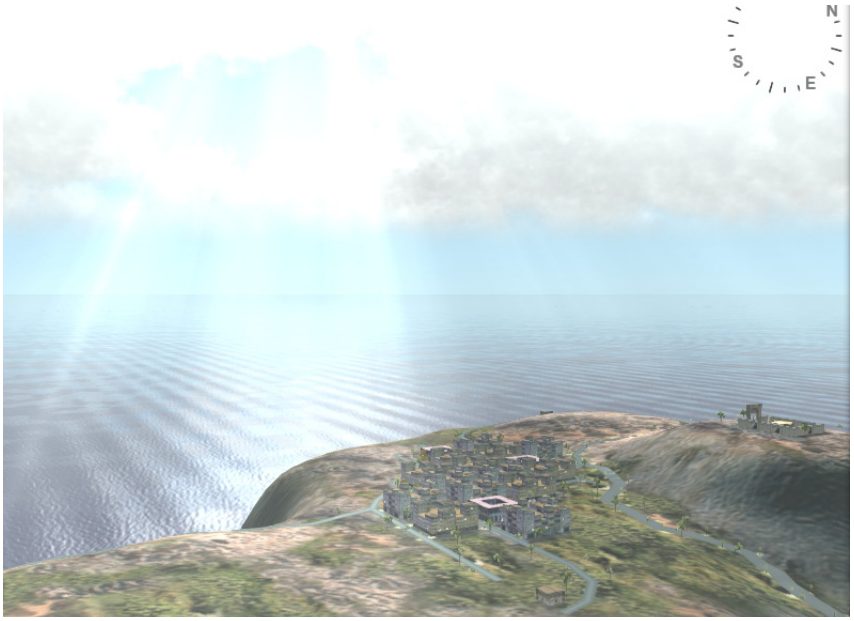


Figure 45-7. Crepuscular rays

To enable or disable crepuscular (sun) rays:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 45-8).

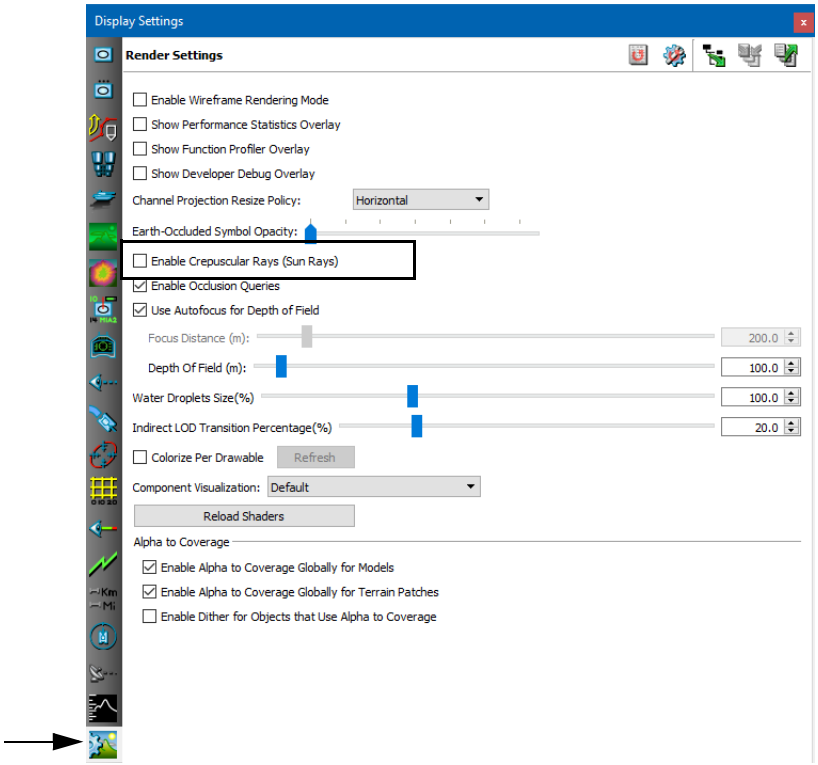


Figure 45-8. Render Settings page

3. Select or clear the Enable Crepuscular Rays (Sun Rays) check box.

45.7. Configuring the Skybox Cube Map

A skybox cube map is a graphics technique used to render reflections and sky effects. The skybox cube map is rendered from the location of the observer. As the observer moves, the cube map needs to be recalculated, which is computationally expensive.

VR-Forces lets you configure the distance that the observer has to move (the skybox cube map regeneration threshold) before it recalculates the cube map. Increasing the threshold can improve performance, at the cost of visual quality.

You can also choose to omit clouds in the skybox cube map. Omitting clouds can increase performance.

To configure the skybox cube map:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 45-4).
3. Adjust the Skybox Cube Map Regeneration Threshold slider or enter a value in the box.
4. Optionally, select the Omit Clouds in Cube Map check box.

45.8. Flashlight Mode

VR-Forces can light a circular area as if the observer is using a flashlight. You can move the flashlight beam and increase or decrease the diameter of the flashlight beam.



Figure 45-9. Flashlight mode

- To turn flashlight mode on or off, press **f**.
- To move the flashlight beam, hold down **ALT** and the right mouse button and move the mouse.
- To change the size of the flashlight beam, hold down **ALT** and move the mouse scroll wheel forward or back.

45.9. Using High Dynamic Range Lighting

High dynamic range (HDR) lighting makes scenes look more realistic. They look particularly good on HDR monitors. HDR is enabled by default.

To configure HDR lighting:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the High Dynamic Range Settings page (Figure 45-6).
3. To enable or disable HDR lighting, select or clear the Enable High Dynamic Range check box.
4. Enable other settings as desired.
5. For those settings that are enabled, adjust their parameters. Table 45-1 describes the HDR settings.

Table 45-1: HDR settings (Sheet 1 of 2)

Setting	Description
Enable High Dynamic Range	Turns High Dynamic Range (HDR) on and off. When HDR is turned on, the scene is rendered in 16-bits per color floating point and then tone mapped to 8-bits per color before being sent to the display. Additionally, there are post process effects applied to objects that are greater than the display range.
Enable Auto Exposure	Turns auto exposure on and off, which is a post process that scales the intensity of the rendered image based on the intensity of the screen. When auto exposure is turned off, the global lighting values are used to scale the intensity to a reasonable value.
Enable Bloom	Turns the bloom post process effect on and off. This effect is a blurring of high intensity objects in the scene which gives a glow around bright objects like light points.
Enable Lens Flare	Turns the lens flare post process effect on and off. This effect creates a lens flare effect for high intensity objects.
Enable Streaks	Turns the streak post process effect on and off. This effect creates a star effect by streaking the high intensity objects in all directions.
Dynamic Light Scale	When HDR is turned on, the intensity of dynamic lights are scaled by this setting.
Emissive Texture Scale	When HDR is turned on, the intensity of the emissive textures is scaled by this setting.
Light Point Scale	When HDR is turned on, light point intensity is scaled by this setting.
Shininess Scale	When HDR is turned on the lighting shininess component is scaled by this setting. Exposure adjustments are all related to how the intensity of the scene is scaled before being tone mapped. These values are in order based on when they are applied to the scalar.

Table 45-1: HDR settings (Sheet 2 of 2)

Setting	Description
Specular Scale	When HDR is turned on the lighting specular component is scaled by this setting.
Rate Scale	When Auto Exposure is enabled, this scales the time it takes for the auto exposure to adjust to changes in the scene. Increase this value to reduce the time it takes for the auto exposure to adjust.
Pre Scale	When Auto Exposure is enabled, this scales the calculated intensity for the scene. Increase this value to reduce the overall intensity of the final image.
Min Difference	When Auto Exposure is enabled, this clamps the minimum value of the calculated intensity to a portion of the global lighting value. This is used to prevent the auto exposure from going too low compared to the global lighting value.
Max Difference	When Auto Exposure is enabled, this clamps the maximum value of the calculated intensity to a portion of the global lighting value. This is used to prevent the auto exposure from going too high compared to the global lighting value.
Post Scale	Scales the scalar after all Auto Exposure calculations have been done. This also affects the global lighting value when Auto Exposure is turned off. Increase this value to reduce the overall intensity of the final image.
Final Clamp Min	The minimum value for the final value that will be used to scale the intensity prior to tone mapping.
Final Clamp Max	The maximum value for the final value that will be used to scale the intensity prior to tone mapping.
Bloom Scale	Scales the bloom before it is applied to the scene. Increase this value to increase the intensity of the bloom effect on high intensity objects.
Lens Flare Scale	Scales the lens flare effect before it is applied to the scene. Increase this value to increase the intensity of the lens flare effect.
Streak Intensity Scale	Scales the streaks before they are applied to the scene. Increase this value to increase the intensity of the streak effect on high intensity objects.
Number of Streaks	Number of streaks that the streak effect will create for each high intensity object. Maximum of 8 streaks.
Streak Length Scale	Scales the length of the streak effect. Increase this value to increase the length of the streak effect.
Streak Rotation Offset	Rotates the streak effect so that it changes the direction that the streaks point.
Gamma	Final color correction used to adjust the image based on the display. Change this value based on the characteristics of the display being used.

45.10. Disabling Lighting and Clouds

When you are using XR observer mode and you move the observer out from the terrain, the lighting and cloud effects can interfere with your ability to view simulation objects. Therefore, VR-Forces lets you quickly disable lighting effects and clouds. By default, they are enabled in Stealth observer mode and disabled in XR observer mode.

➤ To enable or disable lighting and clouds, choose **Observer** → **Sky**.

45.11. Displaying Shadows (3D Only)

VR-Forces can display shadows for entities, props, clouds, and terrain features (Figure 45-10).



Figure 45-10. Shadows

You can enable and disable shadows and you can configure shadow quality.

Enabling shadows can affect performance.

To enable and disable shadows globally:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Shadow Settings page (Figure 45-11).

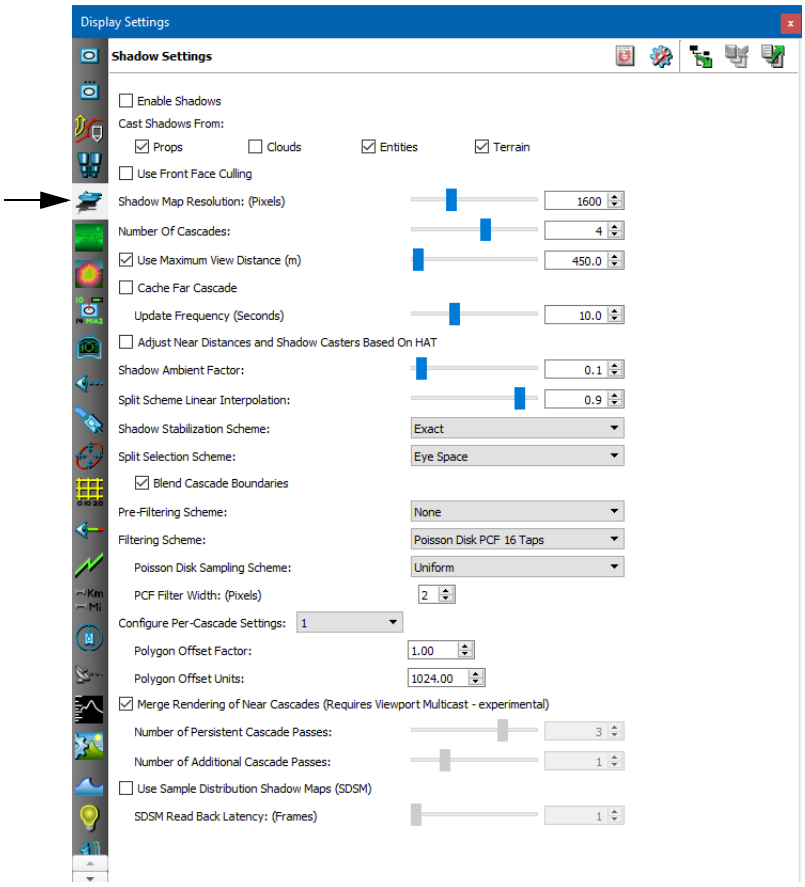


Figure 45-11. Display Settings, Shadow Settings page

3. Select or clear the Enable Shadows check box.
4. To enable or disable shadows for specific object types, select or clear the check boxes under Cast Shadows From.



You can also enable and disable shadows by choosing **Settings** → **Shadows**.

45.11.1. Enabling Shadows on Trees

You can enable and disable casting of shadows on SpeedTree trees.

To enable or disable casting shadows on SpeedTree trees:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the SpeedTree Settings page (Figure 45-12).

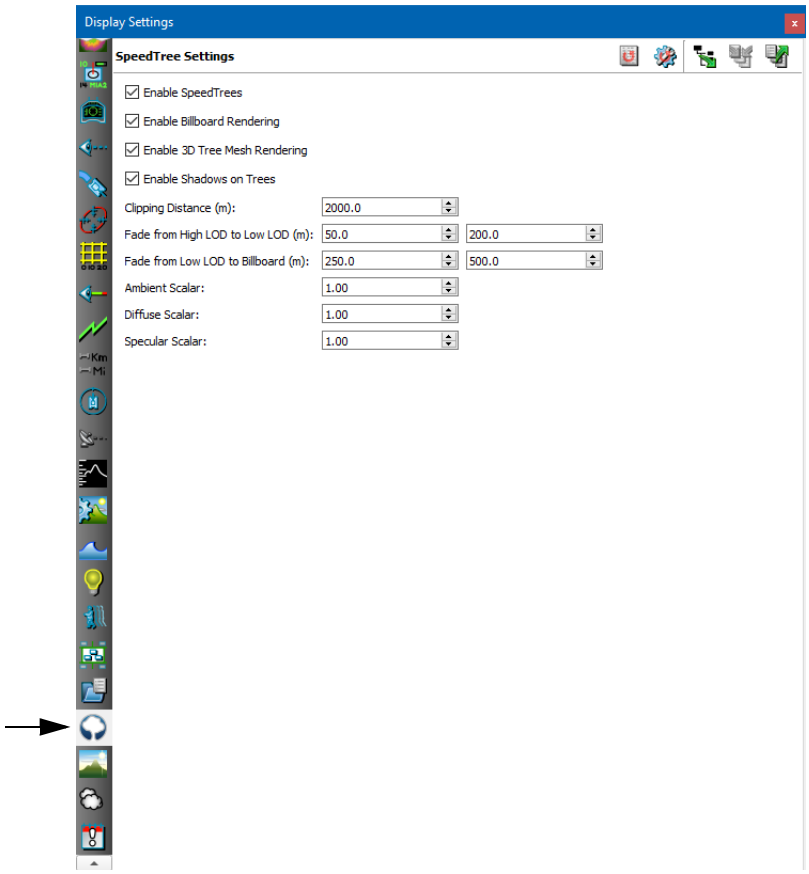


Figure 45-12. SpeedTree Settings page

3. Select or clear the Enable Shadows on Trees check box.

45.11.2. Configuring Shadows

You can configure which objects cast shadows and parameters that affect performance and the quality of shadows.

Shadows are drawn based on a shadow map. A shadow map determines what parts of a scene are in shadow relative to a light source. Then the scene is drawn showing the shadows that would be visible from the viewpoint of the observer.

Scenes are split into sections, called cascades. The cascades that are closest to the camera show the highest resolution shadows. Many of the settings control the size of cascades and how they merge together.

For an in-depth understanding of how shadows are rendered in 3D scenes, please see the many articles available on the world wide web.

To configure how shadows are rendered:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Shadow Settings page (Figure 45-11).
3. Change the parameters as desired. Table 45-2 briefly describes them.

Table 45-2: Shadow settings (Sheet 1 of 2)

Setting	Description
Use Front Face Culling	Use of face front culling is a choice between more shadow acne (an effect similar to z-fighting) and more light bleeding artifacts.
Shadow Map Resolution	Increasing shadow map resolution provides more detail at the cost of performance and increased memory use.
Number of Cascades	The more cascades there are, the more detail is rendered, at the expense of memory usage and performance.
Use Maximum View Distance	If enabled, specifies how far away shadows are displayed. The smaller the area in which shadows are displayed, the more quickly they get displayed and the higher the quality. If disabled, uses the scene values.
Cache Far Cascade	Caches the farthest cascade and updates it less frequently than the near cascade, which increases performance.
Adjust Near Distances and Shadow Casters Based On HAT	As the observer moves higher above the terrain, the near cascade shrinks and only entities are rendered to them. The far cascade becomes responsible for the terrain shadows.
Shadow Ambient Factor	Fades out shadows. It determines how much light gets through.
Split Scheme Linear Interpolation	Adjusts how cascades are distributed between a logarithmic approach and a linear approach.

Table 45-2: Shadow settings (Sheet 2 of 2)

Setting	Description
Shadow Stabilization Scheme	A list of different ways to reduce the shimmer that is common in shadow maps.
Split Selection Scheme	A list of different ways to choose which cascade a piece of geometry is in.
Pre-Filtering Scheme	A list of different shadow casing techniques to try to smooth shadows.
Filtering Scheme	Affects how blurred the shadow map is.
Configure Per-Cascade Settings	Lets you configure the polygon offset for cascades.
Merge Rendering of Near Cascades	Lets you draw cascades all at once instead of individually. This improves performance. However, it is only available on newer video cards (NVidia 980 or newer). If disabled, you can configure the number of cascades passes to draw shadows.
Use Sample Distribution Shadow Map (SDSM)	Adjusts quality based on the depth complexity of an area of the scene.



46. Marine Effects

This chapter describes different marine effects and water droplets.

Displaying Water Droplets and Splash Effects 46-2
- Enabling Sea Spray 46-4
- Setting the Size of Water Droplets 46-5
- Enabling and Disabling Rotor Wash 46-6
- Enabling High Quality Rotor Wash 46-6

Displaying Wakes and Spray Effects 46-7

Enabling Buoyancy for Surface Entities 46-8

46.1. Displaying Water Droplets and Splash Effects

When it is raining, VR-Forces can simulate raindrops hitting the observer's camera lens (Figure 46-1). If the observer is close to the surface of the ocean (as if a periscope were being raised or lowered), VR-Forces can simulate the effect of waves and water droplets splashing on the observer's camera.



Figure 46-1. Rain effect

When dynamic ocean is enabled, VR-Forces can simulate sea spray (most visible in high seas) (Figure 46-2). It can also display rotor wash for hovering helicopters. You can display rotor wash at two levels of quality.

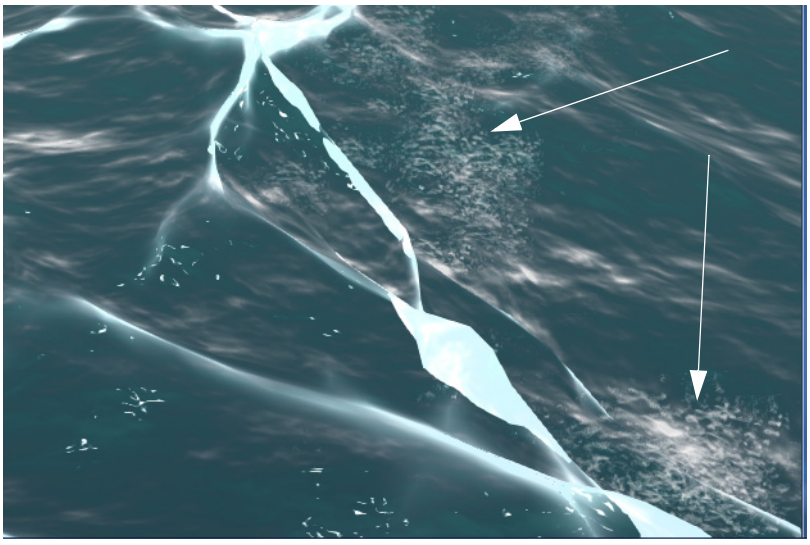


Figure 46-2. Sea spray effects

You can change the size of the water droplets for splash effects and sea spray. For details, please see "Setting the Size of Water Droplets," on page 46-5.

To enable or disable splash effects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 46-3).

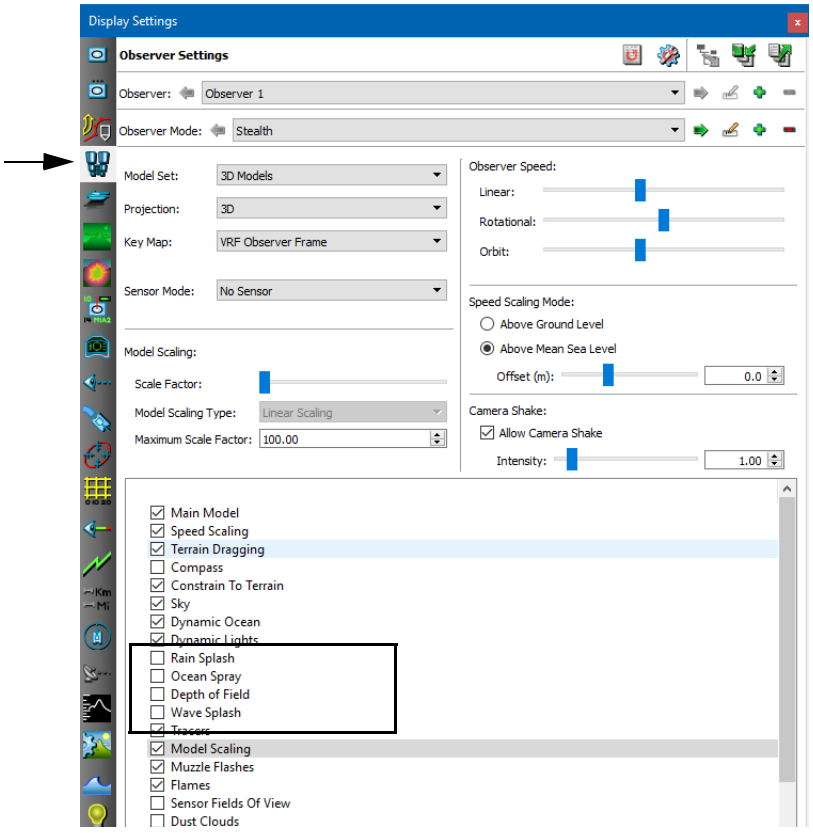


Figure 46-3. Observer Settings page

3. Select or clear the Rain Splash, Wave Splash, or Ocean Spray effect check boxes.

46.1.1. Enabling Sea Spray

To enable sea spray effects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 46-4).

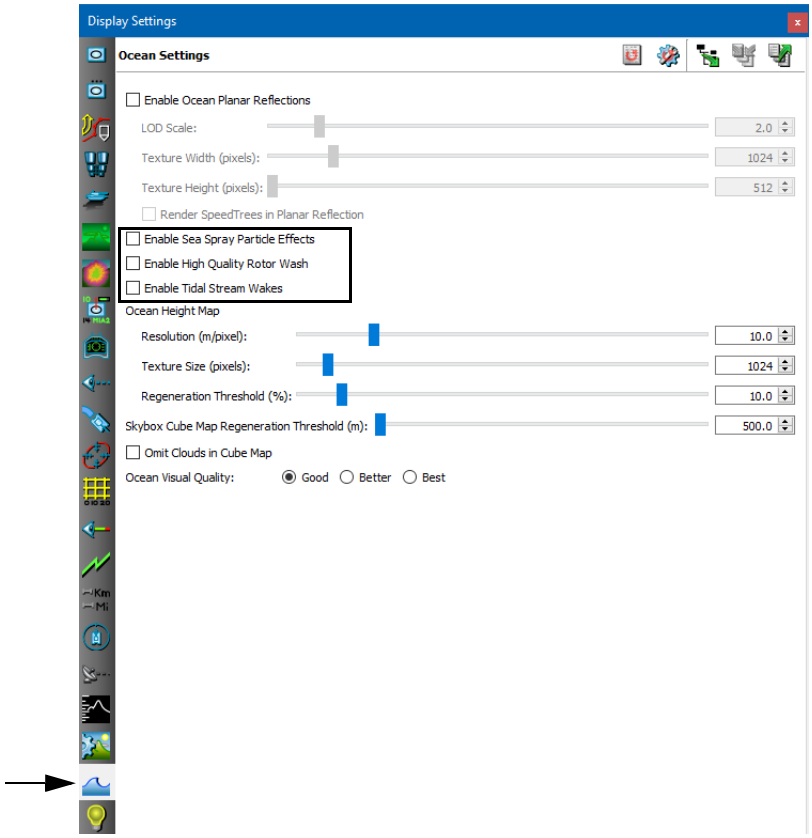


Figure 46-4. Ocean Settings page

3. Select Enable Sea Spray Particle Effects.

46.1.2. Setting the Size of Water Droplets

You can change the size of water droplets for splash effects and sea spray.

To change the size of water droplets:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 46-5).

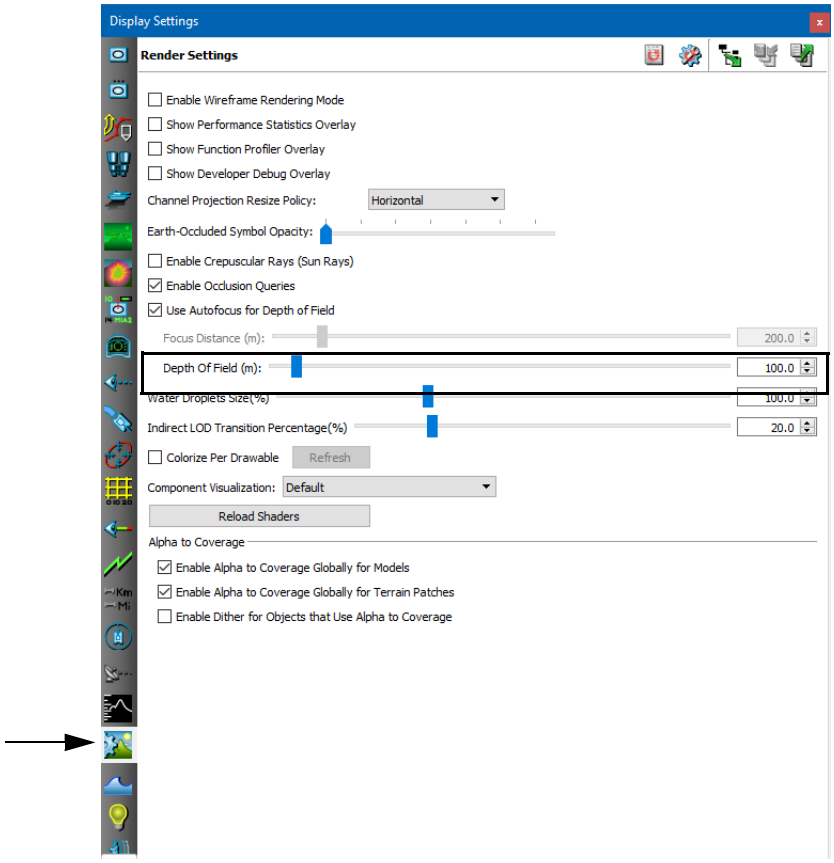


Figure 46-5. Render Settings page

3. Adjust the Water Droplets Size slider or enter a value in the box.

46.1.3. Enabling and Disabling Rotor Wash

If dynamic ocean is enabled, VR-Forces can display rotor wash for rotary-wing entities. You can enable or disable rotor wash globally.

To enable or disable rotor wash on dynamic ocean:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 46-6).

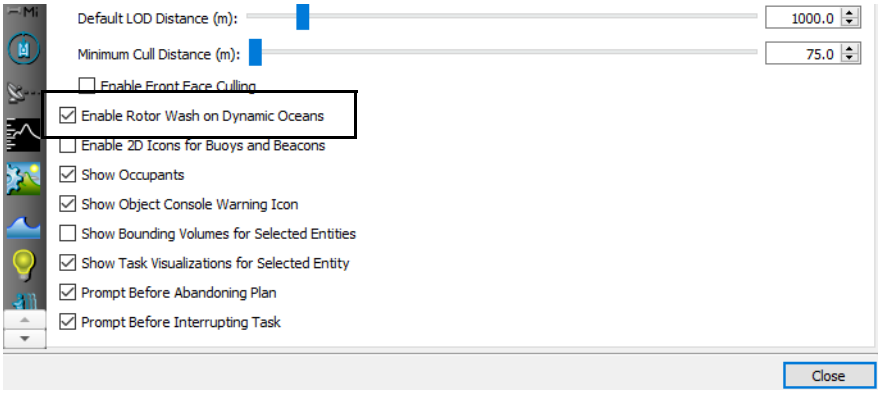


Figure 46-6. Rotor wash settings

3. Select or clear the Enable Rotor Wash on Dynamic Oceans check box.

46.1.4. Enabling High Quality Rotor Wash

To enable high quality rotor wash:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 46-4).
3. Select Enable High Quality Rotor Wash.

46.2. Displaying Wakes and Spray Effects

When dynamic ocean is enabled, surface entities display wakes and spray effects. However, these effects affect performance. You can disable them if you need to emphasize performance over visual effects.

Wakes have their own schema, with many attributes, including length, offset, the amount of time over which they fade out, and LOD distance. You can edit these attributes for individual entities in the Visual Model Editors dialog box.

To enable or disable wakes and spray effects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 46-7).

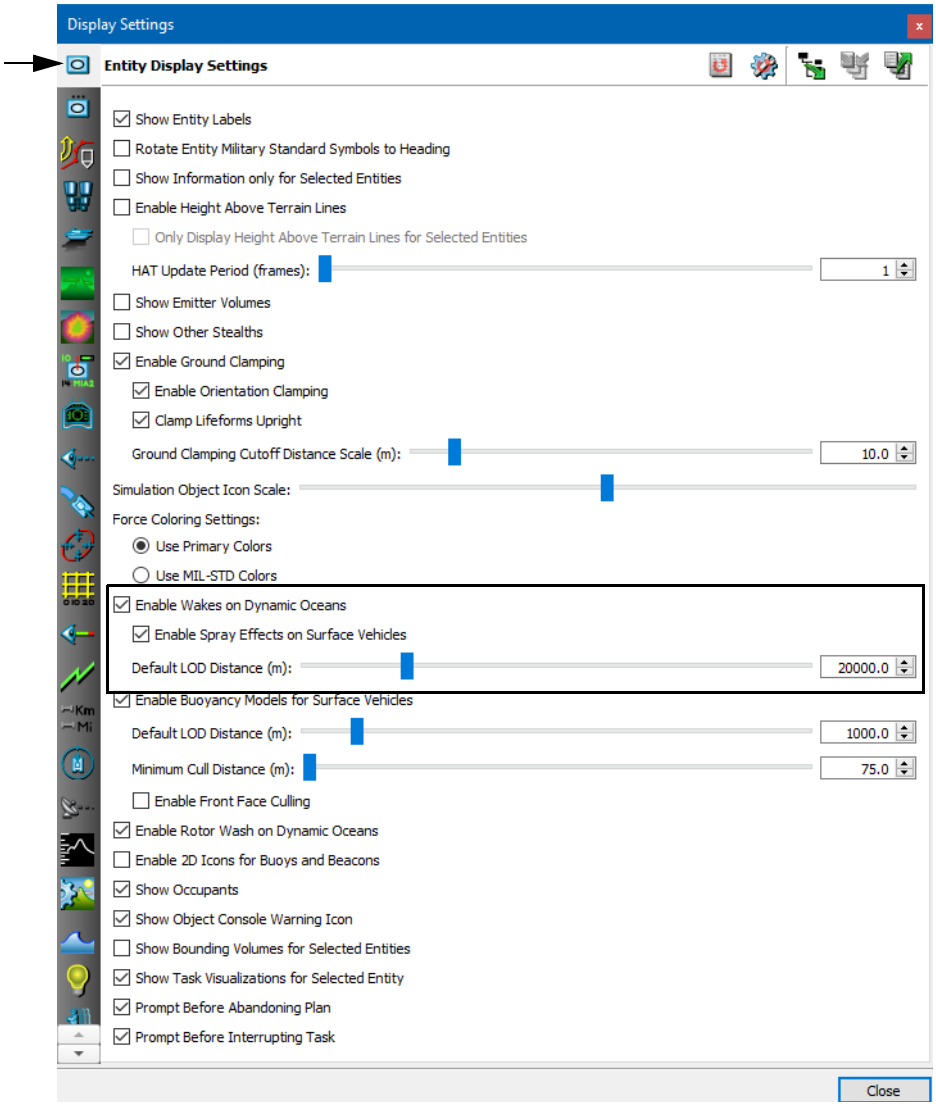


Figure 46-7. Entity Display Settings page

3. To enable or disable wakes, select or clear the Enable Wakes on Dynamic Oceans check box.
4. If wakes are enabled, select Enable Spray Effects on Surface Vehicles to enable spray effects. Clear the check box to disable them. Disabling wakes also disables spray effects.

46.3. Enabling Buoyancy for Surface Entities



When dynamic ocean is enabled, surface entities can bob up and down with wave action. However, the buoyancy models affect performance. You can disable buoyancy if you need to emphasize performance over visual effects. You can also control the area in which buoyancy models are calculated.

If buoyancy is enabled, VR-Forces calculates buoyancy only for models within certain distances of the observer, as follows:

- Buoyancy models are always calculated for entities within the minimum cull distance. This ensures that if the observer turns, entities that are relatively near will show buoyancy immediately.
- Buoyancy models are not calculated for entities that are beyond the maximum buoyancy LOD range. You can set this value on a per-entity basis in its entity model definition. If it is not set in the model definition, VR-Forces uses the default LOD distance specified on the Entity Display Settings page. If the default distance is set to 0.0, then entities that do not have an LOD range set in their model definitions will have an infinite buoyancy range (although buoyancy still will not be calculated past the far clipping plane).
- Buoyancy models are not calculated for entities beyond the far clipping plane or nearer than the near clipping plane.



VR-Forces uses the nearer of the far clipping plane or the maximum buoyancy LOD range to determine the distance at which it stops calculating buoyancy models.

When buoyancy is enabled, you can specify that VR-Forces use face front culling. VR-Forces calculates buoyancy models for all entities within the minimum range and all entities between the near clipping plane and the far clipping plane or maximum buoyancy LOD range, whichever is closest. In Figure 46-8, the shaded area is the area in which buoyancy models are calculated.

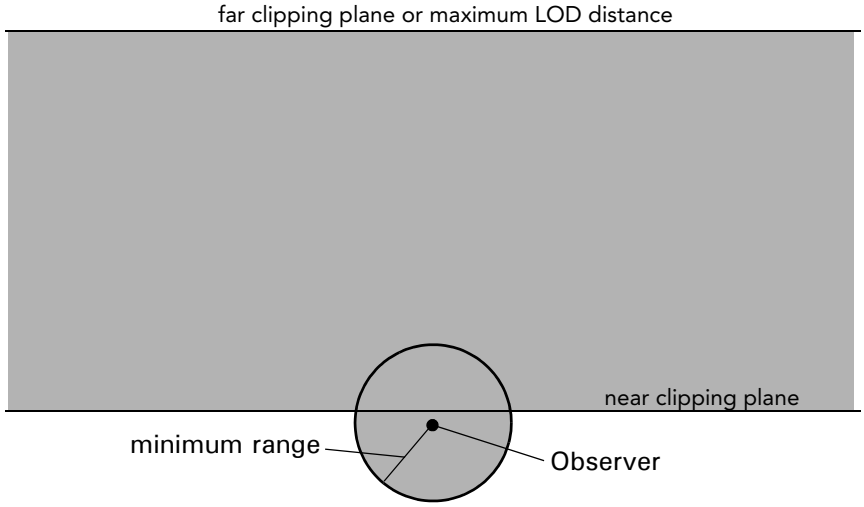


Figure 46-8. Face front culling

By default, face front culling is disabled and only distance culling is used.

To enable or disable the buoyancy models:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 46-7).
3. Select or clear the Enable Buoyancy Models for Surface Vehicles check box.
4. To specify the distance beyond which buoyancy models are not calculated, adjust the Default LOD Distance slider or change the value in the box. (If the default distance is set to 0, then entities that do not have an LOD range set in their model definitions will have an infinite buoyancy range. However, buoyancy will not be calculated past the far clipping plane.)
5. To specify the radius within which all entities have their buoyancy models calculated, adjust the Minimum Cull Distance slider or change the value in the box.
6. Optionally, specify use of face front culling.



47. Using CameraFX Sensor Views

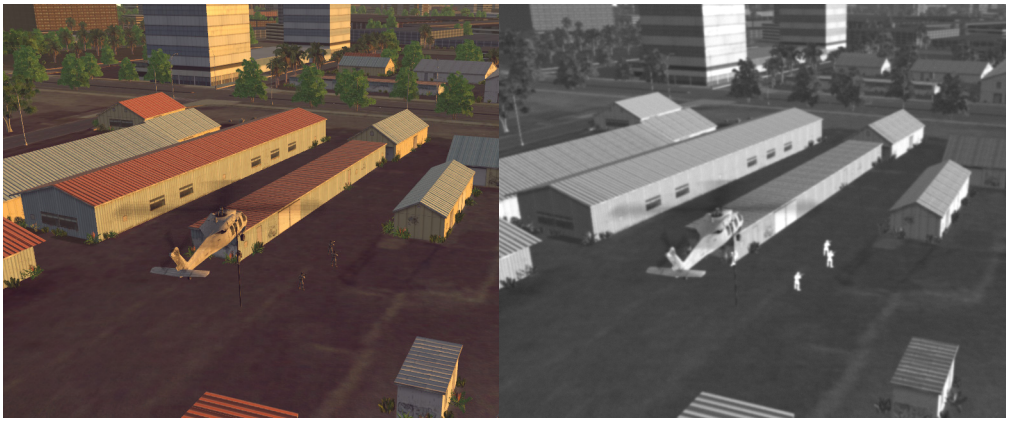
This chapter explains how to enable and configure simulated night-vision goggle (NVG) and infrared views of the scene.

Using Sensors 47-2
Enabling Sensors 47-3
Configuring Sensors 47-4
Adding New Sensors 47-5

47.1. Using Sensors

VR-Forces can simulate the way that the 3D scene would look if an observer were using visual sensor devices such as night vision goggles or viewing the infrared spectrum (Figure 47-1). This is an observer-specific setting. You can adjust the contrast, blur, and noise of the view.

VR-Forces includes one sensor module – CameraFX, which uses the SenSim libraries from JRM Technologies. The sensors do not take into account the materials of the objects that you are viewing. They simply filter the view to produce the desired effect.



No sensor (6 A.M.) IR sensor

Figure 47-1. CameraFX



The sensors described in this chapter are strictly visual effects. They have no relationship to the sensor systems used by simulation objects.

CameraFX does not affect the coloring of most tactical graphics.

47.2. Enabling Sensors

Sensors are observer-specific (Figure 47-2). VR-Forces includes observer modes that have sensors enabled. The names of the observer modes match the sensor modes that they enable.

- NVG (CameraFX) (night vision goggles).
- IR (CameraFX) (infrared).
- IR BlackHot (CameraFX).
- Camera (CameraFX). This mode uses the Visual Camera sensor mode.

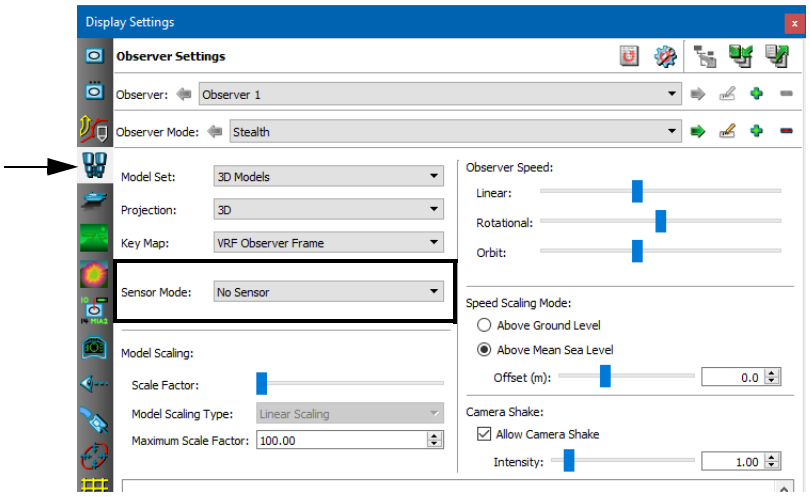


Figure 47-2. Sensor setting for observer mode

The default settings for the Camera observer mode make it look very similar to Stealth observer mode. The difference is that when you use the Camera observer mode, you can change the view by changing the sensor settings, such as blur, noise, and gain.

You can also specify use of a sensor configuration as a channel attribute.

To enable sensors, do one of the following:

- Select an observer mode that has a sensor configured.
- Edit an observer mode to use a sensor.
- Create a new observer mode and configure it with a sensor.
- Change the Sensor attribute for a channel.

For details about creating and editing observer modes, please see "Creating and Editing Observer Modes," on page 50-4. For details about editing channel attributes, please see "Changing a Channel's Attributes," on page 82-7.

47.3. Configuring Sensors

You can configure the following aspects of sensors:

- Blur. The sharpness of the image.
- Noise. The graininess of the image.
- Gain. The contrast and brightness of the image. (IR only)
- Lighting overrides. You can adjust the amount of ambient and diffuse light and the intensity of light emitted by DI-Guys and other entities.

You can configure a separate set of settings for each sensor type. VR-Forces remembers the settings. When you change the settings for a sensor type, they affect the display only if that sensor type is in use. For example, if you select NVG in the sensor list on the Observer Settings page, and select IR in the Sensors list on the Sensor Settings page, the changes you make to the IR configuration do not affect the NVG display you are looking at. If you then switch to IR in the VR-Forces window, the new IR settings take effect.

To configure sensors:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Sensor Settings page (Figure 47-3).

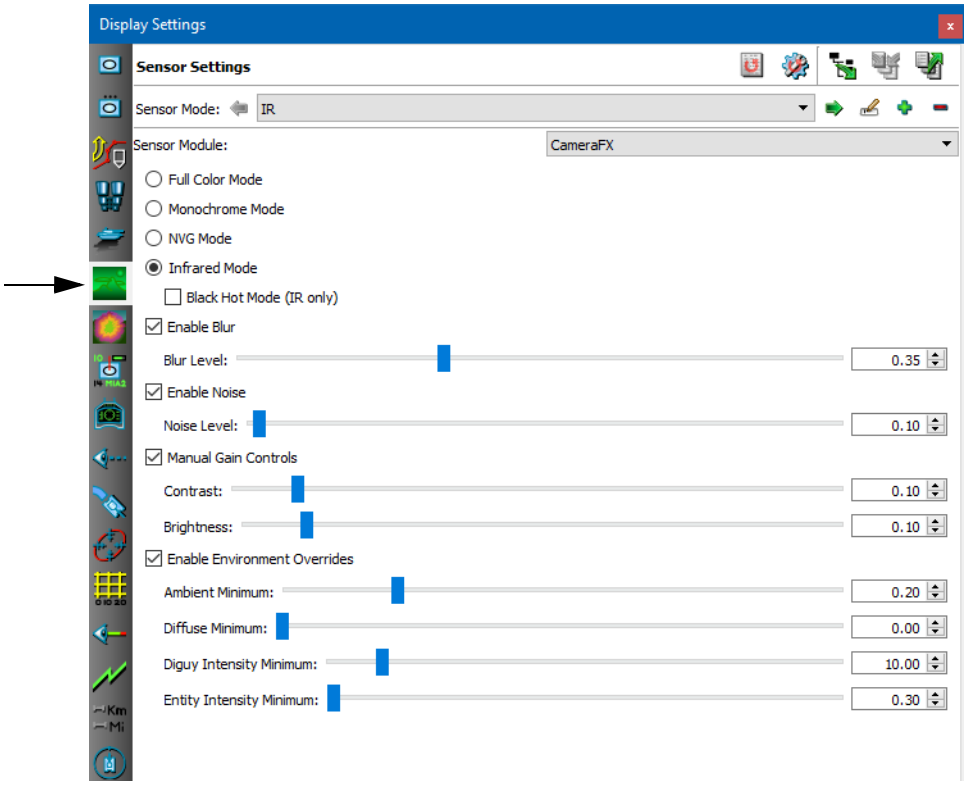


Figure 47-3. Sensor Settings page

3. In the Sensor Mode list, select the sensor that you want to configure.
4. Select the Sensor module from the list.
5. Select the sensor mode (Monochrome, NVG, and so on). If you select Infrared Mode, select or clear the Black Hot Mode check box.



- It may seem obvious that if, for example, you select NVG from the Sensor Mode list, you would select it for the sensor type. However, VR-Forces lets you add additional sensor configurations to the list and in that case, you would need to specify the correct sensor type for the new sensor. For example, you might create a sensor called NVG-High Contrast. Then you would select NVG as the sensor type and adjust the contrast accordingly.
- Full Color Mode provides natural coloring, as you would see with no sensor modes applied. The benefit of creating a sensor mode using this option is that you can apply noise or blur to the scene.

6. For each setting you want to change, select the controlling check box and adjust the sliders.

47.4. Adding New Sensors

You can add, rename, and delete sensors.

To add a sensor:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Sensor Settings page (Figure 47-3).
3. On the Sensor Mode line, click the Add button (). A new sensor called sensor_*number* is added. There are no configuration options on the page. (To rename the sensor, click the Rename button () and enter a new name.)
4. In the Sensor Module list, select CameraFX. The available configuration options are added to the page.
5. Select the type of sensor that you want this to be (Full Color, Monochrome, Infrared, or NVG).
6. Set the Blur, Noise, and Gain levels.



48. Intervisibility

This chapter explains how to use the intervisibility feature.

Intervisibility (Line-of-Sight) 48-2
Types of Intervisibility 48-3
Point-to-Point Intervisibility 48-3
Intervisibility Fan 48-4
Simulation Object Intervisibility 48-4
Intervisibility Objects can be Transient or Permanent 48-5
Creating a Transient Intervisibility Object 48-6
Creating a Permanent Intervisibility Object 48-8
Editing an Intervisibility Object 48-9
Configuring Intervisibility Lines and Fans 48-10
Pinning an Intervisibility Object to a Simulation Object 48-12
Showing and Hiding Intervisibility Lines 48-13
Displaying Intervisibility Line Information 48-14
Displaying an Intervisibility Line's Terrain Profile 48-15
Deleting Intervisibility Objects 48-16

48.1. Intervisibility (Line-of-Sight)

You can display intervisibility (line-of-sight or LOS) between points, within areas, and between a point and all simulation objects in a defined area. Intervisibility displays can be transient or permanent. Permanent intervisibility displays are objects. You can resize them and move them around the terrain. Intervisibility objects can be tied to a particular simulation object.

You can display a terrain profile for point-to-point intervisibility lines.

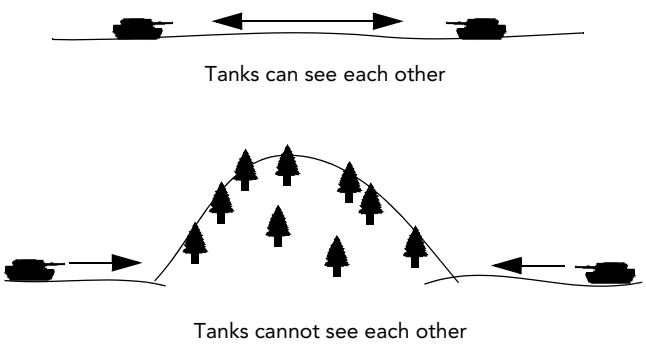


Figure 48-1. Intervisibility

You can configure parameters for intervisibility objects, such as the height above the terrain from which intervisibility is calculated. You can also change the colors used to display intervisibility.

- By default, intervisibility between simulation objects is calculated from the center of their bounding volumes. The VR-Forces target detection sensor uses an offset, which is often above the center of the entity. Unless you change the Origin LOS Elevation value in the Intervisibility Options dialog box to use the same offset as the target detection sensor, simulation objects may shoot at targets for which they do not appear to have line-of-sight.
- Linear and areal features do not block line-of-sight. Only point features block LOS.

48.2. Types of Intervisibility

VR-Forces supports the following types of intervisibility:

- Point-to-point (intervisibility line).
- Intervisibility fan.
- Entity intervisibility.

The intervisibility fan and entity intervisibility are displayed as a circular area. However, intervisibility calculations are made within a cylindrical area extending above and below the terrain, so that intervisibility calculations can be made for air platform entities and subsurface entities.

In the 2D view, intervisibility lines are fully visible on the terrain. In the 3D view, they may pass through the terrain.

48.2.1. Point-to-Point Intervisibility

Point-to-point intervisibility (Figure 48-2) shows the line-of-sight between any two points. The portion of the line that is visible from the starting point is green. The portion that is not visible is red. You can change the colors used.

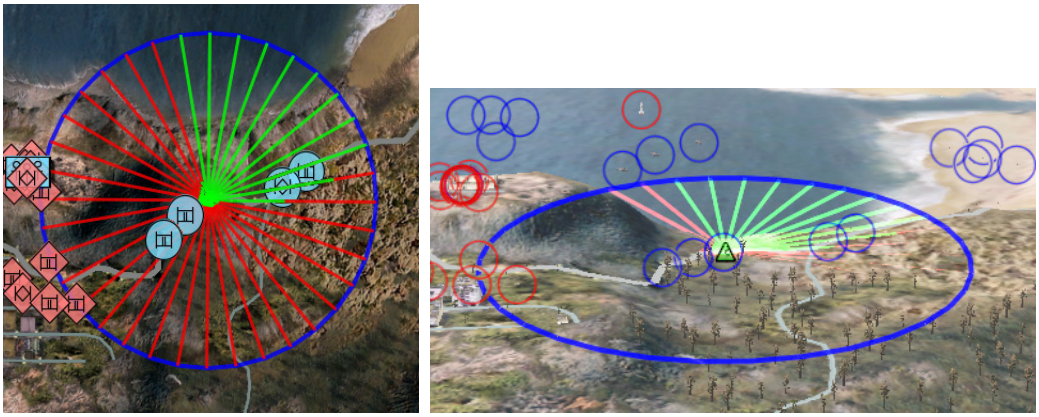


2D 3D

Figure 48-2. Point-to-point intervisibility

48.2.2. Intervisibility Fan

An intervisibility fan (Figure 48-3) shows a series of lines radiating from the center of a circle to its outer edge. Each ray shows intervisibility along its path from the origin to the border. You can configure the distance between rays. (For details, please see "Configuring Intervisibility Lines and Fans," on page 48-10.)



2D 3D

Figure 48-3. Intervisibility fan

48.2.3. Simulation Object Intervisibility

Simulation object intervisibility (Figure 48-4) shows which simulation objects are visible from the center of a defined area and whether or not the simulation objects can see the center point. If you center the intervisibility circle on a point on the terrain, you can see which simulation objects can see that point. If you lock the circle to a simulation object, you can see which simulation objects can see that object and which it can see. (For details, please see "Configuring Intervisibility Lines and Fans," on page 48-10.)



2D 3D

Figure 48-4. Simulation object intervisibility

48.3. Intervisibility Objects can be Transient or Permanent

You can configure intervisibility objects to be transient or permanent. (For details, please see "Configuring Intervisibility Lines and Fans," on page 48-10.) A transient intervisibility object displays intervisibility from the origin point (defined by a mouse click) to wherever you move the cursor. If you click the mouse again, you reset the origin. You cannot use the mouse for other actions until you exit intervisibility mode (by right-clicking). This mode is convenient when you want to quickly check intervisibility, but do not want to keep intervisibility objects in the display.

A permanent intervisibility object stays in the display after you create it. You can resize the object, edit drawing options, and move it around the display. Intervisibility objects are not saved as part of the scenario.

48.3.1. Creating a Transient Intervisibility Object

Transient intervisibility objects do not stay in the display after you quit intervisibility mode.

To display transient intervisibility:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 48-5).

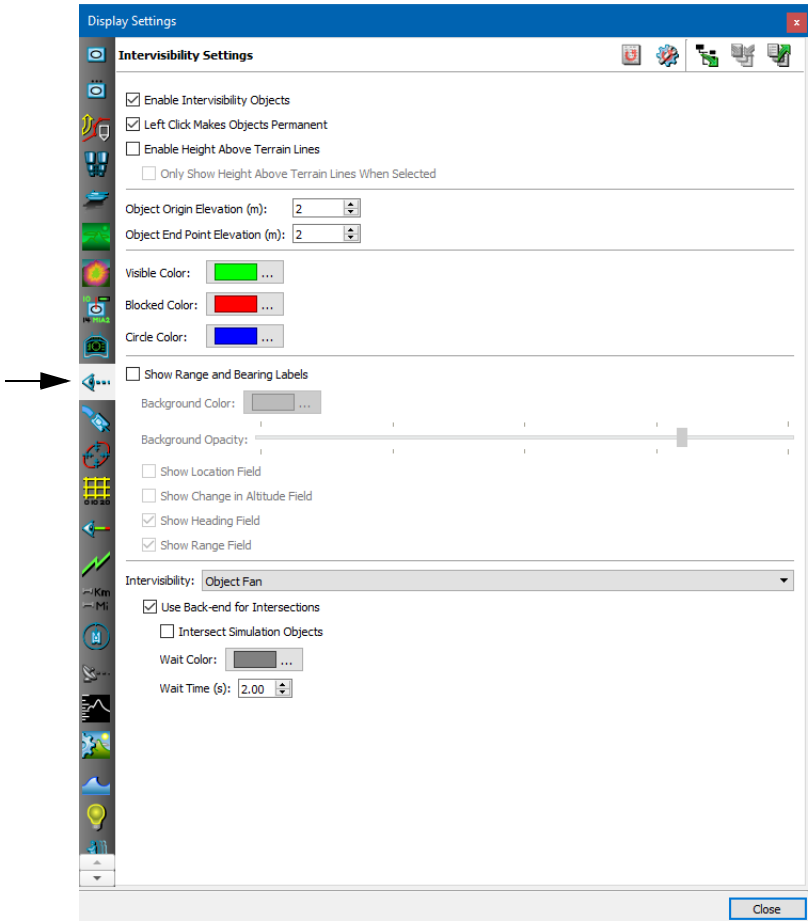


Figure 48-5. Intervisibility Settings page

3. Clear the Left Click Makes Objects Permanent check box.
4. Choose **Intervisibility** → **Create** *intervisibility_type*, where *intervisibility_type* is a type of intervisibility object, or click one of the buttons on the Intervisibility Toolbar.
5. Click on the terrain to place the origin of the intervisibility object.
6. Move the cursor around the terrain to expand and contract the object. For point-to-point intervisibility, moving the cursor also changes the direction of the line.

7. To change the origin of the intervisibility display, click another location on the terrain.
8. Right-click to cancel intervisibility mode.

48.3.2. Creating a Permanent Intervisibility Object

Permanent intervisibility objects remain in the display after you create them. (However, they are not saved as part of the scenario.)

To create a permanent intervisibility object:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 48-5).
3. Select Enable Intervisibility Objects.
4. Select Left Click Makes Objects Permanent.
5. Choose **Intervisibility** → **Create** *intervisibility_type*, where *intervisibility_type* is a type of intervisibility object, or click one of the buttons on the Intervisibility Toolbar.
6. Click on a simulation object or on a location on the terrain where you want the origin of the intervisibility object to be located. An intervisibility object is displayed. As you move the cursor closer to the origin, or farther away from it, the circle or line contracts and expands. If you are displaying point-to-point, you can rotate the line around the origin before you specify the end point.
7. Click on the terrain where you want to fix the end point of a line or border of a circle.

48.4. Editing an Intervisibility Object

You can change the following aspects of an intervisibility object:

- The location of either end point of an intervisibility line, to change its length and direction.
- The origin of an intervisibility fan, thereby moving the entire fan.
- The diameter of an intervisibility fan.



If an intervisibility object is anchored to a simulation object, you cannot change the origin.

To change the length or diameter of an intervisibility object:

1. To edit a line, double-click its end point.

   To edit a fan, select the object, then double-click the outer ring of the fan.

   The cursor changes (✣).
2. Drag the end point or circle towards or away from the origin. If you are editing an intervisibility line, you can move the end point to any location.
3. Click to exit edit mode.



To revert to the original size of the intervisibility object, right-click while you are still in edit mode.

To move an intervisibility fan or line:

1. Double-click a fan's center point or any place on a line. The cursor changes.
2. Drag the object to a new location.
3. Click to set the location.



To revert to the original location of the intervisibility object, right-click while you are still in edit mode.

48.5. Configuring Intervisibility Lines and Fans

You can configure the following attributes of intervisibility objects:

- The elevation of the origin and end point.
- The sampling rate for intervisibility fans.
- The update rate for entity intervisibility fans.
- Display of height above terrain lines (3D) and height labels (3D and 2D).
- The colors used to display intervisibility objects.
- Display of range and bearing labels. For more information about range and bearing labels, please see "Displaying Intervisibility Line Information," on page 48-14.

When you change colors or the sampling rate, changes affect all existing intervisibility objects. Changes to the origin and end point elevation only affect newly created objects.

VR-Forces can calculate intersections in the front-end or in the back-end.

To configure intervisibility objects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 48-5).
3. For Intersection configuration, select the intervisibility object to configure in the Intervisibility list.
4. Change the settings as desired. Table 48-1 describes the settings.

Table 48-1: Intervisibility settings

Option	Description
Enable Height Above Terrain Lines	Enables or disables display of height above terrain lines (3D) and height labels (3D and 2D).
Show Height Above Terrain Lines Only for Selected	If selected, HAT lines are displayed only for the selected intervisibility objects.
Object Origin Elevation	The height, in meters, above the terrain at the origin point from which to calculate intervisibility.
Object End Point Elevation	The height, in meters, above the terrain at the end point to use to calculate intervisibility.
Fan Line Sample Interval	The distance, in degrees, between each ray in a intervisibility fan. After changing this value, press **ENTER** or click away from the field.
Entity Fan Update Interval	How frequently, in seconds, to update the entity intervisibility fan.
Visible Color Blocked Color	The colors to use for the portion of an intervisibility line that a simulation object can see and the portion where visibility is blocked.

Table 48-1: Intervisibility settings

Option	Description
Circle Color	The color to use for the outer circle of an intervisibility fan.
Range and Bearing	
Show Range and Bearing Labels	Enable or disable display of range and bearing labels.
Background Color	Specify the background color for range and bearing labels.
Background Opacity	Specify the opacity of the background for range and bearing labels.
Show Location Field	If range and bearing labels are displayed, show the location of the endpoints.
Show Change in Altitude Field	If range and bearing labels are displayed, show the difference in altitude between the endpoints.
Show Heading Field	If range and bearing labels are displayed, show the heading of each endpoint relative to the other point.
Show Range Field	If range and bearing labels are displayed, show the length of the intervisibility line.

48.6. Pinning an Intervisibility Object to a Simulation Object

When you pin an intervisibility object to a simulation object, the intervisibility object moves with the simulation object and you know that its origin is centered on the simulation object.

To pin an intervisibility object to a simulation object:

1. Select the simulation object to which you want to pin the intervisibility object.
2. Choose **Intervisibility** → **Create** *intervisibility_type*, where *intervisibility_type* is a type of intervisibility object, or click one of the buttons on the Intervisibility Toolbar. The origin is placed on the simulation object icon or model.
3. Specify the end point for the intervisibility object.

48.7. Showing and Hiding Intervisibility Lines

You can toggle the display of intervisibility lines on and off.

To enable or disable the display of intervisibility lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 48-5).
3. Select or clear the Enable Intervisibility Objects check box.



You can also enable and disable display of intervisibility objects by choosing **Intervisibility** → **Enable/Disable Intervisibility**, or by clicking the Enable/Disable Intervisibility button () on the Intervisibility Toolbar.

48.8. Displaying Intervisibility Line Information

You can display the following information about an intervisibility line:

- Location of each endpoint.
- Heading of the line from each endpoint.
- Altitude of each endpoint.
- Length of the line (range).



Figure 48-6. Intervisibility line information (2D)

The information label has a background. You can adjust the color and opacity of the background.

To display information about an intervisibility line:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 48-5).
3. Select or clear the Show Range and Bearing Labels check box.
4. Select the check boxes for the type of information you want to display. For descriptions of the check box options, please see Table 48-1.
5. To change the background color for the label:
   a. Click the Background Color button. A color picker opens.
   b. Select the color you want to use for the background.
   c. Click OK.
6. To change the opacity of the background, adjust the Background Opacity slider.

48.9. Displaying an Intervisibility Line's Terrain Profile

By default, a terrain profile window is displayed automatically when you create an intervisibility line. The terrain profile can be very helpful when you try to understand how terrain changes are affecting intervisibility. Figure 48-7 shows an obstructed intervisibility line and its terrain profile, whose graph illustrates the line intersecting the terrain.

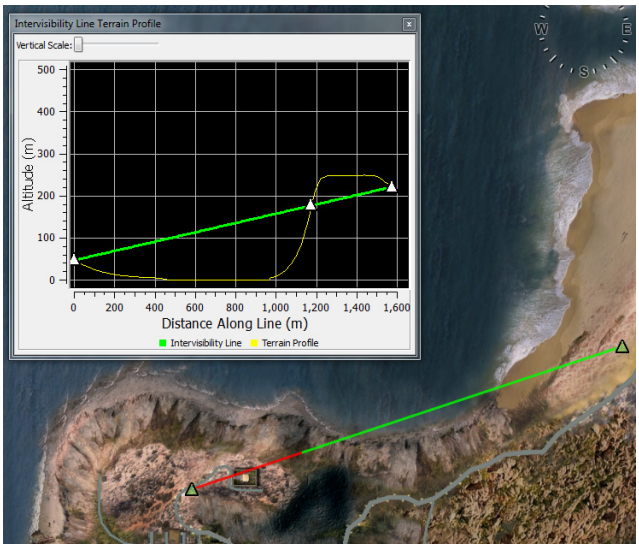


Figure 48-7. Terrain profile for an intervisibility line

You can disable this setting on the Display Settings dialog box, Terrain Profile Settings page. For details, please see "Configuring Terrain Profiles," on page 56-3.

48.10. Deleting Intervisibility Objects

Before you delete an intervisibility object, note the following:

- If you delete a simulation object that has an intervisibility object pinned to it, the intervisibility object also gets deleted.
- If an intervisibility object is pinned to a simulation object, you can delete the intervisibility object without affecting the simulation object by selecting the outer circle or end point and deleting it.

➤ To delete an intervisibility object, select the object and choose **Intervisibility** → **Delete**.



49. Sound Effects

VR-Forces can play sound effects for entities, fire events, and detonation events.

Introduction to Sound Effects 49-2
Enabling Sound Effects 49-2
Configuring Sound Effects 49-3
Associating Sound Files with Entities and Events 49-4

49.1. Introduction to Sound Effects

VR-Forces can associate sound effects with entities, fire events, and detonation events. You can enable and disable use of sound effects and you can specify settings that determine how sounds are perceived.

Using sound effects can degrade performance.

49.2. Enabling Sound Effects

To enable sound effects:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens (Figure 49-1).

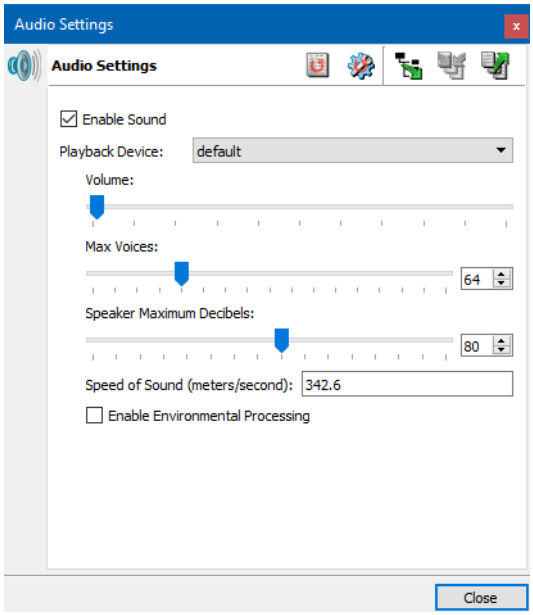


Figure 49-1. Audio Settings page

2. Select the Enable Sound check box.
3. Optionally, specify the playback device.

You can also enable and disable sound effects by clicking the Audio button ( ) on the Display Settings toolbar.

49.2.1. Configuring Sound Effects

You can configure the following sound settings:

- Volume. This is the volume of the system.
- Max Voices. You can have from 1 to 256 voices, or maximum simultaneous playing sounds. Sounds that play at the same time are mixed into a single sound wave prior to being sent to the speaker. This mixing has a CPU cost. A high number of allowable voices can affect performance when there are many audio sounds trying to play.
- Speaker Maximum Decibels. Calibrates the audio system for the speakers that the system is played on. This affects how sounds at a distance fall off so that the sound loudness is realistically replicated. The default setting of 80db represents headphones or a desktop PC's speakers.
- Speed of Sound. This is the speed of sound in the atmosphere in meters/second. The default value is derived by using a standard day as parameters (20 C, 50% relative humidity, 1013.25 mbar/29.92 inHg). This affects the delay in playing sounds at a distance. For example, an explosion 2km away has a several second delay from the time you can see the visual effect to when you can hear the audio effect.
- Enable Environmental Processing. Enables low pass filtering. As sound propagates through air, higher frequency sounds get absorbed more the further you move from the source of the sound. This setting lets you replicate air absorption of a sound based on the distance the sound is from the listener.

To configure sound settings:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens (Figure 49-1).
2. To set the volume, adjust the Volume slider.
3. To specify the maximum number of voices, adjust the Max Voices slider or change the value in the text box.
4. To set the speaker maximum decibels, adjust the Speaker Maximum Decibels slider or change the value in the text box.
5. To set the speed of sound, type a value in the Speed of Sound box.
6. To enable lowpass filtering, select the Enable Environmental Processing check box.

49.3. Associating Sound Files with Entities and Events

VR-Forces includes a set of sound files (WAV format) for generic entity platforms and for fire and detonation events. The files are in *./data/Audio/Sounds*. You add and edit mappings between entity types, fire events, or detonation events and sound files in the Visual Models Editor. Sound files are associated with model definitions. For example, the **filename** parameter for the heli_blades model definition specifies *$(DATA_DIR)/Audio/Sounds/heli_blades.wav*. The Rotary Wing element definition has a parameter called **Entity Sound**. The value for the **model definition** attribute is `heli_blades`.

You can quickly see which model definitions are available for sounds by filtering the list of the Visual Model Editors, Model Definitions Editor page for the Sound schema.



X. The Observer and Viewing Scenarios

The observer, or eyepoint, is the location in the 3D or 2D environment from which you observe the scene. You can move the observer (navigate) through the scene. You can attach the observer to a simulation object or prop. When the observer is attached to a simulation object, it moves automatically with that object. You can also control the observer from a remote application using view control messages.

You can have multiple observers in a VR-Forces session. They might be associated with different windows. For example you could have an observer for the main window and one for an inset window.

The observer is such a pervasive component of the VR-Forces front-end that it is not covered in just one chapter of this manual. For more information, please see:

- Chapter 50, *The Observer and Observer Modes*.
- Chapter 51, *Moving the Observer*.
- Chapter 52, *Attaching the Observer to Objects*.



50. The Observer and Observer Modes

This chapter explains how to create, edit, and change observers and observer modes. For details about moving the observer, please see Chapter 51, *Moving the Observer*. For details about attaching the observer to simulation objects and props, please see Chapter 52, *Attaching the Observer to Objects*.

Switching Between the 2D and 3D Projections 50-2
Observer Modes 50-2
Changing the Observer Mode 50-3
Creating and Editing Observer Modes 50-4
- Creating an Observer Mode 50-4
- Deleting an Observer Mode 50-5
- Editing an Observer Mode 50-5
- Renaming an Observer Mode 50-6

Adding an Observer 50-6
- Adding an Observer on the Observer Settings Page 50-7
- Adding an Observer in the Display Engine Configuration Editor . 50-7

Removing an Observer 50-9
Introduction to XR Mode 50-10
Model Sets 50-11
- Changing the Model Set 50-11

Communicating Information About the Observer 50-11
- Making the Observer Visible to Other Applications 50-11
- Viewing and Attaching to Other Observers 50-12

Displaying Models for Remote Observers 50-12

50.1. Switching Between the 2D and 3D Projections

VR-Forces supports both 2D and 3D projections. You can switch between them simply by changing the observer mode from Stealth or Out-the-Window, to Plan View. (Changing observer modes is described in "Changing the Observer Mode," on page 50-3.) When you go from a 3D observer mode to 2D, VR-Forces calculates a zoom level so that the observer should appear as if it is roughly the same height above the terrain. When you go from a 2D observer mode to 3D, VR-Forces calculates an altitude to do the same thing, and keeps the 3D view pointing down (so the change in overall view stays fairly small).

50.2. Observer Modes

Observer modes let you configure VR-Forces settings that affect how the observer moves and what it sees. You can then quickly change from one group of settings to another by switching between observer modes. An observer mode specifies:

- A model set. A collection of models of a particular type, such as 3D models.
- A key map. A coordinated set of key mappings to movement functions.
- A projection (2D or 3D).
- Settings for simulation object graphics and observer movement.

VR-Forces includes the following observer modes:

- Stealth. Stealth observer mode is a 3D projection configured for situational awareness.
- Out-the-Window. Out-the-Window observer mode is configured for when you want to use VR-Forces as a desktop image generator. It hides many of the explanatory visual effects present in the Stealth observer mode.
- Plan View. The Plan View observer mode provides a 2D "big picture" view of the terrain.
- XR. XR, or exaggerated reality, observer mode is a 3D mode that helps you understand the relationships between entities and the terrain when the observer is too far away from entities to view them at realistic sizes. It allows you to exaggerate entity size and terrain.
- IR BlackHot (CameraFX). A view using the CameraFX IR (infrared) BlackHot sensor.
- IR (CameraFX). A view using the CameraFX IR sensor.
- NVG (CameraFX). A view using the CameraFX NVG (night vision goggles) sensor.
- Camera (CameraFX). A view using the Visual Camera CameraFX sensor.

You can edit observer modes and add new ones. For details, please see "Creating and Editing Observer Modes," on page 50-4.

50.3. Changing the Observer Mode

When you change the observer mode, terrain may be displayed differently and the model sets used to display simulation objects may change. For descriptions of the observer modes, please see "Observer Modes," on page 50-2.

To enable marine effects (dynamic ocean):

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-1).

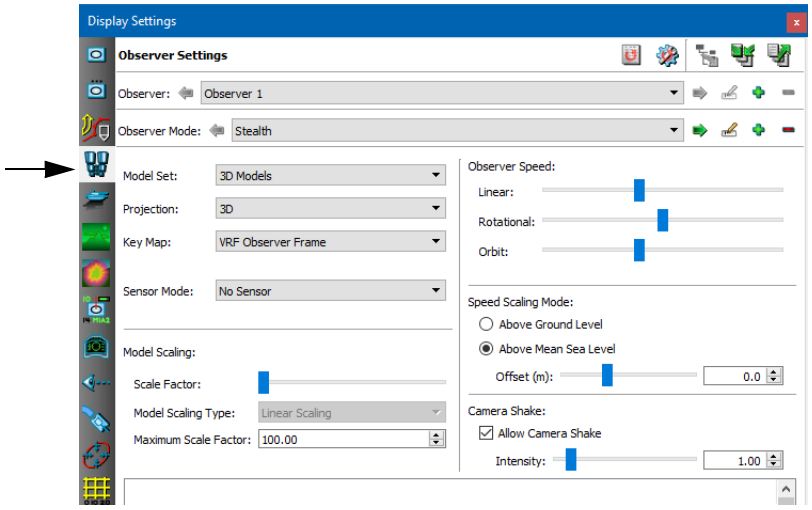


Figure 50-1. Observer Settings page

3. Select an option in the Observer Mode list.

> You can also change the observer mode by choosing **Observer** → **Set Observer Mode** → *mode*, or on the Observer Settings Toolbar, choose an option from the Observer Mode list.

50.4. Creating and Editing Observer Modes

VR-Forces comes with 3D and 2D observer modes. They are configured with the display settings, key mappings, and model sets that users most commonly want to use. However, if they do not meet your needs, you can easily edit them or create new observer modes.

50.4.1. Creating an Observer Mode

To create an observer mode:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).

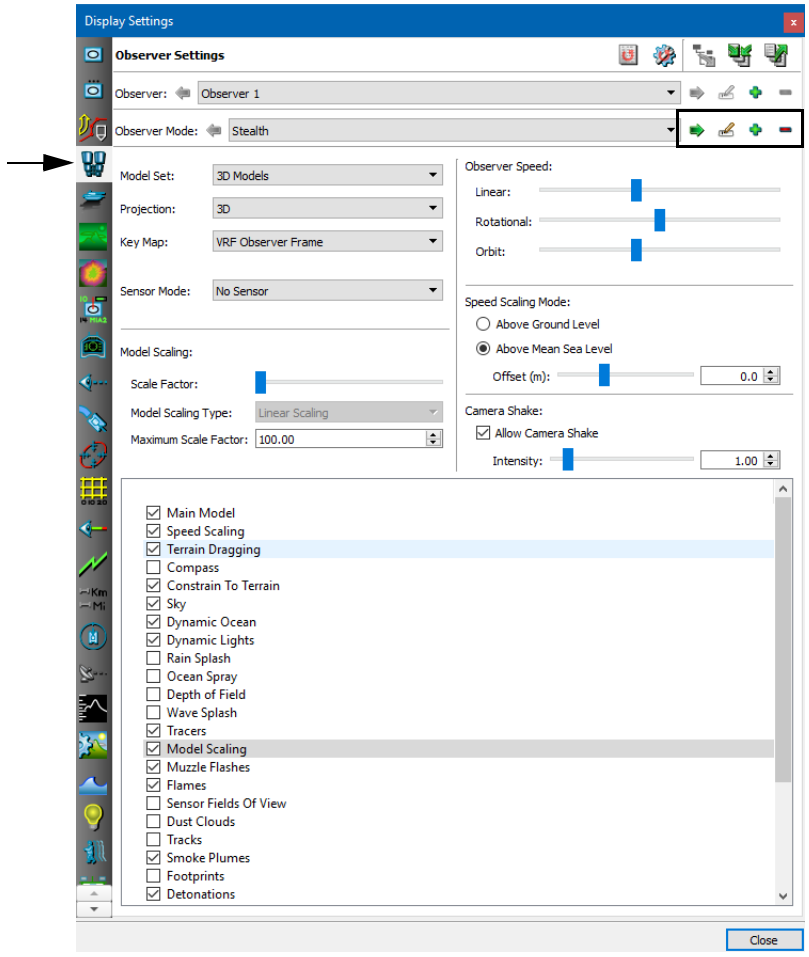


Figure 50-2. Observer Settings page

3. Click the Add button (+) to the right of the Observer Mode list. The Create New Observer Mode dialog box opens (Figure 50-3).

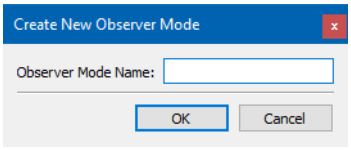


Figure 50-3. Create New Observer Mode dialog box

4. Type a name for the new observer mode.
5. Click OK.
6. In the Model Set list, select a model set.
7. Select a projection (2D or 3D).
8. Select the key map you want to use.
9. Optionally, specify a sensor mode.
10. Optionally, change the observer speed scaling settings and speed scaling mode.
11. Optionally, for 3D model sets, change the Model Scaling settings.
12. Enable or disable the mode-specific settings. The list of settings varies depending on the projection and model set.

50.4.2. Deleting an Observer Mode

To delete an observer mode:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. In the Observer Mode list, select the observer mode that you want to delete.
4. Click the Delete button (−).

50.4.3. Editing an Observer Mode

To edit an observer mode:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. In the Observer Mode list, select the observer mode that you want to edit.
4. Edit the attributes of the observer mode as desired.

50.4.4. Renaming an Observer Mode

To rename an observer mode:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. In the Observer Mode list, select the observer mode that you want to rename.
4. Click the Rename button (✍). The Rename Observer Mode dialog box opens.
5. Type the new name.
6. Click OK.

50.5. Adding an Observer

You can add and remove observers. Having multiple observers lets you assign different observers to different windows or channels. Changing the position and orientation of the observer in one channel does not affect the view of a different observer in a different channel. (For information about windows and channels, please see Chapter 82, *Display Engine and Window Management*.)



If you have multiple observers, the observer listed in the Observer Settings Toolbar is the observer assigned to the window that has focus. If you open the Display Engine Configuration Editor Panel and select a different observer in the Channel Attributes list, the data in relevant panels will change to reflect the values for the selected observer. However, as soon as you return focus to a window, the observer selection will change to match that of the window.

You can add observers in the Display Engine Configuration Editor Panel or on the Observer Settings page. (For information about the Display Configuration Editor Panel, please see Chapter 82, *Display Engine and Window Management*.)

If you want to save your observers, save the display engine configuration in which you created them. (Saving an observer just saves its name. It does not save any information about a particular observer mode or location that you may have used for that observer. To save location information, save an observer view.)

50.5.1. Adding an Observer on the Observer Settings Page

To add an observer on the Observer Settings page:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. Click the Add button (+) to the right of the Observer list. The Create New Observer dialog box opens.
4. Type a name for the new observer.
5. Click OK. The new observer is added to the list.
6. To save the observer, save the display configuration. (Saving the observer settings does not save observers.) (For information saving a display configuration, please see "Saving a Display Engine Configuration," on page 82-5.)

50.5.2. Adding an Observer in the Display Engine Configuration Editor

To add an observer on the Display Engine Configuration Editor Panel:

1. Choose **View** → **Display Engine Configuration Editor Panel**. The Display Engine Configuration Editor Panel opens (Figure 50-4).

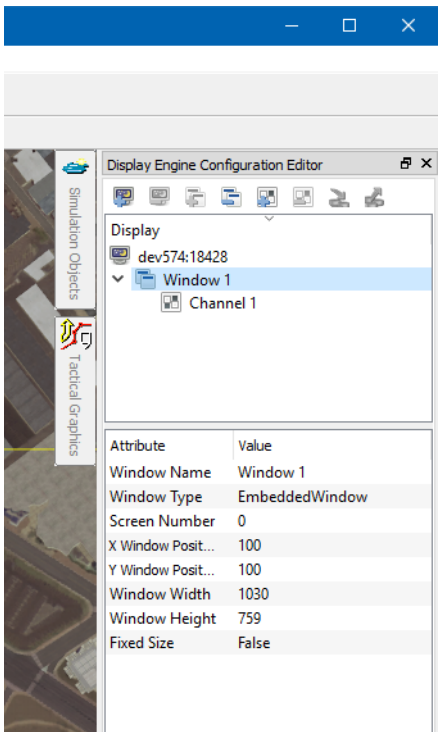


Figure 50-4. Display Engine Configuration Editor Panel

2. Select the channel on which you want to add an observer.
3. Click the Value field for the Observer Name attribute to make it editable.
4. In the list, select New (Figure 50-5).

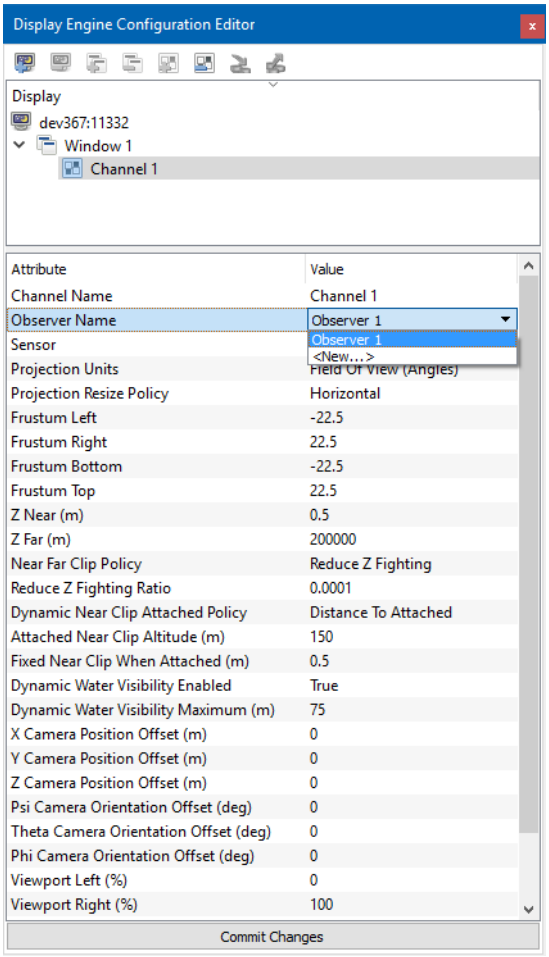


Figure 50-5. Observer Name attribute

5. Click Commit Changes. The Add Observer dialog box opens.
6. Type a name for the new observer.
7. Click OK. The new observer becomes the selected observer on the Observer Settings Toolbar, and is listed in the Observer Name attribute. (You may need to refresh the channel attributes before the observer name is listed.)
8. To save the observer, save the display configuration. (For information about saving a display configuration, please see "Saving a Display Engine Configuration," on page 82-5.)

50.6. Removing an Observer

When you close VR-Forces, all observers get removed. They are not saved unless you save a display configuration. You can also remove observers during a session.

To remove an observer:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. In the Observer list, select the observer that you want to remove.
4. Click the Delete button (−). The observer is removed.
5. If the observer was saved as part of a display configuration, save the updated display configuration.

50.7. Introduction to XR Mode

The Stealth observer mode (3D projection and 3D model set) provides situational awareness at the entity or small unit level. It is not ideal for getting the big picture of a simulation, because when you move the observer out to see large areas of terrain, the entity models, which are scaled to the terrain, are too small to see. The Plan View observer mode provides the big picture of a simulation, but does not provide situational awareness.

XR observer mode bridges the gap between the Stealth and Plan View observer modes. It is a 3D mode, so it provides situational awareness. However, it uses a special model set that makes models visible at greater distances than in Stealth observer mode. The 3D colorized model set shows entities as notional models using force coloring, which helps them stand out from the terrain. It also scales the model size to provided an exaggerated reality (hence XR). (You can also scale the regular 3D model set if you want to.) Figure 50-6 illustrates model scaling with 3D colorized models.

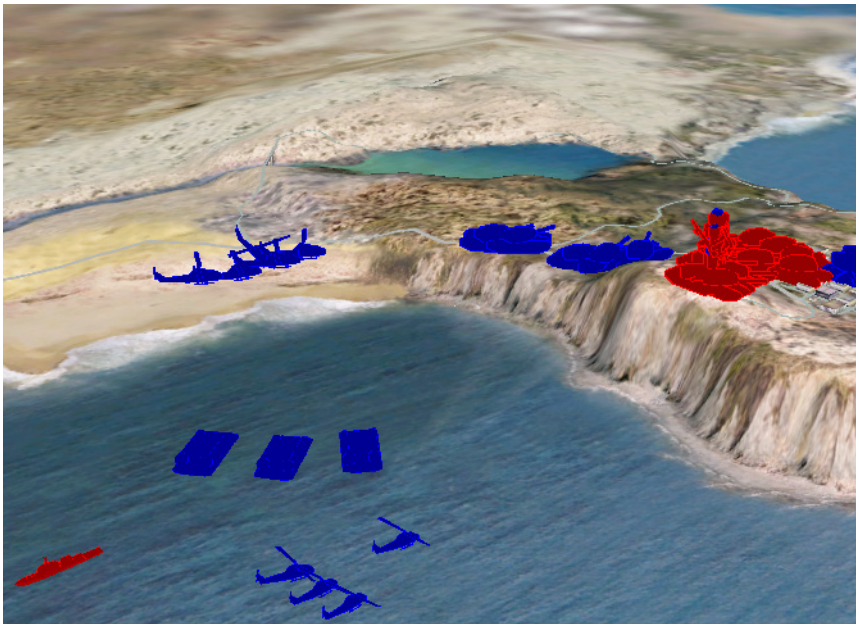


Figure 50-6. Model scaling and colorized models

For details about scaling models, please see "Exaggerating the Scale of 3D Entity Models," on page 44-5.

50.8. Model Sets

VR-Forces includes the following model sets:

- 3D Models. Realistic, 3D, textured models with articulated parts.
- 3D Colorized Models. 3D model caricatures with minimal articulation. These models do not have textures, they are solid colors, by force. You can change the coloring between primary colors and MIL-STD icon colors.
- 2D icons. MIL-STD 2525B icons.

You can use any of the model sets in both 3D and 2D views. In the 2D view, the 3D models can only be viewed from the top.

50.8.1. Changing the Model Set

Each observer mode specifies a model set. You can change the model set associated with an observer mode by editing it (for details, please see "Editing an Observer Mode," on page 50-5), but you can also dynamically change the model set being used.

➤ To change the model set, choose **Observer** → **Set Model Set** → *model_set*, where *model_set* is one of the options on the menu.

50.9. Communicating Information About the Observer

VR-Forces can transmit state updates describing the observer's position and orientation, and can respond to view control messages sent by other applications.

50.9.1. Making the Observer Visible to Other Applications

By default, VR-Forces sends state update messages to make its state information known to other applications. This makes it possible for a plan view display to represent VR-Forces as an icon, or for one VR-Forces application to synchronize its observer movement with that of another VR-Forces application.

➤ To enable or disable broadcasting of observer information, choose **Settings** → **Publish Observers** (or **Stop Publishing Observers**).

Making the Observer Visible to Just One Application (DIS Only)

In DIS you can send observer information to just one other application, rather than to everyone in the exercise. To do this, on the Simulation Connections Configuration dialog box, instead of using a broadcast address, set the destination address to the address of a specific computer.

50.9.2. Viewing and Attaching to Other Observers

If an exercise includes stealth applications, and they are sending state updates, VR-Forces can detect them. A remote observer entity is listed in the object hierarchy, so you can attach the local observer to it just as you would attach to any other simulation object.

Observer icons are displayed in the 2D view, but they are not normally visible in the 3D view, even when they appear in the entity list. If you want to display visual representations of remote observers, map a visual model to the observer's entity enumeration.

For related information, please see "Displaying Models for Remote Observers," on page 50-12.

50.10. Displaying Models for Remote Observers

VR-Forces broadcasts the location and orientation of the observer. By default, it does not display a model for the observer. However, if there are other applications in an exercise that are broadcasting a stealth PDU (such as another VR-Forces), you might want to represent them with a model.

To display a model for remote observers:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 50-7).

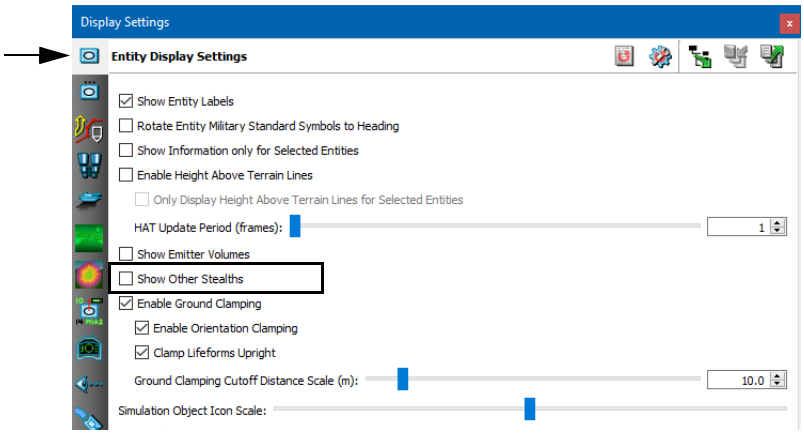


Figure 50-7. Display Settings, Entity Display Settings page

3. Select the Show Other Stealths check box.



51. Moving the Observer

This chapter explains how to move the observer (eyepoint) in the VR-Forces window.

Moving the Observer 51-3
Using the Compass to View the Observer's Heading 51-4
Editing the Compass's Attributes 51-5
Observer Movement Functions and Keyboard Mappings 51-6
3D Observer Movement Coordinate Systems 51-7
2D Observer Coordinate System 51-7
The Attached Context and Observer Movement (3D Only) 51-8
Moving the Observer from the Keyboard in the 3D View 51-10
Navigation Options 51-12
Orbiting 51-13
Moving the Camera 51-13
Moving the Observer Using the Mouse (3D View) 51-14
Dragging the Terrain 51-14
Changing the Observer's Orientation with the Mouse 51-16
Dragging the View 51-16
Orbiting the Terrain with the Mouse 51-16
Teleporting the Observer 51-17
Moving the Observer in the 2D View 51-18
Moving the Observer from the Keyboard in the 2D View 51-18
Moving the Observer Using the Mouse in the 2D Projection 51-19
Zooming the Observer 51-19
Zooming the Observer in the 3D Projection 51-20
Zooming the Observer in the 2D Projection 51-21
Setting the Scale Factor for Paged LODs 51-21
Setting the Magnification LOD Angle for Paged LODs 51-22
Zooming In to the Ocean Over Long Distances 51-23
Zooming to Extents 51-23

Centering the Observer on a Simulation Object 51-24
Centering on an Object from the Object Console Summary Panel 51-24
Centering on an Object From an Information Panel 51-24
Simulating Camera Jitter 51-25
Forcing Orientation North 51-26
Enabling and Disabling View Constraints (3D Only) 51-27
Controlling Navigation Speed 51-27
Enabling or Disabling Speed Scaling 51-28
Changing the Observer's Movement Speed 51-29
Saving and Recalling Views 51-30
Recalling a View 51-31
Deleting Views 51-32
Loading Views Automatically When You Load a Terrain 51-32
Specifying the Scenario Startup View 51-33
Saving Views to a File 51-34
Loading Views 51-34
Controlling the Observer from Other Applications 51-35
Enabling the Processing of View Control Messages 51-35

51.1. Moving the Observer

When you view a simulation, you usually want to move the observer (eyepoint) to different locations to view the terrain and scene elements being simulated. In VR-Forces, you can move the observer using the keyboard and the mouse. There are two basic contexts in which you can move the observer:

- You can move it independently of any scene element. This is called absolute, or free-fly mode.
- You can attach the observer to a simulation object, in which case it moves with the object. (You can also attach the observer to a prop. For details about attaching the observer to a simulation object or prop, please see Chapter 52, *Attaching the Observer to Objects*.)

Absolute mode and the various attachment types are collectively called attach modes.

Regardless of which attach mode you are using, you can change the position and orientation of the observer. In the 3D view, position is defined by X, Y, and Z coordinates that identify its location in the 3D world. Orientation describes the observer's yaw (heading), pitch, and roll (Figure 51-1).

In the 2D view, the Z coordinate is fixed. You can change the X and Y coordinates. You can change the observer's yaw, but pitch and roll are fixed.

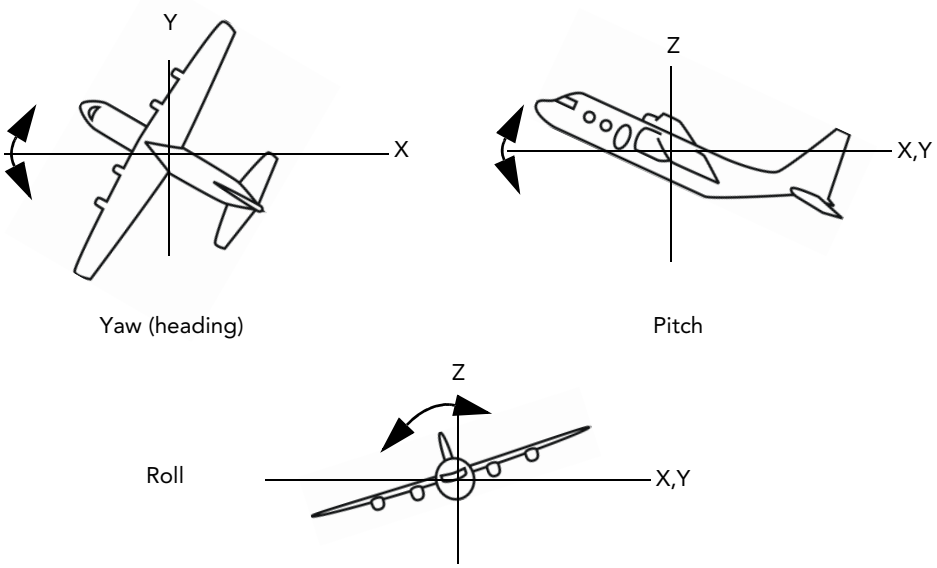


Figure 51-1. Orientation

The Observer Information Panel (Figure 51-4) displays the location and orientation of the observer in whichever coordinate system you select.

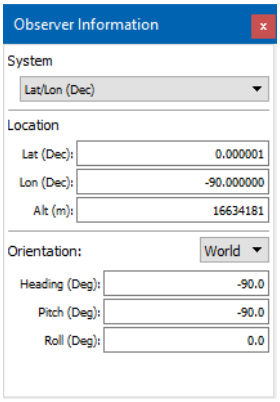


Figure 51-2. Observer Information Panel

51.1.1. Using the Compass to View the Observer's Heading

VR-Forces can display a compass that always shows the heading of the observer. The compass is an observer mode setting. In Figure 51-3, the observer's heading is approximately Northwest.

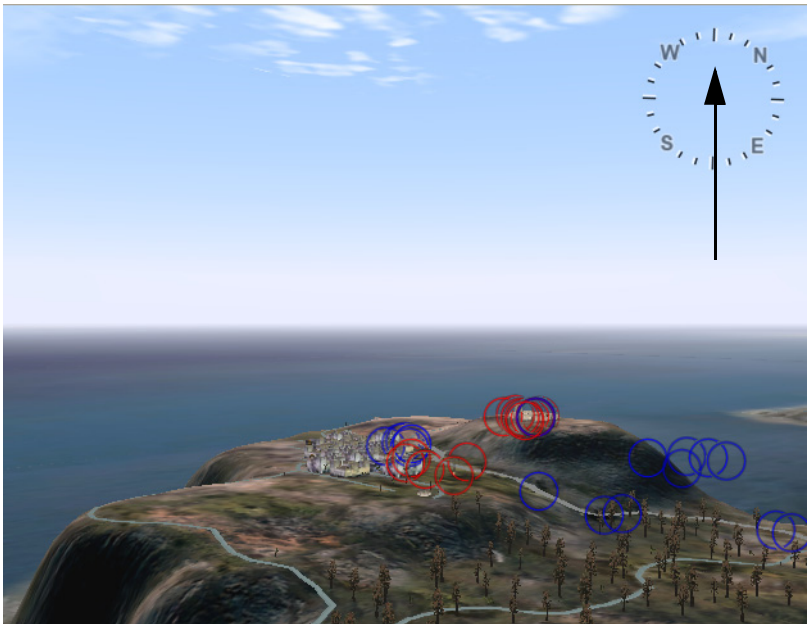


Figure 51-3. Compass

To enable or disable the compass:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. Select the observer mode for which you want to change the compass setting.
4. In the list of observer settings, select or clear Compass.



You can also enable or disable the compass by choosing **Observer** → **Compass**.

51.1.2. Editing the Compass's Attributes

You can configure the compass by editing its schema (ChannelCompass) and model definition (smallCompass and largeCompass). (For details about editing schemas, please see "Creating and Editing Schemas," on page 86-2.)

By default, all channels have the largeCompass keyword. To change the style of compass used for a channel, change the keyword in the Keywords attribute for the channel. To completely remove the compass, delete the keyword. For details about setting channel attributes, please see "Changing a Channel's Attributes," on page 82-7.

You can change where the compass is displayed in the window by editing the location attribute. The permissible values are top, topRight, right, bottomRight, bottom, bottomLeft, left, and topLeft.

You can change the color of the compass by editing the darkTickColor and lightTickColor attributes. The color is defined using the OpenGL scheme for specifying RGBA. The first three values are the RGB color values. The fourth value specifies opacity. Each value can be from 0.0 through 1.0.

51.2. Observer Movement Functions and Keyboard Mappings

When you move the observer, you are executing observer movement functions. An observer movement function is a specific way that the observer can move, such as Move Observer Forward or Move Level Observer Forward. (These functions both move the observer forward, but they use different observer coordinate systems. Observer movement coordinate systems are described in "3D Observer Movement Coordinate Systems," on page 51-7.)

The various movement functions are mapped to the keys on the keyboard and to mouse buttons and the mouse wheel. If you use the default key mappings for the default observer modes, then most of the time, the observer will behave exactly as you would expect it to. However for some types of movement, such as forward and back, the observer can move differently depending on the view, the observer coordinate system, or the attach mode. Therefore, VR-Forces provides several key mappings that let you choose which type of observer movement to use. You can change these key mappings and create your own key mappings. (For details about editing key maps, please see Chapter 85, *Creating and Editing Key Mappings*.) This section describes how observer movement can vary based on the observer coordinate system and the attach mode.

Table 51-1 lists the observer modes and key map settings provided with VR-Forces.

Table 51-1: Observer modes

Observer Mode	Projection	Key Map
CameraFX modes	3D	Observer Frame
Stealth	3D	Observer Frame
XR	3D	Observer Frame
Plan View	2D	2D Level Observer Frame
The observer modes all support the three model sets: 3D models, 3D colorized models, and 2D icons.		

51.2.1. 3D Observer Movement Coordinate Systems

In the 3D view, the observer moves in the context of one of the following coordinate systems:

- Level Observer Coordinates: Movement is with respect to the terrain. The level observer coordinate system is defined by the current up vector (Z), and a plane normal to that vector into which the observer's X and Y axes are projected. (Essentially, this means that the observer's heading is used while its pitch and roll are considered to be 0.0.)



Movement is always with respect to a plane tangent to the earth's surface. However, in a projected terrain database, such as UTM, Lambert Conformal Conic, or flat earth, the plane is fixed, whereas in a geocentric terrain database the up vector (and hence the tangent plan) changes as the observer's position changes, so movement follows the curvature of the earth.

- Observer Coordinates: Movement is with respect to the observer's body coordinate system.
- Entity Level Body Frame Coordinates: Movement is with respect to an entity's body frame, but ignores pitch and roll.
- Entity Body Frame Coordinates: Movement is with respect to an entity's body frame, including pitch and roll.

The practical difference between observer coordinates and level observer coordinates is that in level observer coordinates, if you are looking down at the terrain, when you move forward and backward, you fly over the terrain, looking down. In observer coordinates, when you move forward and backward, you move down towards the terrain or away from it at whatever angle you are oriented towards it.

51.2.2. 2D Observer Coordinate System

When you use the 2D view, the observer uses a variation of level observer coordinates. Movement is restricted to two dimensions, so there are no movement functions for changing the altitude, pitch, or roll. You can zoom in and out in the 2D view, but this does not change the observer's altitude, it changes the field of view and resolution of the terrain image.

51.2.3. The Attached Context and Observer Movement (3D Only)

The key maps control navigation using a particular coordinate system (observer frame or level observer frame). They also support navigation in alternate coordinate systems. The alternate system varies depending on the attached context of the observer. It represents MAK's opinion of which alternate coordinate system would be most useful given the attach mode that the observer is using.

The Move Attached Context *direction* (forward, back, up, down, left, right) functions move the observer in the context of its attach mode. Movement is modified by attach modes in the following ways:

- Follow mode: movement is in the entity's level body frame.
- Track mode: movement is in the entity's level body frame.
- Mimic mode: movement is in the entity's body frame.
- Tether mode: movement is in level observer frame.
- Unattached: movement is in level observer coordinates.

Figure 51-4 and Figure 51-5 illustrate the differences between moving the observer in an attached context and without an attached context.

Figure 51-4 illustrates movement in entity level body coordinates (Follow and Track modes). The Move Observer Left function (**A** in Observer Frame key map) moves the observer to the left in observer coordinates, that is left relative to the observer's heading. The Move Attached Context Left function (**SHIFT** + **A**) moves the observer left in the attached context. In entity level body frame, this means the observer moves left relative to the entity's heading.

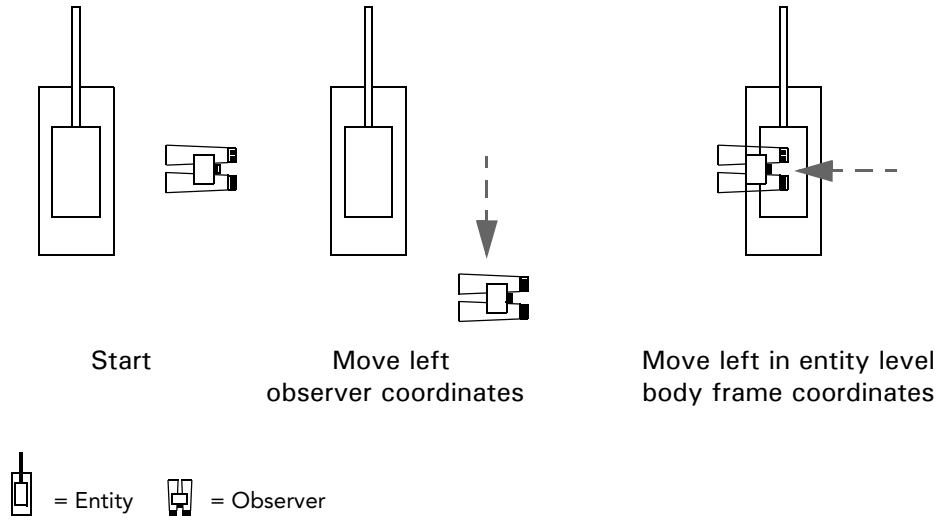


Figure 51-4. Observer movement in entity level body frame coordinates

Figure 51-5 illustrates movement in entity body frame (Mimic mode). The Move Observer Left function moves the observer to the left in observer coordinates, that is left relative to the observer's heading, but ignoring pitch and roll. The Move Attached Context Left function moves the observer left in the attached context. In entity body frame, this means the observer moves left relative to the entity's heading and its pitch and roll.

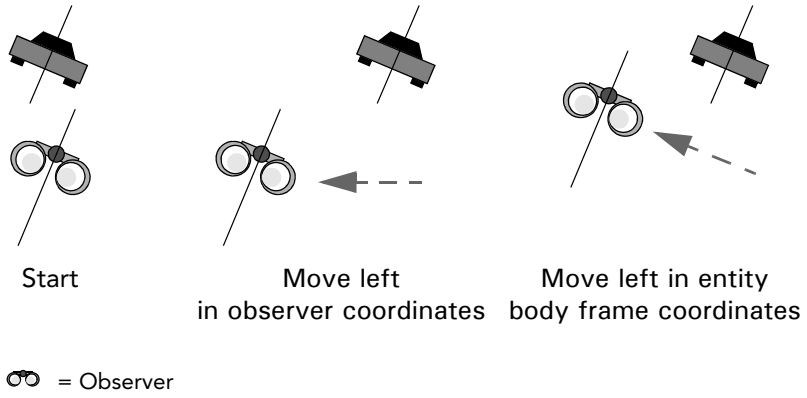


Figure 51-5. Movement in entity body frame coordinates

51.3. Moving the Observer from the Keyboard in the 3D View

VR-Forces maps the observer movement functions to the keyboard using key maps that support the most typical ways that users want to move through the 3D world, either in absolute mode or attached to a simulation object.

! References to movement key combinations use the key mappings provided with VR-Forces. If you change these key mappings, the instructions may no longer be valid.

As described in "Moving the Observer," on page 51-3, the observer typically moves with respect to the terrain in level observer coordinates or observer coordinates. The corresponding key maps are Level Observer Frame and Observer Frame.

Figure 51-6 illustrates the positional movement keys. Figure 51-7 illustrates the keypad layout for changing orientation and movement speed. For a complete list of movement functions and their keyboard and mouse mappings, please see *VR-Forces Quick Reference Card* or the Key Mapping Editor.

! To use the keypad for navigation, NumLock must be on.



Figure 51-6. Primary movement keys, observer coordinates (3D)

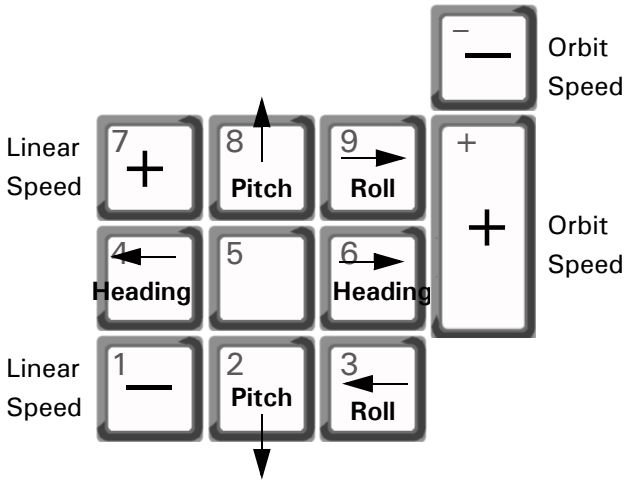


Figure 51-7. Keypad layout, heading, pitch, and roll (3D)

If your keyboard does not have a keypad, as on most laptops, the Laptop key map lets you use the arrow keys to control the left, right, up, and down movement mapped to 4, 6, 8, and 2 on keypads.

➤ To move the observer, press a key combination and hold it for as long as you want the observer to keep moving.

You can press more than one key at a time to combine movement functions. For example, pressing the move forward and move left keys at the same time moves the observer diagonally left and forward.

51.3.1. Navigation Options

This section lists the different ways that you can move the observer using the keyboard. For a complete list of movement functions and their keyboard and mouse mappings, please see *VR-Forces Quick Reference Card*. You can:

- Move forward and backward (observer, level observer, entity, level entity, and attached context coordinates).
- Move up and down (observer, level observer, entity, level entity, and attached context coordinates).
- Move left and right (observer, level observer, entity, level entity, and attached context coordinates).
- Rotate (heading) left and right (observer and level observer coordinates).
- Rotate (pitch) up and down (observer and level observer coordinates).
- Rotate (roll) left and right (observer and level observer coordinates).
- Orbit left and right.
- Orbit up and down.
- Look (move camera) left and right.
- Look (move camera) up and down.
- Tilt (move camera) left and right.
- Reset the observer to the default position and orientation for the attached mode.
- Reset the camera to the default position and orientation.
- Zoom the view.

For details about orbiting, please see "Orbiting," on page 51-13. For details about moving the camera, please see "Moving the Camera," on page 51-13.

When you view a geocentric terrain and move the observer more than 6 million meters away from the earth's surface, the view is forced to focus toward the center of the earth. The practical effect of this mode is that the left and right movement keys orbit the eyepoint around the earth, rather than moving the entire earth to the left or right.

51.3.2. Orbiting

You can orbit the observer (revolve left, right, up, and down) around an object or the point on the terrain that is at the center of the window. (If the center of the screen is in the sky, orbit does not work.) When you orbit, the observer's heading continuously changes so that it always faces the point or object around which it orbits. You can change the camera to look at something other than the orbit point while you are orbiting.

To orbit around an object:

1. Attach the observer to the object.
2. Press the right or left arrow keys.

To orbit around a point:

1. If you are attached to a simulation object, detach (press **z**).
2. Press the right or left arrow keys.

51.3.3. Moving the Camera

By default, the observer always looks in the same direction as its heading. However, just as the turret on a tank can swivel and fire in directions other than its body's heading, the observer can look in a direction other than its heading. The ability to look around is enabled in the camera. You can move the observer's camera up, down, left, right, tilt left, and tilt right. (The default key mappings are **CTRL** + *Keypad number*, where *number* is 2, 3, 4, 6, 8, or 9.)

To use the keypad for navigation, **NumLock** must be on.



You can tell that the camera is moving, but the heading is not, by observing the compass. If you change the heading, the view of the terrain changes and the compass moves. If you move the camera, but keep the heading static, the view of the terrain changes, but the compass does not move.

➤ To move the camera, press the keys mapped to the camera movement functions.

51.4. Moving the Observer Using the Mouse (3D View)

You can change the observer's position and orientation with the mouse. (To see the full list of mouse mappings, view the Mouse Settings page on the Application Settings dialog box.)

Using the mouse, you can:

- Drag the terrain.
- Drag the view.
- Orbit the terrain.
- Change orientation.
- Teleport (jump).

51.4.1. Dragging the Terrain

Dragging the terrain combines the move forward, backward, left, and right functions.

To drag the terrain:

1. Click a point on the terrain and hold down the left mouse button.
2. Drag the mouse left, right, forward, and backward.



Dragging the terrain is most effective when you are moving large distances, such as several kilometers. To move small distances, use the keyboard.

Disabling Terrain Dragging

To enable or disable terrain dragging:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 51-8).

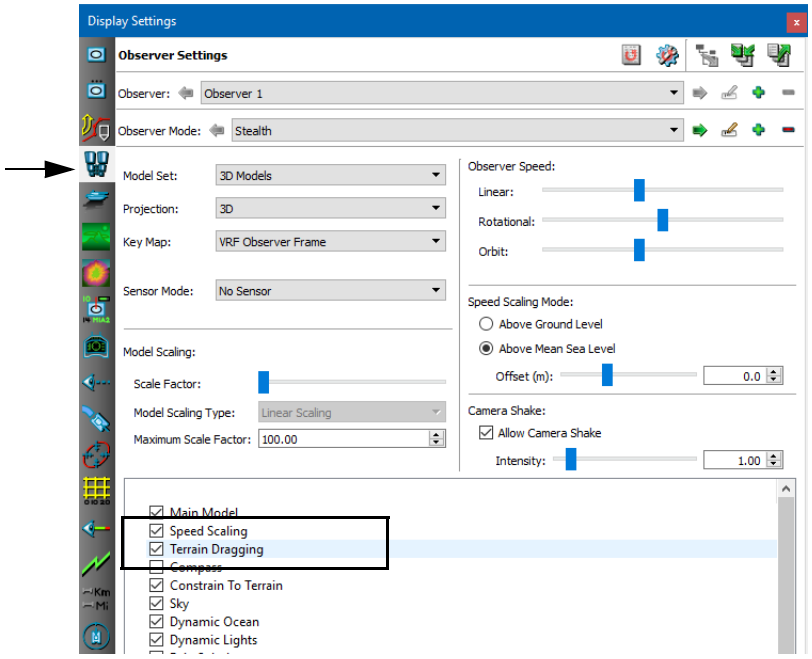


Figure 51-8. Observer Settings page

3. Select or clear the Terrain Dragging check box.



You can also enable or disable terrain dragging by clearing the Allow Terrain Dragging check box on the Observer Control Panel (Figure 51-9).

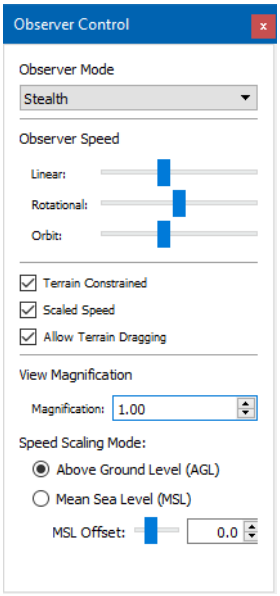


Figure 51-9. Observer Control Panel

51.4.2. Changing the Observer's Orientation with the Mouse

➤ To change the observer's orientation, hold down the right mouse button and drag the mouse. The observer rotates in the direction you drag the mouse.

51.4.3. Dragging the View

Dragging the view combines the observer up, down, left, and right movement functions. Unlike dragging the terrain, which requires you to click a point on the terrain, you can drag the view from any observer location.

➤ To drag the view, hold down the middle mouse button (or wheel) and drag the mouse.

51.4.4. Orbiting the Terrain with the Mouse

➤ To orbit the terrain with the mouse, press **CTRL** and drag the mouse.

51.4.5. Teleporting the Observer

You can teleport the observer to a simulation object, a prop, or a place on the terrain. Teleporting moves the observer 90% of the way from its current position to the location or object clicked on.

To teleport the observer:

- **SHIFT**+click on the terrain or prop that you want to jump to.



If you **SHIFT**+click on a filled area, the observer attaches to the area; it does not teleport. If you want to teleport to the terrain under an area, hide the area first. You can hide an area by hiding all tactical graphics or hiding the overlay on which the area is located.

- **CTRL**+**SHIFT**+click on the simulation object that you want to jump to.

51.5. Moving the Observer in the 2D View

In the 2D view, you move the observer using the mouse and keyboard just as you do in the 3D view. Many of the key and button mappings for the 2D view are the same as those in the 3D view. However, the 2D mappings do not support movement that is not meaningful in two dimensions, such as up and down (altitude), pitch, and roll.



When Orient North is enabled, you cannot change the heading of the observer. For information about Orient North, please see "Forcing Orientation North," on page 51-26.

51.5.1. Moving the Observer from the Keyboard in the 2D View

Figure 51-10 illustrates the positional movement keys for the 2D view. Figure 51-11 illustrates the keypad layout for changing heading and movement speed.

To use the keypad for navigation, `NumLock` must be on.

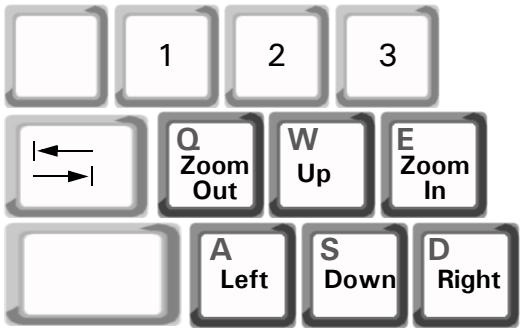


Figure 51-10. Primary movement keys, observer coordinates (2D)

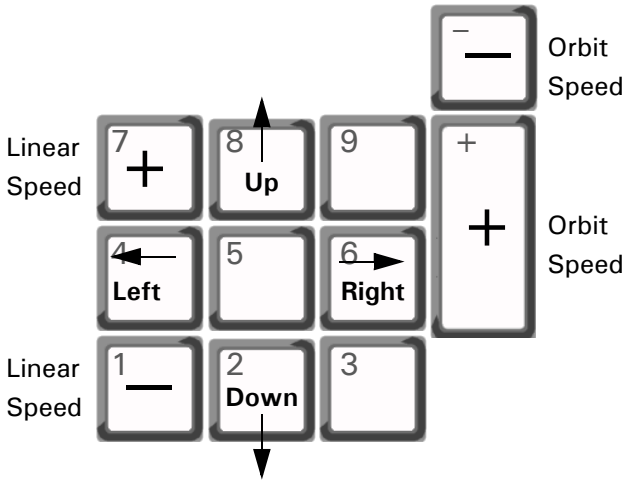


Figure 51-11. Keypad layout, heading, pitch, and roll (2D)

51.5.2. Moving the Observer Using the Mouse in the 2D Projection

You can change the observer's position and heading with the mouse. You can change the observer's position by dragging the terrain, as described in "Dragging the Terrain," on page 51-14. You can change heading by orbiting, as described in the next section. (To see the full list of mouse mappings, view the Mouse Settings page on the Application Settings dialog box.)

Orbiting the Terrain with the Mouse

When Orient North is enabled, you cannot orbit. For information about Orient North, please see "Forcing Orientation North," on page 51-26.

➤ To orbit the terrain with the mouse, press the right mouse button and drag the mouse.

51.6. Zooming the Observer

You can zoom the observer in the 3D and 2D projections. Although the apparent effect is similar for top-down views, they are conceptually and behaviorally different. In addition to a straight "in-and-out" zoom, you can zoom the view to the extents of a simulation object or group of simulation objects.

51.6.1. Zooming the Observer in the 3D Projection

If you want to see something at a higher level of detail without moving the observer, you can zoom the observer view. Zooming the observer narrows the field of view and magnifies objects in the view. The magnification level is saved in saved views. 3D zoom has the following characteristics:

- When you zoom the observer view, its movement and mouse look speeds are decreased as the magnification increases, to allow for increased control.
- When you switch between 3D and 2D, VR-Forces calculates a 2D zoom level to match the 3D view. If you then switch back to 3D, it remembers the observer location and zoom level.
- If you zoom in from a very high altitude there may be scene corruption due to insufficient precision in the z-buffer. This can be limited by moving the observer closer to the subject before zooming in. This issue is typically only encountered above 10,000 meters.
- When you zoom, VR-Forces respects the environmental conditions at the observer's location. For example, if the observer is above the cloud layer, zooming does not let you see below the clouds. Or, if it is raining at the observer's location, the visibility will not improve as you zoom in on a distant object.
- After zooming in, when you zoom out, you can only zoom out to the 0 magnification level. That is, you can only zoom out back to the observer's location. If you want to see more of the terrain, you must move the observer to a greater distance from the terrain.
- Some terrains are not designed to be viewed at higher magnification. Performance may decrease if magnification increases the level of detail of large areas of the terrain. For example, suppose you are looking down a line of trees that is in the center of the view. At the normal magnification level there are a certain number of trees at each quality level, say 10 high, 20 medium, and 100 low (for a total of 130 trees). Suppose also that performance will be acceptable only when visualizing 15 or fewer high quality trees. If you magnify the scene 2x, you may now see 20 high, 40 medium, and 70 low quality trees. You see the same number of trees, but the performance will decrease because you are visualizing more high quality models.

➤ To zoom in, press the **=** key, press **CTRL** and scroll the mouse wheel forward, or on the Observer Control Panel specify a value in the Magnification box.

➤ To zoom out, press the **–** (dash) key, press **CTRL** and scroll the mouse wheel backward, or on the Observer Control Panel specify a value in the Magnification box.

Each press of the **=** or **–** key, or increment of the mouse wheel, increases or decreases magnification by a factor of 1.25.

➤ To reset magnification to 1.0, press zero (**0**) or press the middle mouse button.

51.6.2. Zooming the Observer in the 2D Projection

In the 2D projection you can zoom the observer view closer to and away from the terrain. However, unlike in 3D, in which you cannot zoom out past the location of the observer, there is essentially no limit on zooming in and out because in 2D, the observer does not actually have altitude.

- To zoom in, press **e**, scroll the mouse wheel forward, or to step the zoom, press the **=** key.
- To zoom out, press **q**, scroll the mouse wheel backward, or to step the zoom, press the **-** (dash) key.
- To reset the observer to the default zoom level, press the **Spacebar**.

51.6.3. Setting the Scale Factor for Paged LODs

When you magnify the view, all distances used for determining LODs (including paged LODs) are scaled by the view magnification. In some cases, this is more aggressive than desired for the paged LODs. To remedy this problem, you can change the amount by which the paged LOD distance is scaled relative to the view magnification. For example, if view magnification is 8.0 and the Paged LOD Magnification Scale Factor is 2.0, then the distance used for paging would be scaled by 4.0 (8.0/2.0) rather than by the full factor of 8.0.

To set the scale factor for paged LODs:

1. Choose **Settings** → **Experimental Features**. The Experimental Features dialog box opens (Figure 11-13).
2. Adjust the Paged LOD Magnification Scale Factor slider.

51.6.4. Setting the Magnification LOD Angle for Paged LODs

When the observer zooms in, VR-Forces pages in higher LOD tiles for the terrain. This increases memory usage. You can tune performance by restricting high LOD tiles to the areas you are interested in and using lower LOD tiles for the rest of the terrain. The Paged LOD Magnification LOD Angle setting lets you adjust the angle within which higher LOD tiles are used (Figure 51-12).

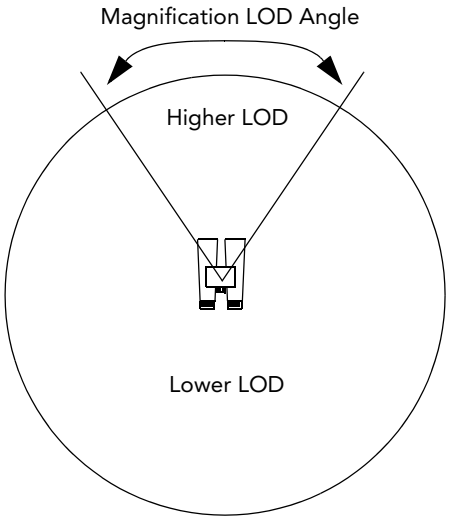


Figure 51-12. Paged LOD magnification LOD angle

If you want the observer to be able to quickly scan a wide angle of terrain, you should increase the magnification LOD angle or you may have to wait for high LOD tiles to page in as you move the observer.

On some terrains, when high LOD tiles are adjacent to low LOD tiles, there can be cracks in the terrain along the boundaries. If you experience this problem, increase the magnification LOD angle.

This setting is saved when you close VR-Forces.

To set the magnification LOD angle for paged LODs:

1. Choose **Settings** → **Experimental Features**. The Experimental Features dialog box opens (Figure 11-13).
2. Adjust the Paged LOD Magnification LOD Angle slider.

51.6.5. Zooming In to the Ocean Over Long Distances

If you zoom in to the ocean over very long distances, the ocean might not be rendered at the correct altitude and ships may be hidden (beneath the rendered ocean). If you experience this problem, in *./data/environment/ocean/Triton.config* set `use-mesh = no`. If after making this change you see performance decrease, the following settings will improve performance:

```
prop-wash-default-fade-time = 60.0
wake-wash-length = 1000
wake-wave-generation-distance = 1.0
wake-kelvin-wave-min-amplitude = 0.1
```

51.6.6. Zooming to Extents

You can zoom the observer to the extents of the terrain or selected simulation objects. Zoom to extents works as follows:

- If no simulation objects are selected and the observer is not attached to a simulation object, the observer view is reset to the terrain center. This is the same behavior as pressing **`Spacebar`**.
- If the observer is attached to a simulation object, the observer resets to the default attached view. This is the same behavior as pressing **`Spacebar`**.
- If simulation objects are selected and the observer is not attached to a simulation object, the observer view is redisplayed to show the extents of the selected simulation objects. In the 3D view, this is a top-down view.

Zooming the Observer to the Extents of Selected Simulation Objects

You can quickly change the observer view to show the extents of a selected simulation object or simulation objects. In 2D the observer zooms to the extents. In 3D the observer moves to the correct altitude to view the extents of the simulation objects while looking down at them.

To zoom the observer to the extents of selected simulation objects:

1. Select the simulation objects that you want to zoom in on. To zoom to the extents of all simulation objects, do not select any simulation objects.
2. Choose **View** → **Zoom to Extents**.

Zooming to Simulation Object Extents when a Scenario Loads

You can configure a scenario so that it zooms to the extents of all simulation objects when the scenario loads. This can be convenient when you have simulation objects located in one part of a large terrain.

If a scenario has a saved observer view that is designated as the initial view for the scenario, it overrides Zoom to Scenario Extents on Load.

To zoom to simulation object extents when a scenario loads:

1. Open the scenario that you want to zoom to extents on load.
2. Choose **Settings** → **Application**. The Application Settings dialog box opens.
3. Select the GUI Settings page (Figure 7-6).
4. Select the Zoom to Scenario Extents on Load check box.
5. Save the scenario. The next time you load the scenario, if the Zoom to Scenario Extents on Load check box is selected, it zooms to the entity extents.

51.7. Centering the Observer on a Simulation Object

You can center the observer on a simulation object from the Object Console Summary Panel, the Last Clicked Object Panel, or the Information dialog box. If you center on a simulation object from the Object Console Summary Panel, the observer gets attached to the object. It does not get attached using the other methods.

51.7.1. Centering on an Object from the Object Console Summary Panel

If you see that there is a message in the Object Console Summary Panel regarding a simulation object, you may want to quickly view that object. The Object Console Summary Panel lets you center the view on a simulation object. When you center the observer view on a simulation object, the observer attaches to it in Tether mode. For details about Tether mode, please see "Tether Mode," on page 52-13.

To center the observer on a simulation object:

1. In the Object Console Summary Panel, right-click a message from the simulation object you want to center on. A menu is displayed.
2. Choose **Center Entity**.

51.7.2. Centering on an Object From an Information Panel

You can center the observer on a simulation object from the Last Clicked Object Panel or the Information dialog box. When you center the observer this way, it does not get attached to the object. If the simulation object is moving, it moves away from the center of the display.

If you center on a simulation object in Plan View observer mode, the icon is centered in the window. If you center on a simulation object in Stealth observer mode, the observer stays in the current orientation relative to the centered object. If you press the **Spacebar**, the view changes to a top down view, similar to Plan View mode.



If the observer is attached to a simulation object, you cannot center the observer on a different object using this procedure.

➤ To center the observer on a simulation object from the Last Clicked Object Panel or the Information dialog box, click the simulation object icon on the panel (for example, Figure 51-13).

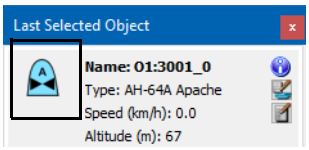


Figure 51-13. Simulation object icon

51.8. Simulating Camera Jitter

You can simulate the jitter that cameras attached to aircraft or unsteady human hands may experience.

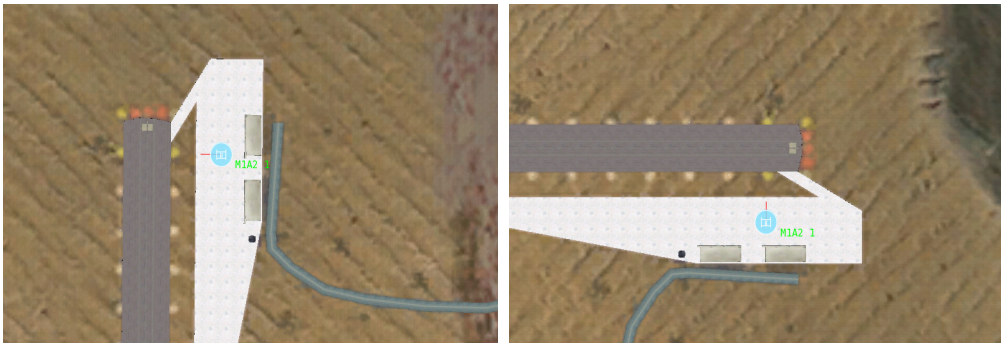
To simulate camera jitter:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. In the Camera Shake section, select Allow Camera Shake.
4. Optionally, adjust the Intensity slider or change the value in the box.

51.9. Forcing Orientation North

In the 2D view, by default the observer's heading is forced to stay North (up), regardless of the heading of the simulation object to which it is attached. You can disable this setting, in which case if you attach to a simulation object, the view of the terrain rotates to match the heading of the attached simulation object.

Figure 51-14 illustrates the view when the observer is attached to an entity with orientation forced to North or matched to that of the entity.



Orient North Orient to entity heading

Figure 51-14. Forcing orientation North

To enable or disable a northward orientation:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
3. In the Observer Mode list, select Plan View.
4. In the list of observer settings, select or clear Orient North.



You can also enable or disable northward orientation by choosing **Observer** → **Orient North**.

51.10. Enabling and Disabling View Constraints (3D Only)

View constraints control how the observer behaves when it approaches solid objects in the view, such as the ground, buildings, and vegetation. When constraints are enabled, the observer has a 1 meter bounding sphere that prevents it from moving through terrain or props.



If view constraints are enabled, you may not be able to move the observer into buildings to view and place entities. Changing the near clipping plane or transparency will let you see through walls, but in constrained mode, you will still not be able to move the observer through the walls.

To enable or disable view constraints, use one of the following methods:

- Press **/** (forward slash).
- Choose **Observer** → **Constrain to Terrain**.
- On the Observer Control Panel (Figure 51-9), select or clear the Terrain Constrained check box.

51.11. Controlling Navigation Speed

When you navigate through the terrain, there are times you want to move quickly, covering large amounts of terrain, and times you want to move slowly, such as when you are moving down a street or within a building. VR-Forces has the following options for controlling navigation speed:

- Set the navigation speed manually. The observer always moves at the specified speed.
- Use speed scaling to automatically move at different speeds under different circumstances. You can change the rate at which speed is scaled.

In either case, the controls for changing speed are the same. Their effect varies depending on whether or not speed scaling is enabled.

51.11.1. Enabling or Disabling Speed Scaling

When speed scaling is enabled, VR-Forces makes the following assumptions:

- If the observer is unattached, the higher above the terrain it is, the faster you want to move.
- The closer you are to an attached object, the more slowly you want to move.

To enable or disable speed scaling, use any of the following methods:

- On the Observer Control Panel (Figure 51-9), select or clear the Scaled Speed check box.
- Press the multiply key (*****) (not the asterisk key).

Setting the Speed Scaling Mode

When speed scaling is enabled, VR-Forces scales observer speed based on the its height above the terrain. Even if the observer is higher than all terrain points, speed can still be affected if there are changes to the terrain, such as mountains. This can be annoying if it happens unexpectedly. Therefore, you can choose to base speed scaling on height above ground level, or height above mean sea level. In that latter case, changes to the terrain would not affect the speed at which the observer moves. If you set speed scaling to use height above mean sea level, you can specify an offset.

To set the speed scaling mode, use one of the following methods:

- On the Observer Control Panel (Figure 51-9), set the Speed Scaling Mode to Above Ground Level (AGL) or Above Mean Sea Level (MSL)
- On the Display Settings dialog box, Observer Settings page, set the Speed Scaling Mode to Above Ground Level (AGL) or Above Mean Sea Level (MSL)

51.11.2. Changing the Observer's Movement Speed

You can change the observer's linear, rotational, and orbital speed. If speed scaling is enabled, these controls change the speed gain. If speed scaling is disabled, the controls change the absolute speed.

To change the observer's speed, use any of the following methods:

- Adjust one of the Observer Speed sliders on the Observer Control Panel (Figure 51-9) or on the Observer Settings page on the Display Settings dialog box. The default (center of the slider) is 1. The range is 0.1 to 10.
- Press one of the following keys to change linear speed:
  - **Keypad 7** (faster) or **1** (slower).
  - **CTRL** + mouse wheel forward or back.
  - Movement key + mouse wheel forward or back.
- Press **Keypad +** (faster) or **–** (slower) to change orbital speed.



When you change the observer's speed using the keyboard or mouse, if the Observer Control Panel or the Observer Settings page is open, you can see the scale slider moving to the new setting.

51.12. Saving and Recalling Views

At any given moment, the state of the observer is called its view. You can capture views so that you can quickly return to them. You can also save them for future use. VR-Forces automatically saves observer views as part of the scenario (*scenario_name.osrx*). You can also save views independently of scenarios. VR-Forces supports the following use cases for saving observer views:

- Associate views with a terrain. Save one or more absolute observer mode views and associate them with a terrain. The views get loaded automatically when the terrain is loaded. You can specify a default view that the observer jumps to immediately.
- Save a mix of absolute mode and attached mode views to a file. This use case is best when you want to create a set of views for use with multiple scenarios. You can import the file into each scenario that you want to use it with.



Nothing prevents you from including observer views that are attached to a simulation object in a terrain's saved views file, but they will not work unless the specific simulation object is available in the simulation.

If the observer is not attached to a simulation object, its view is its position and orientation in the simulated environment. If the observer is attached to a simulation object, a view saves the following information:

- Attach mode.
- Positional offset.
- Marking text.



When you save a view in the 2D view, the X and Y values are saved. The zoom level is not saved.

Saved views are managed in the Observer Views Panel (Figure 51-15).

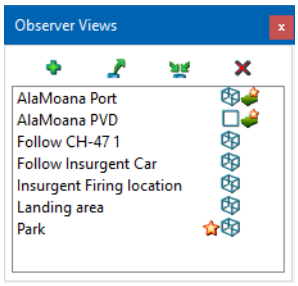


Figure 51-15. Observer Views Panel

Its buttons have the following meanings:

- Save the current view.
- Save the views to a file.
- Load a saved views file and append it to the list.
- Clear the list.

The icons in the observer views list have the following meanings:

- 3D view, such as Stealth, XR, or Out-the-Window.
- 2D view, such as Plan View.
- View is associated with the terrain.
- Default view for the terrain.
- Default scenario view.

Saved views are keyed to the marking text of the object to which the observer is attached.

To save the current view:

1. Choose **View** → **Observer Views Panel**. The Observer Views Panel opens (Figure 51-15).
2. Click the Add button ( ). The current view is added to the list.

51.12.1. Recalling a View

➤ To recall a view, select it in the Observer Views Panel. The view changes immediately.

Animating Transitions Between Observer Views

Normally, when you select an observer view, the observer jumps immediately to the new view. However, VR-Forces can smoothly transition from the current observer view to a saved view.

➤ To transition between observer views, hold down the **CTRL** key and click the view in the Observer Views Panel that you want the observer to move to.

If the observer view that you select uses a different observer mode (for example, from PVD mode to Stealth), VR-Forces does not change the observer mode. The view moves to an approximation of the the desired view in the current observer mode.

51.12.2. Deleting Views

To delete a view:

1. In the Observer Views Panel, select the view you want to delete.
2. On the shortcut menu, select **Delete Observer View**.

Clearing the View List

When you clear the list, VR-Forces removes all scenario-specific views and marks views that are associated with the terrain as hidden. When you load a scenario that has views associated with the terrain, VR-Forces ignores any views that are marked as hidden in the scenario. If additional views were associated with the terrain after the observer views list was cleared for the scenario, they are not marked as hidden and, therefore, get loaded. (Hidden views are recorded in the scenarios *.scn* file.)

➤ To remove all views from the list, in the Observer Views Panel, click the Clear List button (✖).

51.12.3. Loading Views Automatically When You Load a Terrain

You can associate a saved views file with a terrain so that it loads automatically when the terrain loads. You can also specify that one of the views be the default view. This is most useful when you create a new scenario. Once you save a scenario, the observer views saved with the scenario take precedence over the views associated with the terrain.

You associate views with a terrain by saving the views to a file in the same directory as the MTF file (by default, *./userData/terrains*). The file must have the same name as the terrain and the extension *.osrx*. The next time you open the terrain, the views are listed in the Observer Views Panel. An icon (🌲) next to each view shows that it is associated.

The views associated with the terrain do not get saved as part of the scenario. Each time you load a scenario, if there are views associated with the terrain, they get loaded (unless they are marked as hidden in the scenario file). Then the views saved in the scenario get loaded.

ℹ The file used for associating views with a terrain is different from the file that is created when you manually save views to a file, as described in "Saving Views to a File," on page 51-34.

To associate views with a terrain:

1. Click the Export Views button (🌲). A file browser dialog box opens.
2. Change directory to the directory where the terrain's MTF file is, usually *./userData/terrains*.
3. Save the views to a file that has the same name as the MTF file.

Specifying a Default Observer View for a Terrain

If you associate views with a terrain, you can specify that one of the views be the default view to use. A star is added to the association icon for default views ().

You can specify a default 3D view and a default 2D view. When you load a terrain, if there are two default views, and the current observer mode is a 3D mode, it uses the 3D default view. If the current observer mode is a 2D mode, it uses the 2D default view.

To specify a default observer view for a terrain:

1. Right-click the observer view that you want to be the default.
2. On the shortcut menu, choose **Set As Default Terrain View**.

Unsetting a Default View for a Terrain

To unset a default terrain view:

1. Right-click the default observer view.
2. On the shortcut menu, choose **Unset Default Terrain View**.

51.12.4. Specifying the Scenario Startup View

If a scenario has observer views, you can specify that one of them be the view selected when the scenario loads. (The observer view does not get selected when you rewind a scenario.) If a scenario has a default observer view and the terrain has a default observer view, the scenario observer view is used.

To specify a view as the startup view:

1. Right-click the view.
2. On the shortcut menu, choose **Set Default Scenario View**. A star icon next to the view shows that it is the default view.

Unsetting a Default Scenario View

To unset a default observer view:

1. Right-click the default observer view.
2. On the shortcut menu, choose **Unset Default Scenario View**.

51.12.5. Saving Views to a File

You can save views to a file independently of the scenario and load them at a later time. The default location for saved views is *./appData/settings/vrfGui*.

Saving views to a file is not the same thing as associating views with a terrain. When you associate views with a terrain, they get loaded automatically and you can specify a default view to use immediately. When you save views to a file, you need to load them manually, or from the command line. You cannot specify a default view.

Associating views with a terrain is most beneficial for saving a set of Absolute mode observer views that you always want to be available to the terrain, regardless of the simulation that you are connected to. Saving views to a file manually is most beneficial for saving views when the observer is attached to simulation objects in a simulation or for simulation-specific Absolute mode views.

To save all views to a file:

1. In the Observer Views Panel, click the Export Views button (). The Save Observer Views dialog box opens.
2. Select the file to which you want to save the views, or type the name of a new file.
3. Click OK.

Saving an Individual View to a File

You can save a single view to a file.

To save an individual view to a file:

1. Right-click the view that you want to save.
2. On the menu, choose **Save Observer View**. The Save Observer View File dialog box opens.
3. Type a name for the file.
4. Click Save.

51.12.6. Loading Views

If a scenario has observer views saved as part of the scenario, they are loaded automatically when you load the scenario. If you have saved views independently of the scenario, you can load them and append them to the list in the Observer Views Panel.

To load views and append them to the list of views:

1. In the Observer Views Panel (Figure 51-15), click the Load Views and Append button (). The Open Observer Views dialog box opens.
2. Select the saved views file that you want to load.
3. Click OK.

51.13. Controlling the Observer from Other Applications

The VR-Forces front-end can respond to view control messages sent by applications such as the MAK Logger. It also responds to view control messages sent from entity plans and global plans. You can write VR-Link-based applications to send view control messages to VR-Forces using the classes in the VR-Vantage Control Toolkit. You can use view control messages to:

- Specify the attach mode and simulation object.
- Set the observer's position and orientation.

Acceptance of view control messages is a global setting.



The view control PDUs are experimental, and are not an official part of DIS. They are of PDU kind 202.

51.13.1. Enabling the Processing of View Control Messages

➤ To enable or disable the processing of view control messages, choose **Settings** → **View Controls**, or click the View Control button () on the Display Settings Toolbar.



52. Attaching the Observer to Objects

This chapter describes attachment modes and how to attach the observer to simulation objects and props.

Attaching the Observer to a Simulation Object or Prop 52-2
Attaching the Observer to Another Observer 52-4
Attaching to an Articulated Part 52-5
Attaching the Observer in Mimic Track or Tether Track Mode..... 52-5
Attaching the Observer in Mimic Location Track or Tether Location Track Mode..... 52-5
Filtering the Object Attachment List 52-6
Detaching from a Simulation Object..... 52-6
Hiding the Attached Model (3D Projection Only) 52-7
Attaching to Props 52-7
Improving Performance when Attaching to Ground Objects 52-7
Attach Modes in the 3D View 52-8
Absolute Mode..... 52-8
Follow Mode 52-9
Mimic Mode..... 52-10
Mimic Track Mode 52-11
Mimic Location Track Mode..... 52-11
Space Follow Mode..... 52-12
Tether Mode 52-13
Tether Track Mode..... 52-14
Tether Location Track Mode 52-14
Track Mode..... 52-15
Attach Modes in the 2D View 52-16
Attaching the Observer at Startup..... 52-16

52.1. Attaching the Observer to a Simulation Object or Prop

You can attach the observer to a simulation object and follow it as it moves through the simulation. You can also attach the observer to props and to other observers.



This procedure applies to Follow, Tether, Track, Mimic, and Space Follow modes. To attach in Tether Track or Mimic Track mode, please see "Attaching the Observer in Mimic Track or Tether Track Mode," on page 52-5. To attach in Tether Location Track or Mimic Location Track mode, please see "Attaching the Observer in Mimic Location Track or Tether Location Track Mode," on page 52-5.

To attach the observer to a simulation object or prop, do one of the following:

- Right-click an object in the Objects List Panel (Figure 52-1) or on the terrain and select **Observer Attach** → *mode*, where *mode* is one of the supported attach modes. (For details about attach modes, please see "Attach Modes in the 3D View," on page 52-8 and "Attach Modes in the 2D View," on page 52-16.)

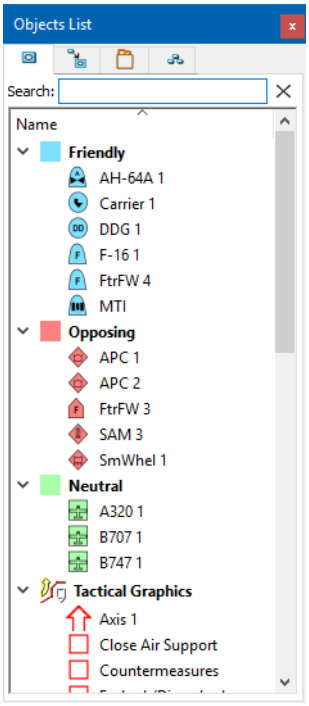


Figure 52-1. Objects List Panel

- Select a simulation object or prop. (For details about selecting simulation objects and props, please see "Selecting Simulation Objects, Tactical Graphics, and Props," on page 15-4.) On the Attachments Panel (Figure 52-2), or on the Attach Object Toolbar (Figure 52-3), click the Attach button ( ) to attach with the current attachment mode.

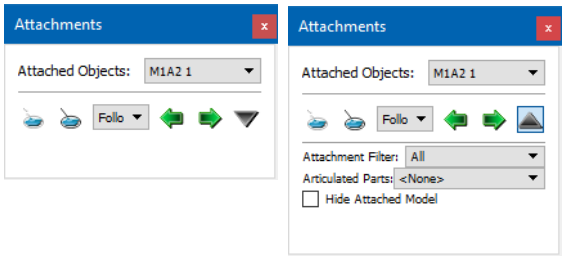


Figure 52-2. Attachments Panel



Figure 52-3. Attach Object Toolbar

- On the Attachments Panel or Attach Object Toolbar, click the next or previous arrows to attach to the next (or previous) object in the object list.
- **SHIFT**+click the object you want to attach to.
- Press greater than (**>**) to attach to the next simulation object or prop in the object list without regard to the attachment filter. Press less than (**<**) to attach to the previous object.
- Press period (**.**) to attach to the next simulation object or prop allowed by the attachment filter. Press comma (**,**) to attach to the previous object.

- When you select the next or previous object, VR-Forces chooses next and previous based on the order in which the objects were discovered. This may or may not represent the order in which they are listed in the Objects List Panel.
- When you move the attachment from one object to the next, VR-Forces preserves whatever changes you have made to the observer's view relative to the attached object. So if, for example, you moved the observer so that it is not looking at the attached object, when you attach to the next simulation object, the observer will not be looking at this object. It will be in the same relative position to the object that you previously established. To quickly return to the default offset for the attachment mode, press **Spacebar**.

52.1.1. Attaching the Observer to Another Observer

You can attach observers to other observers if you have more than one window open.

To attach an observer to another observer:

1. Filter the Objects List to show All objects or to show observers.
2. Select the window whose observer you want to attach to another observer.
3. In the Objects List, select the observer you want to attach to.
4. Right-click and select **Observer Attach** → *mode*, where *mode* is one of the supported attach modes.

52.1.2. Attaching to an Articulated Part

You can attach the observer to an articulated part on an entity, such as a turret. When you attach to an articulated part, the observer moves with the articulated part instead of the main body of the entity.

To attach to an articulated part:

1. Attach the observer to an entity, as described in "Attaching the Observer to a Simulation Object or Prop," on page 52-2.
2. On the Attachments Panel (Figure 52-2), click the down arrow. The Attachments Panel expands.
3. In the Articulated Parts list, select the part to which you want to attach the observer.

52.1.3. Attaching the Observer in Mimic Track or Tether Track Mode

To attach the observer in Tether Track and Mimic Track modes, you must specify a primary simulation object and a secondary simulation object (the tracked simulation object).

To attach the observer in Mimic Track mode or Tether Track mode:

1. Right-click the primary simulation object in the Objects List Panel or in the window. A shortcut menu is displayed.
2. Choose **Observer Attach** → **Attach Mimic Track** (or **Attach Tether Track**). The cursor changes to show an arrow and a Plus sign.
3. Click the secondary simulation object in the Objects List or in the window.

52.1.4. Attaching the Observer in Mimic Location Track or Tether Location Track Mode

To attach the observer in Tether Location Track and Mimic Location Track modes, you must specify a primary simulation object and a location.

To attach the observer in Mimic Location Track mode or Tether Location Track mode:

1. Right-click the primary simulation object in the Objects List Panel or in the window. A shortcut menu is displayed.
2. Choose **Attach Mimic Location Track** or **Attach Tether Location Track**. The cursor changes to show an arrow and a Plus sign.
3. Click a location in the display window.

52.1.5. Filtering the Object Attachment List

Although you can attach the observer to simulation objects and props, it is likely that at any given time you will only want to attach to one type of object (usually simulation objects). To reduce the number of objects that you have to iterate through when using Attachments Panel arrows to pick the next or previous object, you can filter the attachment list.



The filter for buoys and beacons applies to S-57 feature data, not buoys that are simulation objects, such as sonobuoys.

To filter the attachment list:

1. On the Attachments Panel (Figure 52-2), click the down arrow. The Attachments Panel expands.
2. In the Attachment Filter list, select the type of scene element that you want to attach to.

52.1.6. Detaching from a Simulation Object

➤ To detach the observer from the simulation object to which it is attached, on the Attachments Panel (Figure 52-2), click the Detach button (), or press **z**.

52.1.7. Hiding the Attached Model (3D Projection Only)

You can hide the model for the simulation object to which the observer is attached.

To hide the attached model:

1. On the Attachments Panel (Figure 52-2), click the down arrow. The Attachments Panel expands.
2. Select the Hide Attached Model check box.

52.1.8. Attaching to Props

Since props do not move, attaching the observer to a prop does not result in any observer movement. The principal benefit of attaching to a prop is to let you quickly refocus the observer on a particular location in the scene. If you attach in Track mode, using the keyboard movement keys allows you to orbit around the prop.

52.1.9. Improving Performance when Attaching to Ground Objects

When the observer is attached to air entities, Z-fighting can affect visual quality. To prevent this problem, VR-Forces needs to calculate the entity's height above the terrain. However, the calculation can affect performance. Unfortunately, when this calculation is enabled (the default), it must be applied to all entity types. If you know that you will want to attach the observer to ground simulation objects, but not air simulation objects (or if you do not care about Z-fighting), you can disable the height above terrain calculation, which may improve performance when the observer is attached to simulation objects. To disable the height above terrain calculation, start VR-Forces with the `-S | --suppressHatIntersectOnAttach` command line option.

52.2. Attach Modes in the 3D View

Attach modes are combinations of position and orientation models and constraints that produce a useful behavior for the observer. Except for Absolute mode, attach modes describe how movement of the simulation object to which the observer is attached affects the position and orientation of the observer.

VR-Forces supports the following attach modes:

- Absolute: The observer is not attached to a simulation object; it moves in space independently of all simulation objects.
- Follow: The observer position and heading changes with the simulation object's position and heading.
- Mimic: The observer position and orientation changes with the simulation object's position and orientation.
- Mimic Track: The observer's position changes with the simulation object's position while its orientation is focused on a second entity.
- Mimic Location Track: The observer's position changes with the simulation object's position while its orientation is focused on a location.
- Space Follow: The observer is a fixed distance from the simulation object and the simulation object model is scaled to be observable from a great distance.
- Tether: The observer's position changes with the simulation object's position.
- Tether Track: The observer's position changes with the simulation object's position while its heading is focused on a second simulation object.
- Tether Location Track: The observer's position changes with the simulation object's position while its heading is focused on a location.
- Track: The observer's heading changes to track the simulation object's position and heading.

The attach modes are described in detail in the following sections. For details about how to attach the observer to a simulation object, please see "Attaching the Observer to a Simulation Object or Prop," on page 52-2.

52.2.1. Absolute Mode

In Absolute, or free-fly mode, the observer is not attached to a simulation object. Movement of simulation objects does not affect the observer in any way.

52.2.2. Follow Mode

In Follow mode, the observer maintains its relationship to the simulation object as follows:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading and position change to maintain the offset.
- When the simulation object's pitch or roll change, the observer pitch and roll do not change.

Figure 52-4 illustrates the characteristics of Follow mode.

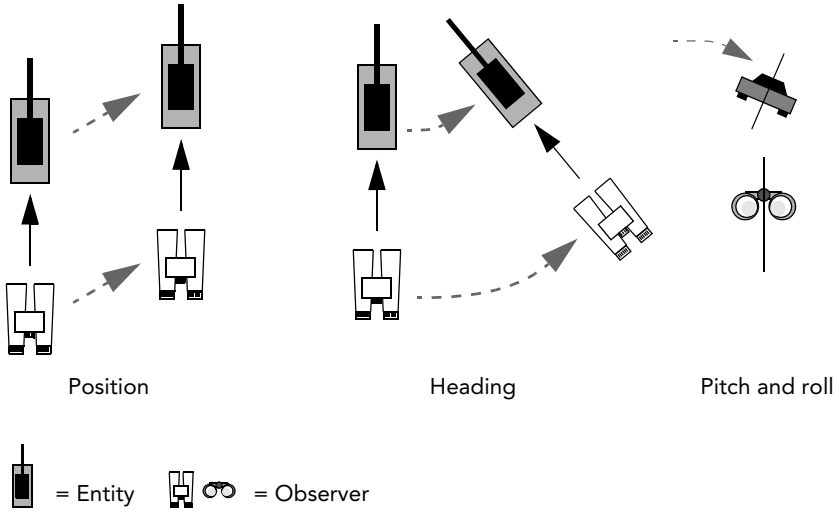


Figure 52-4. Follow mode

52.2.3. Mimic Mode

In Mimic mode, the observer behaves as though it is rigidly connected to the entity. Use Mimic mode to create the effect of riding in a vehicle cockpit, or of following a vehicle from an external location while connected to it by a solid, invisible rod. The observer maintains its relationship to the simulation object is as follows:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading and position change to maintain the offset.
- When the simulation object's pitch or roll change, the observer's pitch and roll change and the observer's position changes to maintain its offset.



When cockpit displays are enabled, you cannot change the position or orientation of the observer relative to the entity.

Figure 52-5 illustrates the characteristics of Mimic mode.

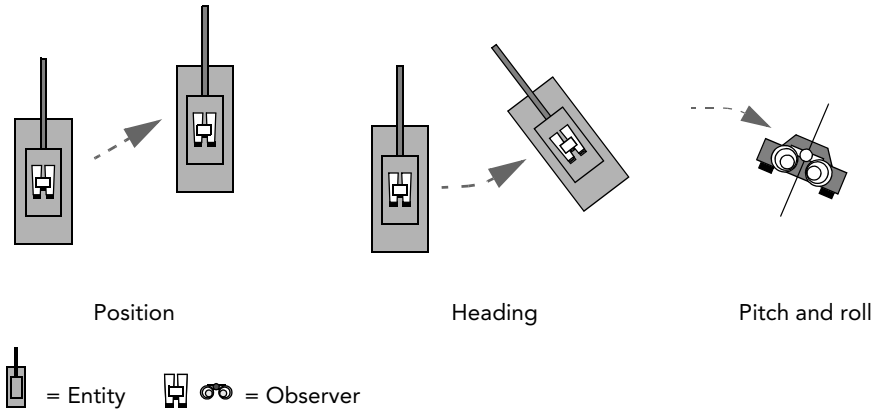


Figure 52-5. Mimic mode

52.2.4. Mimic Track Mode

Mimic Track mode combines Mimic mode and Track mode. The observer's position changes with that of the entity to which it is attached, as in Mimic mode, while its orientation changes to remain focused on a secondary, tracked entity.

Orientation controls are disabled.

Figure 52-6 illustrates the characteristics of Mimic Track mode.

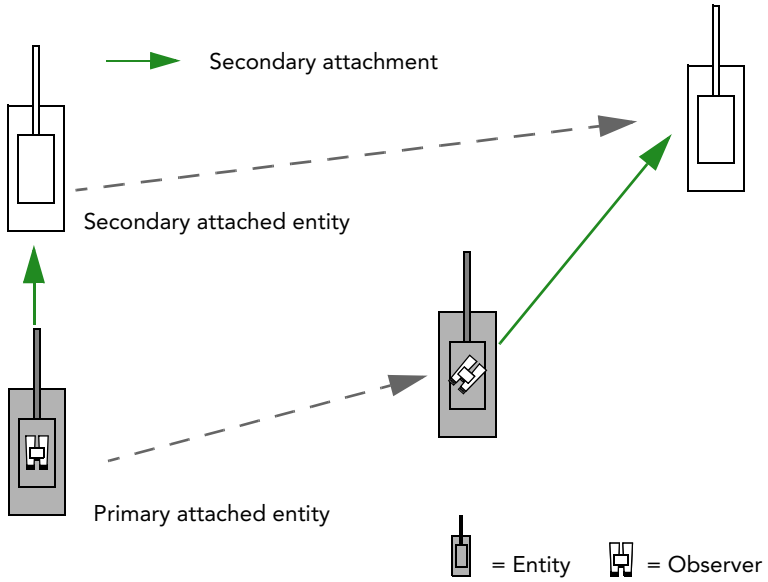


Figure 52-6. Mimic Track mode

52.2.5. Mimic Location Track Mode

Mimic Location Track mode works like Mimic Track mode except it tracks a location on the ground rather than an entity.

52.2.6. Space Follow Mode

Space Follow mode attaches the observer above an entity, looking down at it. Although it can be used with any entity, its primary purpose is for attaching to space objects such as satellites. You can also usefully use it with fixed-wing and rotary-wing entities.

The observer is attached at a fixed distance, determined by the near clipping plane for the object. For a satellite, this distance is approximately 500,000 meters. Since it is not possible to see regular models at that distance, the attached object is scaled to be visible. (You cannot change the scaling of the attached object.) Other objects are not scaled (except to the extent that model scaling might be enabled).

The observer maintains its relationship to the simulation object as follows:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading changes.
- When the simulation object's pitch or roll changes, the observer's pitch or roll does not change.

You cannot move the observer towards or away from the simulation object. You can rotate the observer around the entity, but you cannot go below the horizontal plane of the entity.

Figure 52-7 illustrates the characteristics of Space Follow mode.

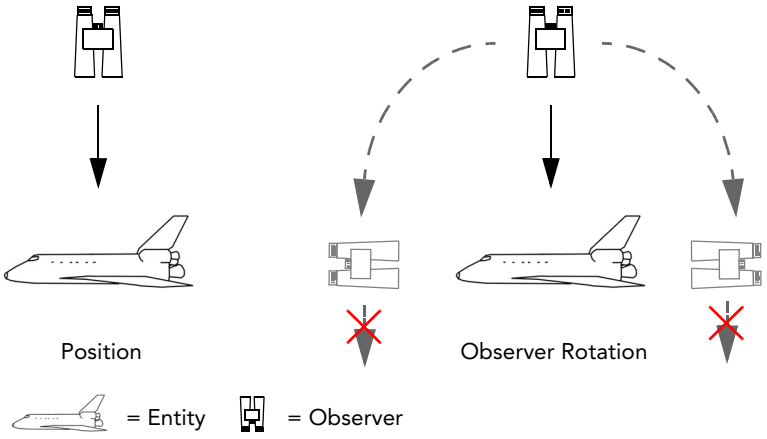


Figure 52-7. Space Follow mode

52.2.7. Tether Mode

In Tether mode, the observer's maintains a constant offset from the simulation object, but it is not affected by the simulation object's heading, pitch, or roll, as follows:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading does not change.
- When the simulation object's pitch or roll change, the observer's pitch and roll do not change.

Figure 52-8 illustrates the characteristics of Tether mode.

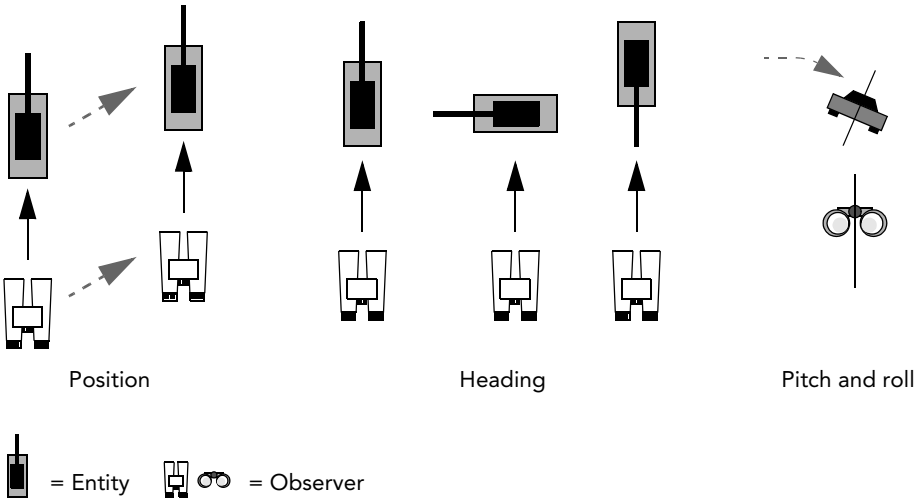


Figure 52-8. Tether mode

52.2.8. Tether Track Mode

Tether Track mode combines Tether mode and Track mode. The observer's position changes with that of the entity to which it is attached, as in Tether mode, while its orientation changes to remain focused on a secondary, tracked entity.

Orientation controls are disabled.

Figure 52-9 illustrates the characteristics of Tether Track mode.

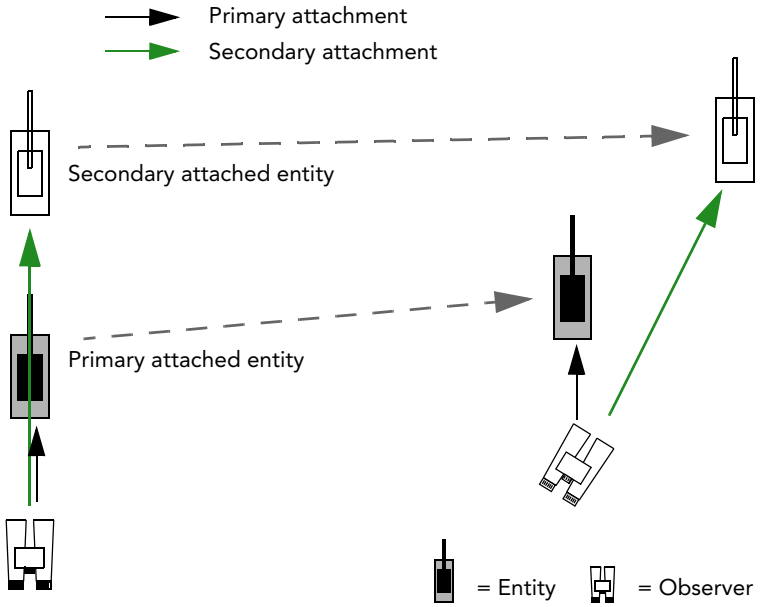


Figure 52-9. Tether Track mode

52.2.9. Tether Location Track Mode

Tether Location Track mode works like Tether Track mode, except that it tracks a location on the ground instead of an entity.

52.2.10. Track Mode

In Track mode, the observer's heading changes to track the simulation object's movement, keeping it centered in the field of view. The observer's position does not change. The observer maintains its relationship to the simulation object as follows:

- When the simulation object's position changes, the observer's heading changes to keep the entity centered in the observer's view; the observer's position does not change.
- When the simulation object's heading changes, the observer's heading does not change.
- When the simulation object's pitch or roll change, the observer's pitch and roll do not change.

Figure 52-10 illustrates the characteristics of Track mode.

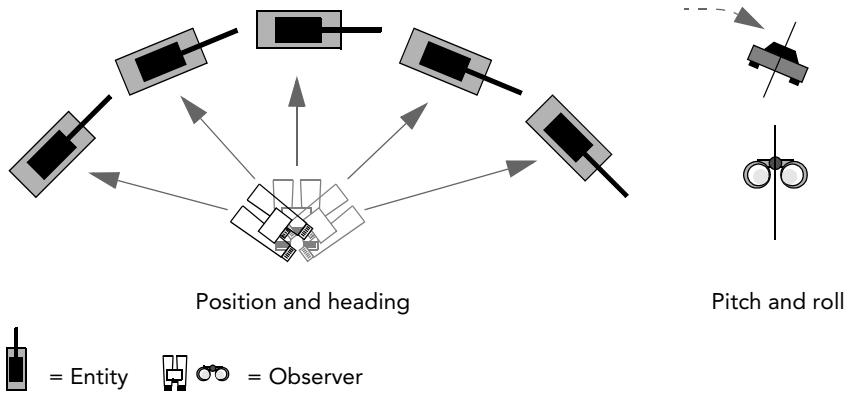


Figure 52-10. Track mode

52.3. Attach Modes in the 2D View

The 2D view supports Follow mode and Tether mode. In Follow mode, the heading of the observer changes with the entity (if Orient North is disabled); in Tether mode, it does not. You can use the standard movement keys (**w**, **a**, **s**, **d**) to offset the observer with respect to the simulation object's position. You can use the orbit keys (left arrow, right arrow) to offset the observer's orientation with respect to the entity's orientation (in Follow mode) or the terrain.

If the observer is attached to an entity in the 3D view and you switch to the 2D view, the attach mode is mapped to either Follow or Tether modes, as shown in Table 52-1. If you then switch back to the 3D view the original attach mode is restored.

Table 52-1: 3D to 2D attachment mode mapping

3D Attach Mode	2D Attach Mode
Follow	Follow
Mimic	Follow
Mimic Track	Follow
Mimic Location Track	Follow
Space Follow	Follow
Tether	Tether
Tether Track	Tether
Tether Location Track	Tether
Track	Tether

52.4. Attaching the Observer at Startup

To attach the observer to an entity when you load a scenario, save a view and specify it as a favorite, as described in "Specifying the Scenario Startup View," on page 51-33.



53. Inset Views, Sensor FOVs, HUDs, and Depth of Field

This chapter describes several kinds of special secondary windows plus head up displays.

Inset Views 53-2
Displaying Cockpit Displays 53-4
Displaying Instrument Panels 53-6
Configuring Entities to Use Instrument Panels 53-7
Displaying Sensor Views 53-8
Displaying a Sensor Field of View Cone 53-9
Controlling the Sensor View 53-10
Using the Depth of Field Effect 53-13
Configuring Depth of Field 53-15

53.1. Inset Views

You can display inset views of individual simulation objects. Figure 53-1 shows a helicopter hovering over the Ala Moana area in the *MAK Earth (online).mtf* terrain with an inset view of a car driving through the town.



Figure 53-1. Inset view

! By default, when you create an inset view, the observer in the inset view is attached to the simulation object in Mimic mode. This has the side effect of causing the 3D model in the main window to disappear. (The reverse occurs if you attach the observer in the main window to the simulation object in Mimic mode.) To make the model reappear, select the inset view and change the attach mode or detach the observer.

You can change the observer position and orientation in an inset view using keyboard controls. You can also edit window and channel properties for the window in the Display Engine Configuration Editor. (For information about the Display Engine Configuration Editor, please see Chapter 82, *Display Engine and Window Management*.)

When you create an inset view, the title bar contains the simulation object's name. However, the window is not actually tied to the simulation object and you can change the view in the inset view and view other simulation objects. The title bar will continue to display the name of the simulation object for which the window was created.

To open an inset view:

1. Select the object for which you want to create an inset view.
2. Choose **Objects** → **Create Window** → **Create Inset View**.

53.2. Displaying Cockpit Displays

VR-Forces can display cockpit displays created with GL Studio. It includes cockpits for fixed-wing, rotary-wing, and ground vehicle entities. As with other entity effects, cockpits have schemas and model definitions that you map to entity types. For details about cockpit display models, please see "Adding New Cockpit Display Models," on page 90-10.

VR-Forces supports 2D cockpits (Figure 53-2) and 3D cockpits (Figure 53-3). A 2D cockpit is fixed to the center of the screen and you cannot zoom in or out on the instrument panel. A 3D cockpit is aligned with the front of the vehicle. If you move the eyepoint, the instrument panel moves in or out of view as you would expect. You can zoom in and out on the instruments.



Figure 53-2. 2D cockpit



Figure 53-3. 3D cockpit

- Cockpits are displayed only when the observer is attached to an entity in Mimic mode.
- VR-Forces includes one 3D cockpit, for the F-16 fighter jet.
- Cockpits are not supported on Linux.

To display cockpits:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Cockpit Display Settings page (Figure 53-4).

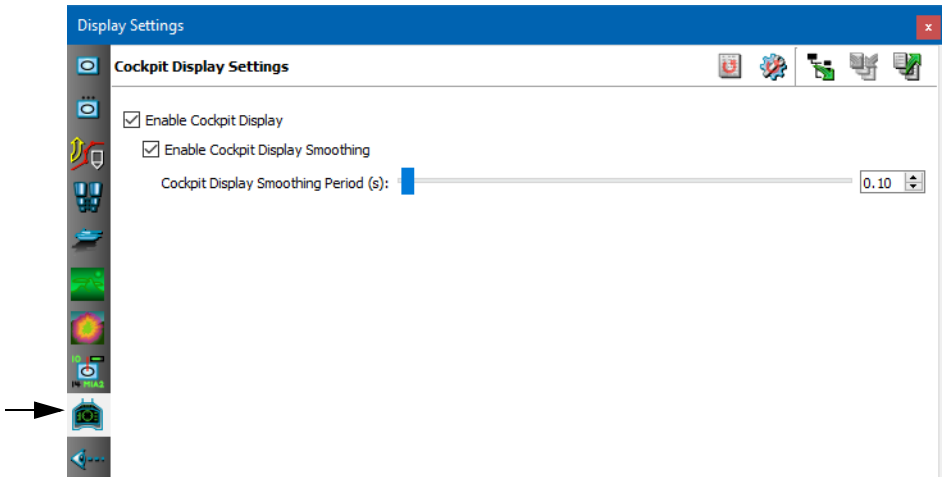


Figure 53-4. Cockpit display settings

3. Select or clear the Enable Cockpit Display check box.
4. Optionally, select the Enable Cockpit Display Smoothing check box. This smooths the display in the dynamic widgets, such as orientation or velocity, so they show a smoothed value rather than instantaneous changes.
5. Attach the observer to an entity in Mimic mode.

You can also enable and disable cockpit displays by choosing **Settings** → **Cockpit Displays** or by clicking the Cockpit Displays button ( ) on the Display Settings Toolbar.

53.3. Displaying Instrument Panels

Instrument panels are inset windows that display a vehicle instrument such as an altimeter or artificial horizon. Unlike a cockpit display, which is a fixed display that operates in Mimic mode, an instrument panel is an independent window (like an inset view) that you can place anywhere on the screen (Figure 53-5). It does not affect the observer mode. You can organize multiple instrument panels to create your own virtual cockpit.



Figure 53-5. Instrument panels

When you create an instrument panel, the title bar of the window lists the entity for which you created it.

- Instrument panels are GL Studio controls.
- Instrument panels are not supported on Linux.
- If the source of the instrument panel is removed from the exercise, for example when a Logger recording rewinds, the instrument panels are removed from the inset windows. To see them again you must open new instrument panels. Because the relationship between an instrument panel window and an entity is not persistent across multiple runs of an exercise, you cannot save an instrument panel layout for an entity. (You could save the layout of the windows to a display configuration, but they would not display instrument panels when you loaded the configuration.)
- If you want to put multiple instrument panels in one window, like the six-pack panel provided with VR-Forces, you must use GL-Studio.

To open an instrument panel:

1. Select the entity for which you want to display an instrument panel.
2. Choose **Objects** → **Create Window** → **Create Instrument Panel** → *panel*, where *panel* is one of the instrument panels listed.

53.3.1. Configuring Entities to Use Instrument Panels

VR-Forces includes a set of instrument panels for fixed-wing aircraft, although many of them may be appropriate for rotary-wing aircraft, and even for ground vehicles. (For example, you might want to show a compass for a ground vehicle to get its heading or an altimeter to show its height above sea level.) The availability of instrument panels to an entity is determined by the instrumentPanelCategory parameter in its model definition.

The instrument panels provided with VR-Forces have the instrumentPanelCategory set to FixedWing. All fixed-wing entities have this parameter set in their model definition. None of the other entities do. If you want other entities to display instrument panels provided with VR-Forces, you can add this parameter to their model definitions. (You could also copy an instrument panel model definition and change the category to some other value if you want to. Regardless, to use an instrument panel, an entity's model definition must include the instrumentPanelCategory parameter and set it to a valid value.)

If you create new instrument panels, you must add instrument panel model definitions for them and specify a value for the instrumentPanelCategory parameter. Then you can add the parameter to the appropriate entity model definitions.

53.4. Displaying Sensor Views

VR-Forces can display the view from gimbaled visual sensors, such as a camera on a UAV (3D only). The view is displayed in an inset window that has information about the observer mode and area being viewed (Figure 53-6). The window has its own observer and you can change the observer mode in the view.

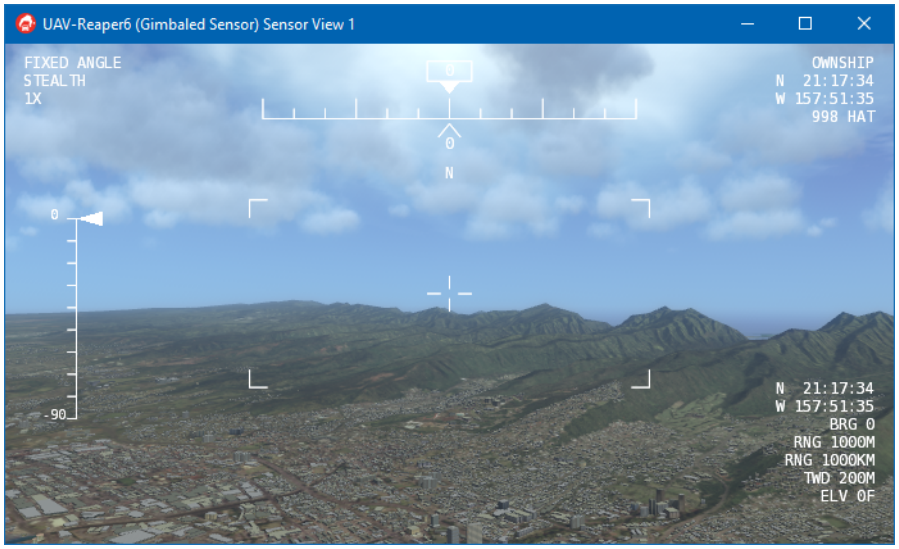


Figure 53-6. Sensor View

Each sensor view has a control panel (Figure 53-7) that you can use to move the sensor, change its aim, and zoom in and out. You can also lase whatever the sensor is looking at.

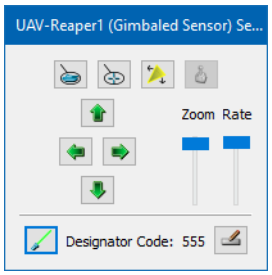


Figure 53-7. Sensor Control Panel

The UAVSensorDemo example scenario demonstrates use of an entity with a gimbaled sensor. The scenario also has a Lua script that shows how to successively aim the sensor at many entities.

To display a sensor view:

1. Select an entity that has a gimbaled sensor.
2. Choose **Objects** → **Create Window** → **Create Sensor View** → *sensor*

53.4.1. Displaying a Sensor Field of View Cone

You can display translucent cones that show the area on the terrain that a gimbaled sensor is observing. Each cone is a different color. Figure 53-8 shows two sensor field of view cones. One is focused on an entity and is zoomed in for a tight focus. The other is focused on a point on the terrain.



Figure 53-8. Sensor field of view cones

To display sensor field of view cones:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 50-2).
3. In the list of settings, select the Sensor Fields of View check box.



You can also enable and disable sensor field of view cones by choosing **Observer** → **Sensor Fields of View** or by clicking the Sensor Fields of View button () on the Observer Settings Toolbar.

53.4.2. Controlling the Sensor View

When a sensor view opens, a control panel opens with it (Figure 53-7). This control panel works only with its associated sensor view. If you open multiple sensor views, there will be a control panel for each one. The Sensor Control Panel lets you do the following:

- Aim the sensor at a simulation object or a point on the terrain. (You can also aim a sensor with the Sensor Aim set data request. For details, please see "Sensor Aim," on page 34-38.)
- Set the sensor to scan mode. (You can also set a sensor to scan with the Sensor Aim set data request.)
- Take control of the sensor with a joystick. The joystick must be configured for use with the sensor view. For details about configuring joysticks, please see "Configuring Joysticks and Keyboard Control," on page 80-4.
- Zoom the sensor. (You can also zoom a sensor with the Sensor Zoom set data request. For details, please see "Sensor Zoom," on page 34-40.)
- Move the sensor field of view. You can change the sensitivity of the movement buttons.
- Lase the center point of the sensor view.
- Set the laser's designator code.



If you zoom the sensor view to the maximum magnification and the terrain turns white, you can fix the problem by setting the clipping plane to Clamp Near Clip When Magnified. For details, please see "Setting the Clipping Planes," on page 82-9.

To aim the sensor at a simulation object:

1. Click the Aim at Simulation Object button (). The Entity to Track dialog box opens.
2. Select the simulation object you want to aim at.
3. Click OK.

To aim the sensor view at a location:

1. Click the Aim at Location button (). The Location to Track dialog box opens.
2. Click a location on the terrain or enter the coordinates in the dialog box.
3. Click OK.

- To set a sensor to scan mode, click the Scan button ( ).
- To control the sensor field of view with a joystick, click the Joystick button ( ).
- To zoom the sensor view, adjust the Zoom slider.
- To move the sensor view, click any of the arrow buttons.
- To change the rate at which the arrow buttons move the sensor view, adjust the Rate slider.

If you enable lasing, the entity the sensor view is attached to lases the center point of the sensor view (Figure 53-9).



Figure 53-9. Lasing a point in a sensor view

- To lase the center point of the sensor view, in the Sensor View control panel click the Lase Object button ( ) on the control panel.

VR-Forces calculates a unique laser code for each sensor view so that there are no duplicate codes. You can change the laser designator code.

To specify the laser designator code when lasing from a sensor view:

1. In the Sensor View control panel, click the Laser Code button ( ). The Laser Code dialog box opens.
2. Type a 3 or 4 digit code between 111 and 1788, using only digits 1 through 8.
3. Click OK.

Configuring a Gimbaled Sensor's Azimuth

The characteristics of a gimbaled sensor, such as where it is located on an entity and the range through which the sensor can scan (the azimuth), are configured in the Gimbaled Visual Sensor system. You can edit many properties in the Simulation Object Editor. In particular, you may want to edit the azimuth. If you want the sensor to swivel 360 degrees, set the Minimum Azimuth Angle and Maximum Azimuth Angle to the same value.

To edit the properties of a gimbaled sensor:

1. Open the Simulation Object Editor. (For details about using the Simulation Object Editor, please see Chapter 67, *The Simulation Object Editor and SMSs*.)
2. Select the entity that has a gimbaled sensor.
3. Select the Systems tab.
4. In the Sensors window, select the Gimbaled Visual Sensor.
5. Click the Properties button ().
6. Edit properties as desired.
7. Click OK.
8. Save the SMS.

53.5. Using the Depth of Field Effect

Depth of field controls the area in the scene that is in focus. Depth of field is calculated based on a focal depth and a focal range. The focal depth is the distance from the observer that is the center (focal point) of the area in focus. The focal range is the distance from the focal point that is in focus.

In Figure 53-10, the focal depth is about 10 meters, fairly close to the observer, and the focal range is also small, so that the scene is out of focus just past the car (note that the first large tree is out of focus). In Figure 53-11, the focal depth is the same, but the range is much greater, so the scene is in focus to the back of the parking lot.

In Figure 53-12 the focal depth is centered at the second large tree. The focal range is, again, fairly narrow, so that the person and car in the foreground are out of focus.

VR-Forces lets you set the focal depth automatically or manually. In either case, you can adjust the focal range.



Figure 53-10. Depth of field (close)



Figure 53-11. Depth of field (far)



Figure 53-12. Depth of field, middle distance

To enable depth of field:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
3. Select the observer mode for which you want to enable depth of field.
4. Select or clear the Depth of Field check box.



You can also enable or disable depth of field by choosing **Observer** → **Depth of Field**.

53.5.1. Configuring Depth of Field

You can enable and disable automatic depth of field and you can adjust the focal depth and focal range.

To configure depth of field:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 53-13).

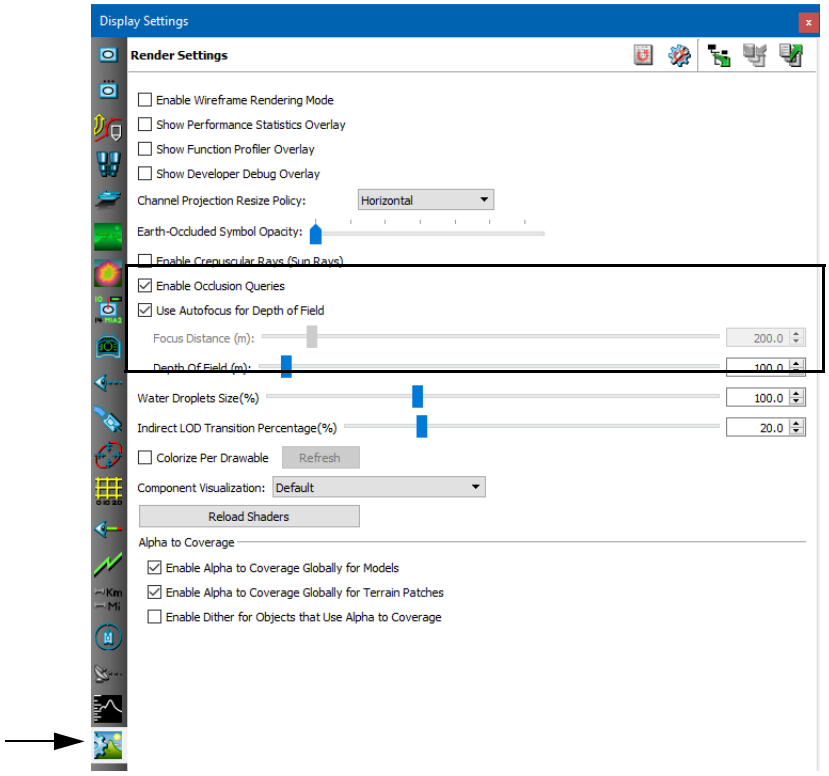


Figure 53-13. Depth of Field settings

3. To use autofocus for focal depth, select the Use Autofocus for Depth of Field check box.
4. To set the focal depth manually, clear the Use Autofocus for Depth of Field check box and adjust the Focus Distance slider or specify a value in the box.
5. To adjust the focal range, adjust the Depth of Field slider or specify a value in the box.



XI. Terrain

Each simulation takes place within the extents of a terrain database. A terrain database provides three-dimensional information about a geographic area and provides the environment within which simulation objects function. With the information in a terrain database, VR-Forces calculates height, intervisibility, and other three-dimensional information.

VR-Forces allows you to load terrain data in the following basic ways:

- Compose a terrain from elevation, imagery, and feature data.
- Open an MTF file.
- Stream terrain from local files or a terrain server such as VR-TheWorld Online.
- Open a terrain database in a supported format, such as FLT, DTED, or GDB.

The recommended work flow is:

1. Create an earth file.
2. Connect to the earth file.
3. Save the terrain as an MTF file.
4. Create a scenario and choose the MTF file as the scenario database.

The decision making process for choosing a terrain when you create a scenario is described in Chapter 7, *Creating and Running Scenarios*.



54. Introduction to Terrain Usage in VR-Forces

This chapter provides introductory information about terrain usage, including supported file formats, coordinate systems supported, and use of large terrain databases.

Composing Terrains (and Creating MTF Files) 54-2
Supported File Formats 54-2
- Terrain Data 54-3
- Elevation Data 54-3
- Supported Image Formats 54-4
- Supported Feature Data Formats in the Front-end 54-4

Feature Layers and Props 54-4
- Geometric Representation of Point Feature Data 54-5
- Feature Data Types Supported by the Back-end 54-5

Coordinate Systems 54-6
- The UTM Coordinate System 54-6
- Geocentric Terrain Databases 54-6

Loading Supported Databases 54-7
Using Large Terrain Databases 54-7
- Terrain Paging 54-8
- Terrain Streaming 54-9

Using Ground Clamping 54-9
- Configuring Ground Clamping 54-11

54.1. Composing Terrains (and Creating MTF Files)

VR-Forces lets you compose terrains from a variety of supported data formats. You can combine elevation data (terrain patches), imagery, and feature data to create terrains.

VR-Forces offers the following basic methods for creating terrains:

- Compose a terrain by loading individual elevation, image, and feature data files. This method is best for relatively small terrains that do not require paging. It supports all of the coordinate systems that VR-Forces supports. For more information, please see "Creating a Composed Terrain," on page 57-2.
- Specify a set of files to load using an osgEarth format earth file. Use this method to stream terrain data from local or remote sources. This method is best for terrains with large amounts of elevation or image data, which would benefit from paging. For more information, please see "Connecting to Terrain Servers," on page 58-2 and Chapter 59, *Loading Data Using Earth Files*.

When you compose a terrain, you can save it as a MAK terrain file (MTF), which is essentially a description of all of the data used to visualize the simulated 2D and 3D environment. Saving a terrain lets you automatically load all of the data that you loaded manually when you first created it.

You can save terrains in a cached format that loads terrain data more quickly than if the terrain is created from the source data. For details, please see "Configuring File Caching," on page 64-2.

54.2. Supported File Formats

VR-Forces can import elevation, terrain, images, models, and feature data. Some types of files can contain more than one type of information. For example GeoTiff files can have elevation data or just have image data. The following sections list the supported formats.



In theory, VR-Forces can load any format supported by osgEarth if you use the earth file format to load them. However, VR-Forces does not include drivers for all of the formats supported by osgEarth. Furthermore, when using osgEarth, VR-Forces only supports geocentric terrains. If you are uncertain about support for a particular format, please contact support@mak.com.

54.2.1. Terrain Data

VR-Forces imports terrain data directly as terrain patches. It supports the following formats:

- MAK terrain database (*.gdb*).
- MAK encrypted data format (*.medf*).
- OpenFlight (*.flt*).
- MetaFlight (*.mft*).
- Shape terrain data (*.shp*).
- TerraPage (*.txp*).
- CDB (using an earth file).

VR-Forces provides limited support for data in the following formats. If you have problems loading data in these formats, MAK will try to assist you, but cannot guarantee that it will be possible to use your data.

- 3DS
- OSG
- IVE
- LWO

54.2.2. Elevation Data

VR-Forces supports the following types of elevation data:

- DTED (*.dt0*, *.dt1*, *.dt2*, *.dted*).
- GeoTiff.
- DEM.
- Arc/Info Binary Data (*.adf*).
- Other formats supported by osgEarth.

54.2.3. Supported Image Formats

You can import images in the following formats:

- GeoTiff (*.tif*, *.tiff*).
- JPEG (*.jpg*, *.jpeg*).
- PNG.
- RGB.
- Windows bitmap (*.bmp*).
- GIF.
- ECW (Windows only).
- Other formats supported by osgEarth using the GDAL driver. For a list of formats supported by GDAL, go to http://www.gdal.org/ and select the Supported Formats link. If you have problems loading an image format that is not in this list, please contact support@mak.com. However, we cannot guarantee that unlisted formats are fully supported. For more information about earth files, please see "Connecting to Terrain Servers," on page 58-2 and Chapter 59, *Loading Data Using Earth Files*.

54.2.4. Supported Feature Data Formats in the Front-end

VR-Forces supports the following feature data formats in the front-end:

- Shape files.
- S-57.
- OGR vector formats referenced in earth files.

54.3. Feature Layers and Props

A terrain can have objects such as buildings and trees embedded in it. These objects cannot be manipulated in this format. However, VR-Forces lets you extract these types of objects from the terrain and render them independently as props. Props are terrain elements that you can manipulate as separate objects.

You can add props to a terrain by extracting them from an external reference in a terrain patch or from a feature layer. You can also add props directly from the Props Palette.

A feature layer is a file that contains feature data such as point features, linear features, and areal features. You can extract point features as props. You can extract props from MAK Terrain Format (GDB) files, shapefiles (SHP), or DFAD data (DFD).

For more information, please see "Adding Props to a Terrain," on page 57-20.

54.3.1. Geometric Representation of Point Feature Data

There are differences between how the front-end and back-end process feature data. This is because the information needed to simulate simulation object interaction with features is different from the information needed to render the scene.

The front-end renders point features such as buildings or trees based on the data provided in the feature layer. These objects could have been extracted as props, in which case they are represented as geometric objects in the terrain.

If you are streaming features using an earth file, you can configure the earth file to extrude buildings into 3D geometry. This capability is demonstrated in some of the insets in *MAK Earth (online).mtf*, for example the Boston and Honolulu insets.

The back-end representation of point features depends on how they are used. Collision avoidance treats them as bounding volumes with infinite height. Line-of-sight calculations treat them as bounding volumes with a defined height. If you extract features as props, the back-end treats them as geometry where that is appropriate and as features otherwise.

The back-end does not recognize extruded geometry.

54.3.2. Feature Data Types Supported by the Back-end

The back-end supports the following types of feature data files:

- MAK terrain database (*.gdb*).
- CTDB 4, 7, 7L (*.c4b*, *.c7b*, *.c7l*).
- Shape terrain data (*.shp*).
- S-57.
- WFS.
- OGR vector formats referenced in earth files.
- VMAP.

The preferred format for adding features is shapefiles. You can extract feature data from GDB, DFD, VMAP, and CTDB into shapefiles with the vectorNetworkToShapeFiles utility. For details, please see Section 65.7, "Extracting Feature Data from GDB to Shapefiles". You can extract feature data from S-57 files into shapefiles with the ogr2ogr tool. For details, please see Section 65.8, "Converting S-57 Feature Data to Shapefiles".

54.4. Coordinate Systems

VR-Forces supports UTM, geocentric, flat earth, and Lambert Conformal Conic coordinate systems. There is no presumption that a particular database format, such as OpenFlight or DTED, will use a particular coordinate system.

54.4.1. The UTM Coordinate System

VR-Forces includes several databases that use Universal Transverse Mercator (UTM) coordinates. UTM databases are well-suited to two-dimensional views and simulations on a relatively small geographic area. However, their accuracy decreases as you move away from the origin, so they are not well-suited for exercises that need to simulate over areas greater than one UTM zone.

There are several types of UTM systems. In general, the earth is divided into 60 zones generally 6 degrees wide in longitude. These zones may be further subdivided into quadrangles by using letter designations from south to north. This complicates references to UTM. For example, standard offset could be used with or without the quadrangles (which affects the northing portion of the coordinate), non-offset does not use the quadrangles, and Milgrid requires the quadrangles.

More specifically, UTM coordinates define two dimensional, horizontal, positions. UTM zone numbers designate 6 degree longitudinal strips extending from 80 degrees South latitude to 84 degrees North latitude. UTM zone characters designate eight degree zones extending north and south from the equator. There are special UTM zones between 0 degrees and 36 degrees longitude above 72 degrees latitude and a special zone 32 between 56 degrees and 64 degrees north latitude.

VR-Forces uses the following definition:

A non-Cartesian coordinate system in which the X, Y, and Z correspond to easting (nearly east), northing (nearly north), and height above an ellipsoid that approximates the surface of the earth. When in UTM mode (the default), VR-Forces uses an offset UTM coordinate system, one in which coordinates are expressed with respect to an origin located at the latitude and longitude specified in a configuration file. Typically, this will be the location of the origin of your terrain database.

54.4.2. Geocentric Terrain Databases

To support large play areas, simulations can use geocentric coordinates. In the geocentric coordinate system, the origin is at the center of the earth. The positive X-axis passes through the prime meridian at the equator; the positive Y-axis passes through 90 degrees east longitude at the equator; and the positive Z-axis passes though the North Pole.

54.5. Loading Supported Databases

MTF files reference elevation, image, and feature files that VR-Forces processes and loads when you load a scenario. However, you can also load terrain databases directly into VR-Forces. They will be converted to the VR-Forces internal format using default values. This process is less efficient than loading MTF terrains, may take more time, and must be repeated each time you load the terrain as part of a scenario or in the front-end.

When you create a scenario you can choose a terrain in any supported format. If you want to open a terrain using the **File** → **Open Terrain** menu option, you can only choose MTF files. To load a non-MTF terrain out of the context of creating a scenario, you must load it as a terrain patch.

If you load a terrain directly, you can still save it to MTF.

54.6. Using Large Terrain Databases

VR-Forces can support large databases, where large might mean a high polygon count, a large geographic area, or both. VR-Forces supports two methods for dealing with large databases – terrain paging and streaming terrain. Both approaches have a common goal – providing access to large amounts of data without loading all of that data into memory at any one time.

VR-Forces cannot arbitrarily page or stream terrains. It can page terrains that have been designed for paging and can stream terrains that are accessed through a terrain server, such as VR-TheWorld Server. For details about support for paged terrains, please see "Loading MetaFlight Terrains," on page 58-5. For details about connecting to terrain servers, please see "Connecting to Terrain Servers," on page 58-2 and "Creating an Earth File to Load Terrain Data," on page 66-2.



Even if you use terrain paging or streaming terrain, it is still possible that a large terrain could exceed the memory limitations of your operating system.

54.6.1. Terrain Paging

Terrain paging breaks up a database into multiple cells that can be loaded independently. By comparison, a non-pageable terrain must be loaded in its entirety. When VR-Forces loads a paged terrain, the front-end loads the terrain at whatever level of detail is required by the zoom level. The simulation engine always loads terrain at the highest level of detail. Therefore, if you rapidly zoom into a section of the terrain in the front-end, there may be a delay in correlation between simulation objects and the terrain until the front-end catches up to the proper level of detail.

In general, the simulation engine does not load terrain until it needs the terrain to support the objects in the scenario. (Some terrain data is always loaded immediately, such as terrain data in OpenFlight format.) For example, if you create a new scenario and specify a paged terrain, the simulation engine does not load terrain until you create a simulation object. Then it only loads the terrain page required for the object.

When you run a scenario, as simulation objects move, the simulation engine tries to predict which terrain pages it will need to support the simulation objects and tries to page them in before the simulation objects need them. If the total polygon count required to page in required terrain exceeds the maximum polygon count configured, VR-Forces pages out (removes) unused pages. The simulation engine pages out the least recently used data.



Predictive terrain paging is done in real time, not just when a scenario is running. For example, when you load a scenario, all required terrain is paged in and when you add objects during scenario creation, terrain is paged in.

You can also manually page in terrain using terrain page-in areas. For details, please see "Manually Paging-In Terrain," on page 58-6.

Paging is done based on the following priorities:

- If a simulation object is moving, paging in terrain that it needs is a higher priority than predictive paging.
- Blocking terrain calls have a higher priority than non-blocking calls. (A blocking call is one that prevents the simulation from progressing until the call is completed. For more information, please see the `assertOnBlockingTerrainCalls` parameter in Table B-1, in "The vrfSim.mtl Configuration File," on page B-2.)
- Paging out terrain to recover memory is a higher priority than paging in terrain. The need to page out is calculated based on the polygon count of the terrain, not on its memory usage.

54.6.2. Terrain Streaming

The term terrain paging is used in reference to static, local terrains. Another way to load terrain data for a large area is to stream it from a terrain server. The idea is similar to terrain paging. Only the data you need at a particular location is loaded. The difference is that a terrain server can serve many computers at a time and normally holds much more data than a static local terrain. It can stream elevation data, imagery, map data, and feature data. It may have data at multiple levels of resolution for a particular portion of the terrain. VR-Forces supports connection configurations to terrain servers, including MAK's VR-TheWorld Server.

54.7. Using Ground Clamping

Because of differences in terrain databases among exercise participants, entities can sometimes appear to be underground or hovering above the terrain surface. To compensate, VR-Forces can clamp land and sea entities to the terrain surface. When ground clamping is enabled, VR-Forces keeps an entity anchored to the terrain surface regardless of the altitude data contained in its state update.



- Use of ground clamping can reduce your frame rate. If you correlate the simulation and visualization terrain databases, you can eliminate the need to use ground clamping.
- Although ground clamping controls are available in aggregate-level scenarios, since aggregate-level scenarios do not use 3D models, they are not useful.

To enable or disable ground clamping:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 54-1).

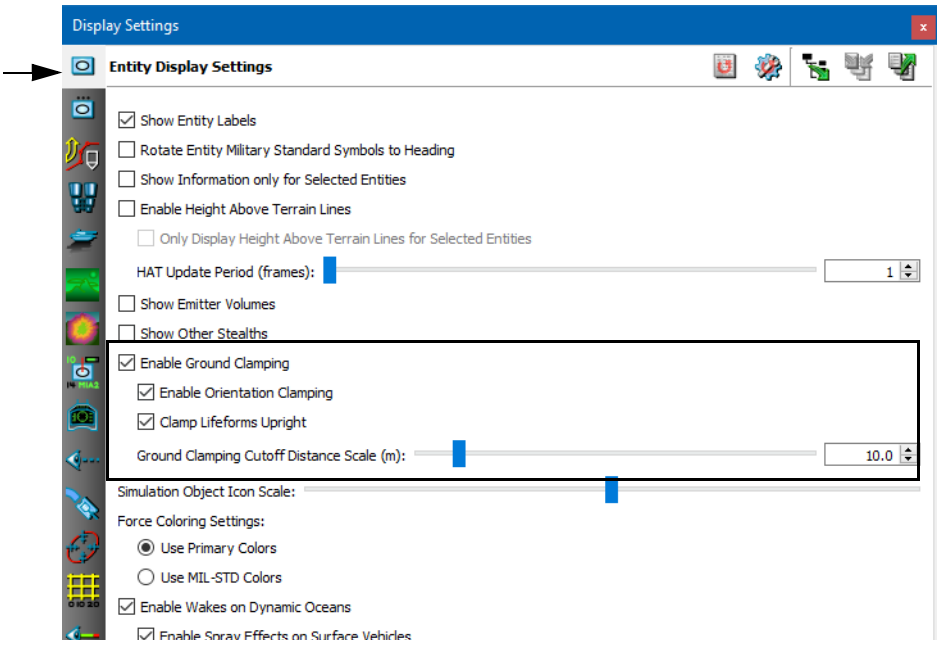


Figure 54-1. Ground clamping settings

3. Select or clear the Enable Ground Clamping check box.



You can also enable and disable ground clamping by choosing **Settings** → **Ground Clamping**, or clicking the Ground Clamping button (▾) on the Display Settings Toolbar.

54.7.1. Configuring Ground Clamping

You can configure the following aspects of ground clamping:

- Enable or disable ground clamping.
- Enable or disable orientation clamping.
- Force lifeforms to remain upright relative to the plane of their location when moving up slopes or steps, rather than being perpendicular to the terrain.
- The distance from the terrain within which entities are clamped.

To configure ground clamping:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 54-1).
3. To enable or disable a ground clamping option, select or clear its check box.
4. To specify the distance from the terrain in which entities are ground clamped, adjust the Ground Clamping Cutoff Distance Scale slider or type a value, in meters, in the box. You can also access the Ground Clamping Cutoff Distance Scale slider by clicking the down arrow next to the Ground Clamping button ( ) on the Display Settings Toolbar.



The setting for the Ground Clamping Cutoff Distance Scale maximizes at 100 meters. If you set it to the maximum, the distance is infinite and all entities are ground-clamped.



55. Displaying Terrain Information

This chapter describes how to change the display of terrain to provide different types of useful terrain information.

Introduction 55-2
Coloring Terrain Based on Elevation 55-3
Configuring Elevation Color Settings 55-4
Displaying Grid Lines 55-6
Configuring Grid Lines 55-7
Displaying Models and Terrain in Wireframe Mode 55-8
Filtering Earth File Layers 55-10
Displaying Feature Data (2D) 55-11
Displaying a List of Vector Features 55-13
Changing the Colors Used for Feature Data 55-14
Displaying Vector Features 55-15
Displaying Coordinates and Soil Type for a Location 55-16
Viewing the Map Scale 55-17
The Map Scale Overlay 55-17
Using Terrain Draw Areas to View Terrain Polygons 55-18
Configuring Visible Surfaces (GDB Soil Types) 55-19
Terrain Databases Provided with VR-Forces 55-21

55.1. Introduction

As you create and view simulations, you may need information about the terrain. By changing how terrain is displayed, you can get some of this information. If your terrain does not have texture data or an image overlay, features such as elevation coloring can provide a more realistic look to the terrain.

For information about how to compose terrains, please see Chapter 57, *Composing Terrains*.

55.2. Coloring Terrain Based on Elevation

You can color the terrain based on elevation.

To enable or disable coloring of terrain by elevation:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Elevation Color page (Figure 55-1).

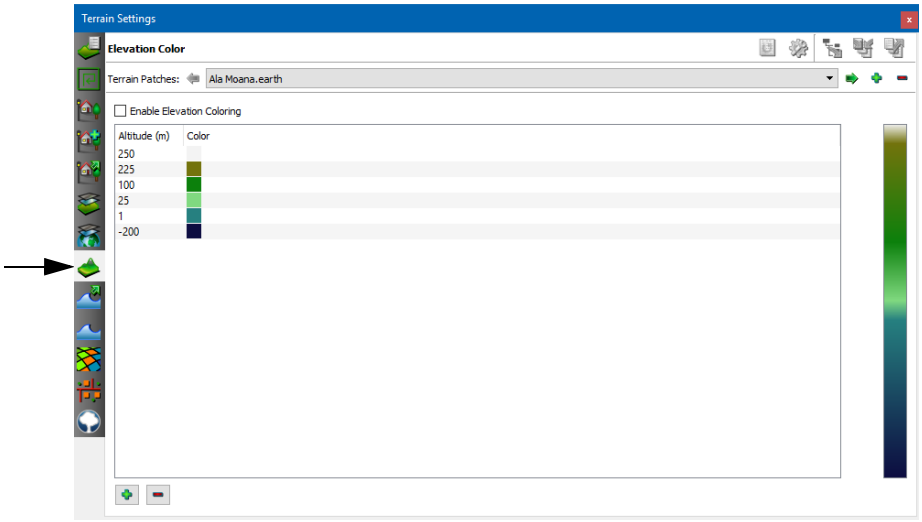


Figure 55-1. Elevation Color page

3. In the Terrain Patches list, select the terrain patch for which you want to enable elevation coloring.
4. Select or clear the Enable Elevation Coloring check box.

55.2.1. Configuring Elevation Color Settings

You can specify the color applied to terrain at a given altitude. You can add altitude points and remove them.

Inserting an Altitude Point

To insert an altitude point:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Elevation Color page (Figure 55-1).
3. In the Terrain Patches list, select the terrain patch for which you want to insert an altitude point.
4. Select an altitude point in the list that is above or below the altitude point you want to add.
5. Click Insert Point. A new point is added with the same altitude as the one you selected.
6. Double-click the new altitude point to make it editable.
7. Set the altitude point to the altitude that you want.
8. Click away from the altitude point to set it.

Deleting an Altitude Point

To delete an altitude point:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Elevation Color page (Figure 55-1).
3. In the Terrain Patches list, select the terrain patch for which you want to delete an altitude point.
4. Select the altitude point that you want to delete.
5. Click Delete Point.

Changing the Color of an Altitude Point

To change the color of an altitude point:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Elevation Color page (Figure 55-1).
3. In the Terrain Patches list, select the terrain patch for which you want to change the color of an altitude point.
4. Double-click the color swatch for the altitude point whose color you want to change. A color chooser dialog box opens.
5. Select or specify a new color.
6. Click OK.

55.3. Displaying Grid Lines

VR-Forces can display grid lines in the 2D view. Grid-lines are an observer-specific setting.

To enable or disable grid lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 55-2).

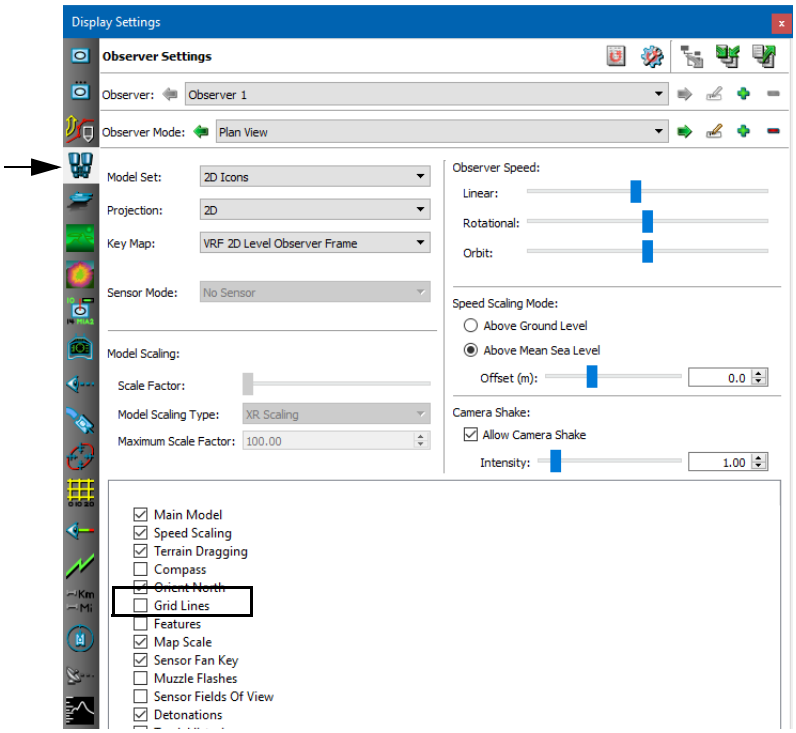


Figure 55-2. Observer Settings page

3. Select Plan View mode (or a locally created 2D observer mode).
4. Select or clear the Grid Lines check box.



You can also enable or disable grid lines by choosing **Observer** → **Grid Lines**.

55.3.1. Configuring Grid Lines

You can configure the coordinate system used for grid lines, the precision, and the spacing. Spacing can be fixed, in meters, or determined by VR-Forces using a specified number of screen pixels. Fixed spacing can result in a very tight grid and unreadable labels as you zoom out. Auto spacing does not have this problem. However, the grid lines will constantly recalculate as you zoom in and out.

To configure grid lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Grid Lines Settings page (Figure 55-3).

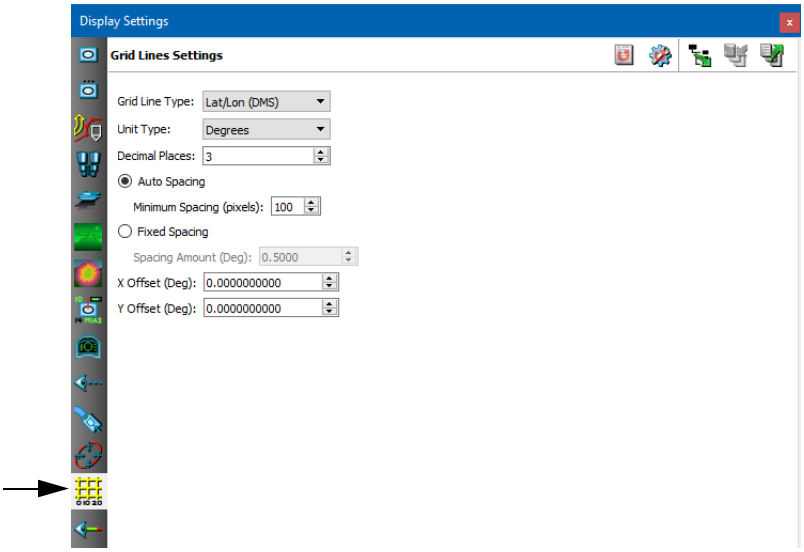


Figure 55-3. Grid Lines Settings page

3. To change the coordinate system used for grid lines, select an option in the Grid Line Type list. The list displays grid line types that are appropriate for the type of database loaded. (If no database is loaded, the coordinate system is set to Flat Earth.)
4. To specify the unit type for the grid line labels, select an option on the Unit Type list.
5. To specify the precision of the grid line labels, type a value in the Decimal Places box.
6. To change the spacing type, select AutoSpacing or Fixed Spacing and specify a value for the chosen spacing type.
7. To specify an offset from the origin, change the values for X Offset and Y Offset.

55.4. Displaying Models and Terrain in Wireframe Mode

In wireframe mode, VR-Forces displays the polygons that make up the terrain and objects (Figure 55-4).



Figure 55-4. Wireframe off and on

To enable or disable wireframe mode:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 55-5).

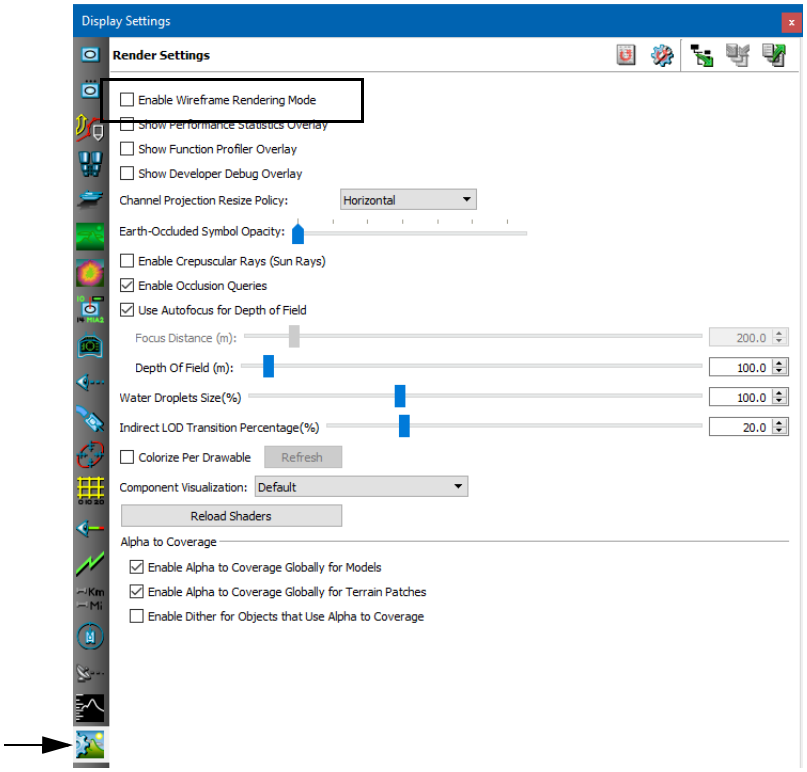


Figure 55-5. Render Settings page

3. Select the Enable Wireframe Rendering Mode check box.



You can also enable and disable wireframe mode by choosing **Settings** → **Wireframe Rendering**.

55.5. Filtering Earth File Layers

When you load a terrain using an earth file, imagery and feature data is specified in image elements and model elements. You can load multiple instances of these types of data, resulting in a rich, complex terrain. However, if you are only interested in certain types of terrain features, such as roads or rivers, displaying all of the terrain data can obscure the data that you are most interested in. To help declutter the map, you can filter out image and model layers globally or by observer mode.

This feature is particularly useful if you want to show some layers in the PVD observer mode and others in one of the 3D observer modes. Settings are saved to the MTF file so that they are in effect in all scenarios that use the terrain.

- Filtering an earth file layer affects all open windows. You cannot hide a layer in one window and leave it open in a different one.
- Filtering only affects the front-end. All layers are loaded in the back-end.

To filter out earth file layers:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Earth Layers page (Figure 55-6).

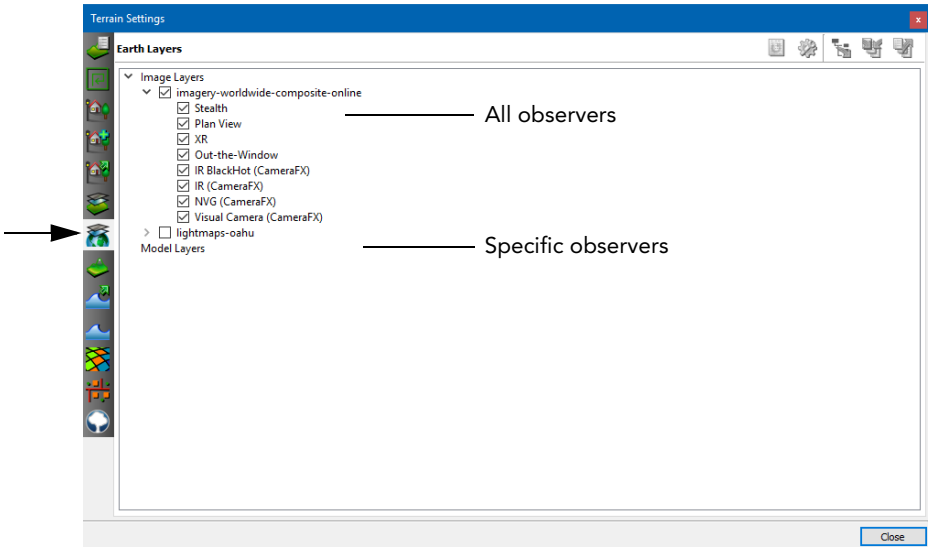


Figure 55-6. Earth Layers page

3. To display or hide an image or model layer for all observer modes, select or clear its check box.

4. To display or hide an image or model layer for a particular observer mode, expand the entry for the layer you want to change and select or clear the check box for the observer mode you want to change.

55.6. Displaying Feature Data (2D)

If a terrain contains feature data, you can display it in the 2D view. Figure 55-7 shows a section of the Little Pond terrain with features disabled and enabled.



Figure 55-7. Feature data

This setting only displays features imported from GDB files or shapefiles. It does not display features streamed from a terrain server. If you load features using an earth file, they are drawn (or not drawn) based on the configuration information in the file.



If your terrain has many features, displaying them can take a long time.

To display feature data:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page.
3. In the Observer Mode list, select Plan View (Figure 55-8).

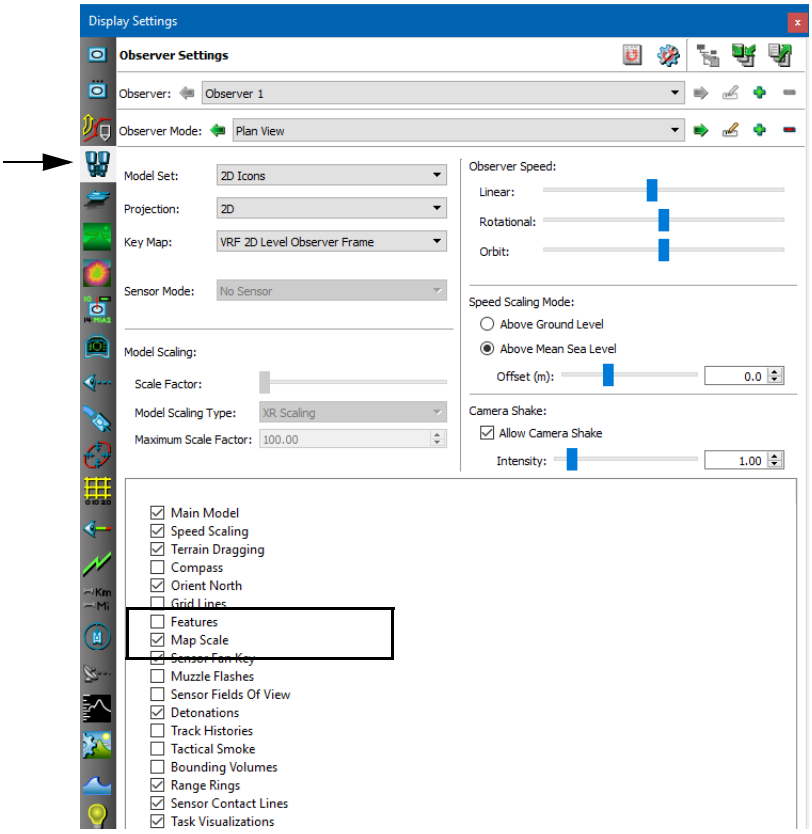


Figure 55-8. Observer Settings page, Plan View observer mode

4. Select or clear the Features check box.



You can also enable or disable display of feature data by choosing **Observer** → **Features**.

55.6.1. Displaying a List of Vector Features

If a terrain has vector features, they are listed on the Vector Network tab of the Terrain Information Panel. If you select a feature in the list, it is highlighted in the display, and the other way around (Figure 55-9).

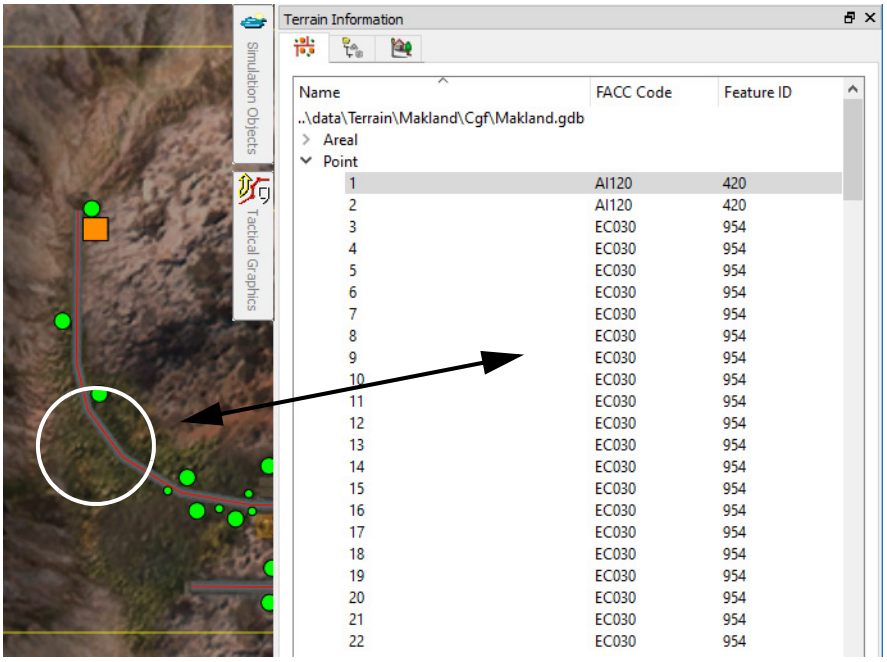


Figure 55-9. Vector feature selected

To display the feature list:

1. Choose **View** → **Terrain Information Panel**. The Terrain Information Panel opens.
2. Select the Vector Network tab (Figure 55-9).
3. Expand the top-level categories to see the lists of individual features.

55.6.2. Changing the Colors Used for Feature Data

If a terrain database contains feature data, you can change the colors used to display it. In Figure 55-10, the Woodland feature in the Little Pond terrain was changed to pink. Compare it to the default color (Figure 55-7).



Figure 55-10. Feature data with color change

To change the color of feature data:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Feature Settings page (Figure 55-11).

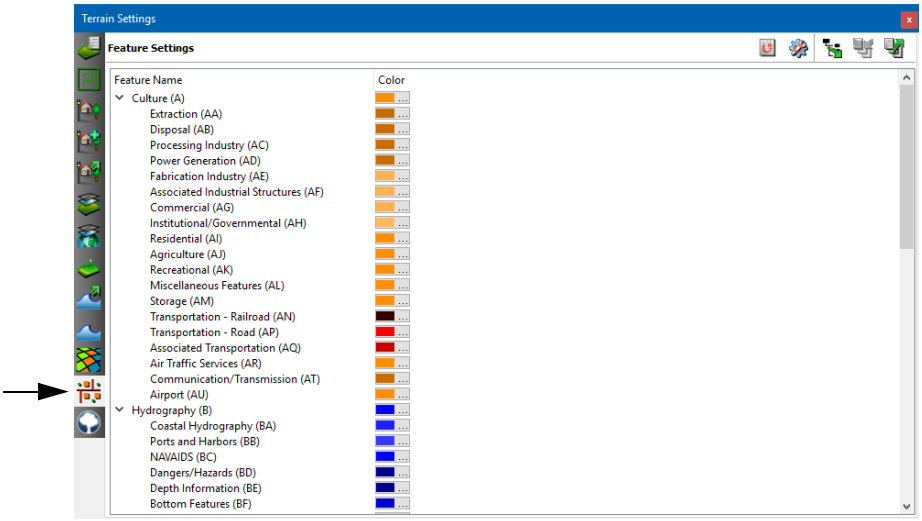


Figure 55-11. Feature Settings page

3. Click the color swatch for the feature that you want to change. A color picker dialog box opens.

4. Select the color that you want to use.
5. Click OK.

55.7. Displaying Vector Features

If a terrain has vector features, you can display them using a Vector Feature Draw Area (Figure 55-12).

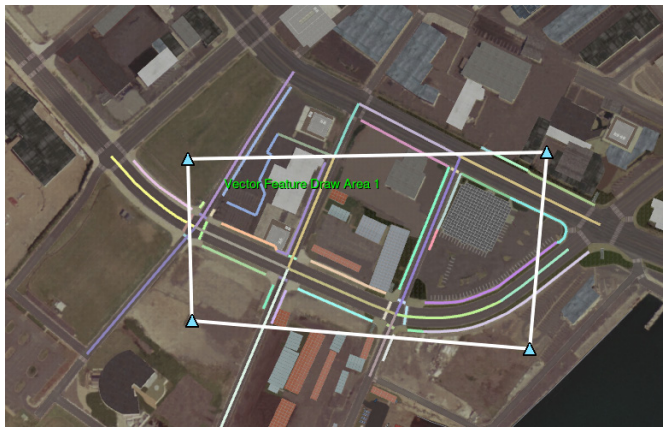


Figure 55-12. Vector feature draw area

To display vector features in a vector feature draw area:

1. On the Tactical Graphics Palette, select Vector Feature Draw Area.
2. Draw the area as you would any areal object.
3. Select the area. Vector features within the area are displayed.
4. Choose **Set** → **Feature Query Parameters**. The Feature Query Parameters dialog box opens (Figure 55-13).

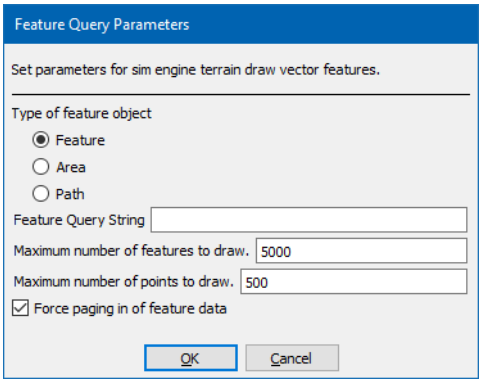


Figure 55-13. Feature query parameters dialog box

5. Specify the parameters you want to use to draw the vector features.

6. Click OK.

55.8. Displaying Coordinates and Soil Type for a Location

You can display the coordinates and soil type (if defined) of features and any point in the terrain. Soil type must be provided by the simulation engine. If you do not query the simulation engine, the Last Clicked Location Panel just displays coordinates using data from the front-end.

The Last Clicked Location Panel becomes unavailable if you query the simulation engine and it takes more than 1/2 second for a response.

To check the coordinates and soil type of a feature or location:

1. If the Last Clicked Location Panel is not open, choose **View** → **Last Clicked Location**.

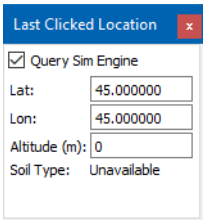


Figure 55-14. Last Clicked Location Panel

2. If you want soil type information, select Query Sim Engine.
3. Click a point on the terrain or click a feature. Information is displayed in the Last Clicked Location Panel.

55.9. Viewing the Map Scale

If you want to know the relationship between distance on your computer screen and distance on the terrain, you can view the Map Scale Toolbar (Figure 55-15). It shows the scale between screen space and world space, that is, one inch of screen = x inches of ground distance, or one meter of screen = x meters of ground. The scale changes as you zoom in and out.

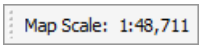


Figure 55-15. Map Scale Toolbar

➤ To enable or disable display of the Map Scale Toolbar, choose **View** → **Map Scale Toolbar**.

55.9.1. The Map Scale Overlay

In Plan View Observer mode, you can display a map scale overlay on the terrain.



Figure 55-16. Map scale overlay

➤ To view or hide the map scale overlay, choose Observer Map Scale.

55.10. Using Terrain Draw Areas to View Terrain Polygons

Terrain draw areas display the polygons known to the simulation engine (Figure 55-17). They can be useful if you are debugging terrain issues. To view polygons, you must select the terrain draw area and the scenario must be running.

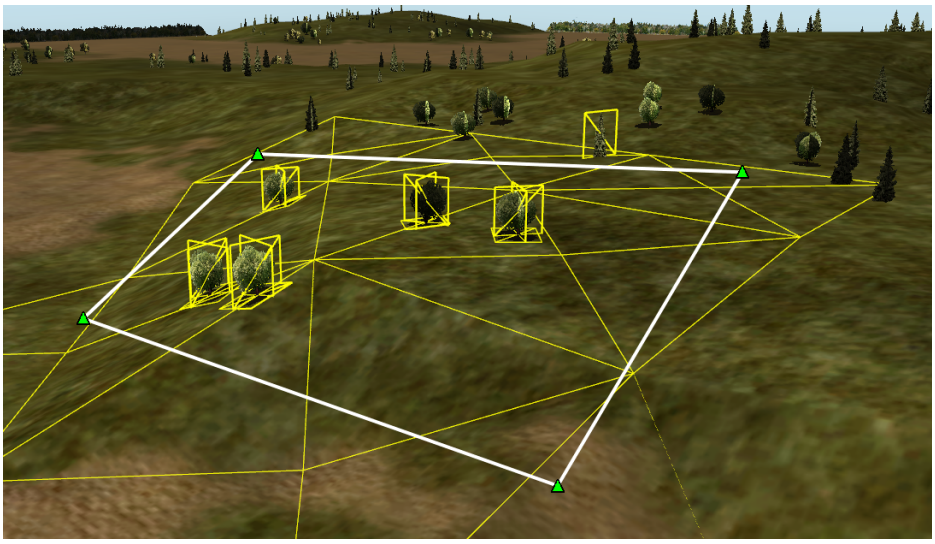


Figure 55-17. Terrain draw area

To view terrain polygons using a terrain draw area:

1. On the Tactical Graphics Palette, select Terrain Draw Area.
2. Draw an area. For details about drawing tactical graphics, please see Chapter 14, *Creating and Placing Objects*.
3. Select the area.
4. Run the scenario.

55.11. Configuring Visible Surfaces (GDB Soil Types)

If a terrain patch was loaded from a GDB file, you can display or hide GDB soil types. Figure 55-18 shows the *makland.gdb* terrain file. The view on the left shows all soil types. The view on the right has Building, Grass, Ocean, and Rock surfaces hidden.

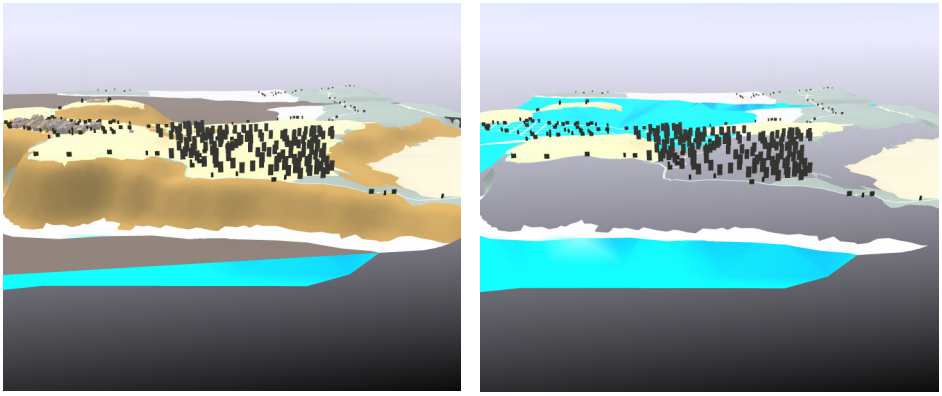


Figure 55-18. Visible surfaces toggled off

To display or hide GDB soil types:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Visible Surfaces page (Figure 55-19).

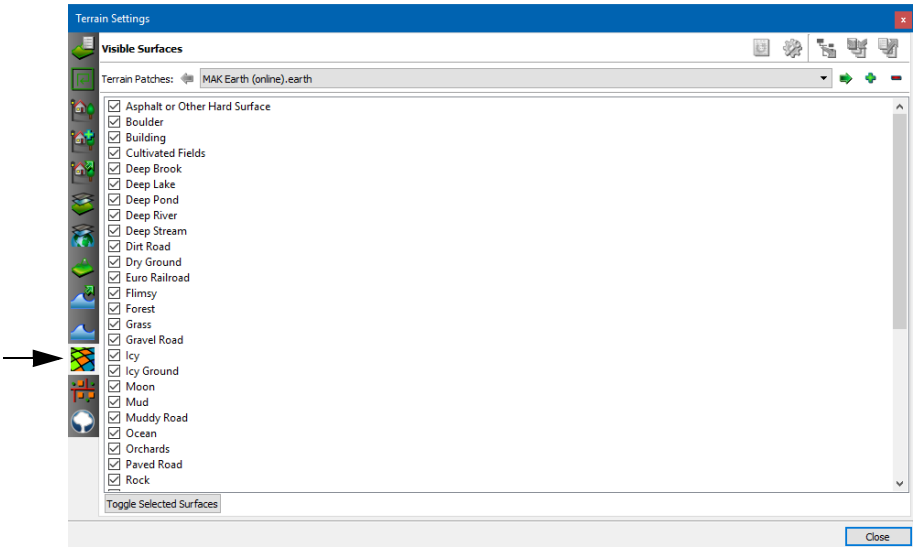


Figure 55-19. Visible Surfaces page

3. In the Terrain Patches list, select the terrain patch that you want to configure.

4. Select the check box for each soil type that you want to be visible.
5. Clear the check box for each soil type that you want to hide.
6. To change the state of multiple soil types at one time:
   a. Select (highlight) the names (not the check boxes) of the soil types whose state you want to change.
   b. Click Toggle Selected Surfaces.

55.12. Terrain Databases Provided with VR-Forces

Table 55-1 describes the terrain databases provided with VR-Forces. Some have variants, which indicate the coordinate system or other variations. Terrains with "(online)" in the title connect to VR-TheWorld Server or another online server and require an internet connection. Some terrains have variations that emphasize a particular set of features or other data. VR-Forces also has a supplemental data package that includes additional terrains. Please contact your MAK salesperson for download details.

Table 55-1: Terrain databases shipped with VR-Forces (Sheet 1 of 3)

Variation	Description
Ala Moana.mtf Ala Moana - Experimental.mtf	A high-resolution area of Honolulu, Hawaii, USA, set into a raster map of the world. This terrain is stored locally and does not require an internet connection. Ala Moana - Experimental.mtf has additional features that may reduce performance.
ArrivalsHall.mtf	An interior model of an international arrivals hall with passport control stations.
Bakersfield Topo Map (online).mtf	VR-TheWorldOnline with topographical maps for the area around Bakersfield, California, USA. This terrain contains some feature data that simulation objects in aggregate-level scenarios can take advantage of.
Brooklyn.mtf Brooklyn (online).mtf	A high resolution city center surrounded by an extruded cityscape. Located geographically in Brooklyn, N.Y., USA, but not an actual location in that city.
California (online).mtf California Air (online).mtf	A procedurally generated earth merged with high resolution imagery and elevation. California Air is optimized for viewing from altitude. It should perform a little better than California.
DestructibleTerrain.mtf	A small terrain for demonstrating destructible terrain based on switch nodes.
Ground-db.mtf	UTM. OpenFlight. A generic relatively flat ground terrain.
LittlePond.mtf LittlePond (online).mtf	Flat earth. DTED. Shape data. Imagery. A detailed suburban inset in a larger, less detailed terrain covering Boston, Massachusetts, U.S.A. and points west.
MAK Earth - Base (online).mtf	The base VR-TheWorld Server terrain with a high resolution inset for Honolulu, Hawaii, USA and the surrounding area. Does not include streaming feature data.

Table 55-1: Terrain databases shipped with VR-Forces (Sheet 2 of 3)

Variation	Description
MAK Earth (online).mtf MAK Earth - Experimental (online).mtf	VR-TheWorld Server terrain with a high resolution inset for Honolulu, Hawaii, USA and other cities. Includes streaming feature data. MAK Earth - Experimental has additional features that may reduce performance.
Makland.mtf	UTM. OpenFlight. A fictional terrain created by MAK. It includes a variety of terrain and feature data, including underwater features. It is placed in the Indian Ocean.
Massachusetts S-57 (online).mtf	A terrain that has S-57 for Boston, Massachusetts, USA harbor.
ProceduralEarth (online).mtf ProceduralEarth - OpenStreetMap (online).mtf	Procedurally generated terrains. Images are generated based on land use data, resulting in high quality visual imagery without the overhead of high resolution static images. Terrains may include airport data for all airports in the world, street data from OpenStreetMap.
RhoneValley (online).mtf	A procedurally generated view of the Rhone Valley in France.
Sadr City (online).mtf Sadr City.mtf	An urban geotypical Middle Eastern terrain in Sadr City, Iraq, a suburb of Baghdad. Ideal for urban scenarios.
SanLuisObispo.mtf	Flat earth. DTED. Imagery. A terrain that covers a portion of California, U.S.A.
SanLuisObispoTopoGdal.mtf	Geocentric. San Luis Obispo in topographic coordinates as local streaming terrain using an earth file.
Test terrains	Multiple terrains used for testing different terrain database capabilities.
TrumanCarrier.mtf	A model of an aircraft carrier as a terrain object rather than a simulation object.
VR-Village.mtf	GDB. Feature data. OpenFlight terrain. VR-Village is a fictional database created by VT MAK. It has a highly detailed small town in a larger, sparsely detailed desert environment. The town includes multi-level buildings, underground tunnels, and a road that tunnels through a mountain. The coordinates place it off the west coast of Africa.
WorldFlatEarthNoElevation.mtf	Flat earth. Low resolution projected image of the world.

Table 55-1: Terrain databases shipped with VR-Forces (Sheet 3 of 3)

Variation	Description
WorldGeocentric.mtf	Geocentric. Low resolution projection of the world.
WorldGlobalWater.mtf	A terrain with no elevation data for land masses.



56. Using Terrain Profiles

This chapter explains what terrain profiles are and how to display and configure them.

Terrain Profiles 56-2
Displaying a Line's Terrain Profile 56-3
Configuring Terrain Profiles 56-3
Analyzing Terrain Using a Terrain Profile Line 56-5

56.1. Terrain Profiles

You can display terrain profile windows for line objects that have height (in other words, routes and lines, but not phase lines), for intervisibility lines, and for a simulation object's track history. You can also create lines specifically for the purpose of showing the terrain profile. A terrain profile displays:

- A graph of the line against the terrain.
- Simulation objects within a specified distance of the line or track.
- Location of any one point along the line.
- The location on the grid of the cursor. This helps you estimate distances between the grid lines.

In Figure 56-1, the yellow line (1) represents the terrain and the red line (2) shows a route. The location of the cursor in grid coordinates is displayed (3) and the location of one point along the terrain is displayed (4). Several entities are close to the route and they are shown as well (5).

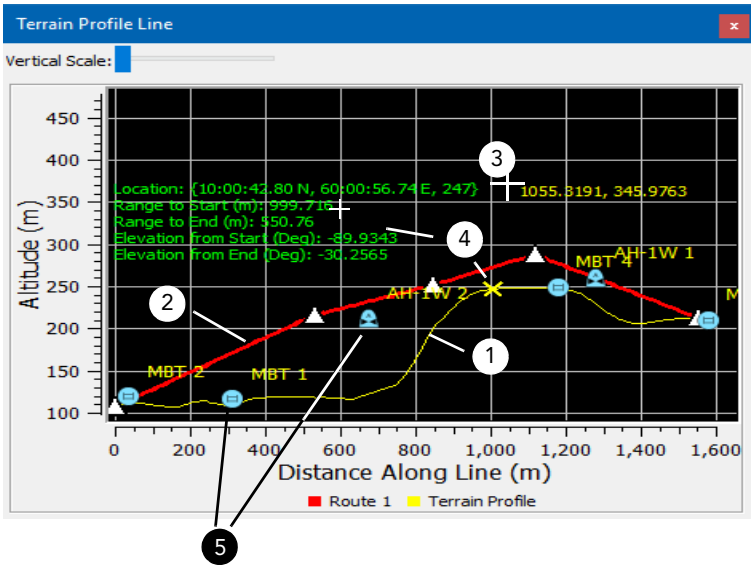


Figure 56-1. Terrain profile

56.1.1. Displaying a Line's Terrain Profile

You can display a terrain profile for routes, intervisibility lines, and terrain profile lines.

To view a line's terrain profile, use one of the following methods:

- Right-click a line and choose **Terrain Profile** from the shortcut menu.
- Open the Edit dialog box for a line and click Terrain Profile. (You can configure terrain profiles to open automatically when you open an Edit dialog box.)

To display the terrain profile for a track history, follow the procedure in "Displaying a Terrain Profile for a Track History," on page 43-7.

56.1.2. Configuring Terrain Profiles

You can configure the following aspects of a terrain profile:

- The frequency, in distance, at which terrain height is sampled to create the graph in the Terrain Profile window.
- The maximum number of points to sample along a line.
- Whether or not to show simulation objects that are close to the line and how close to the line simulation objects need to be to be shown.
- Whether or not to show the vertices of the line.
- Whether or not terrain profiles should be displayed automatically when a line is created or edited.

To configure terrain profiles:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Terrain Profile Settings page (Figure 56-2).

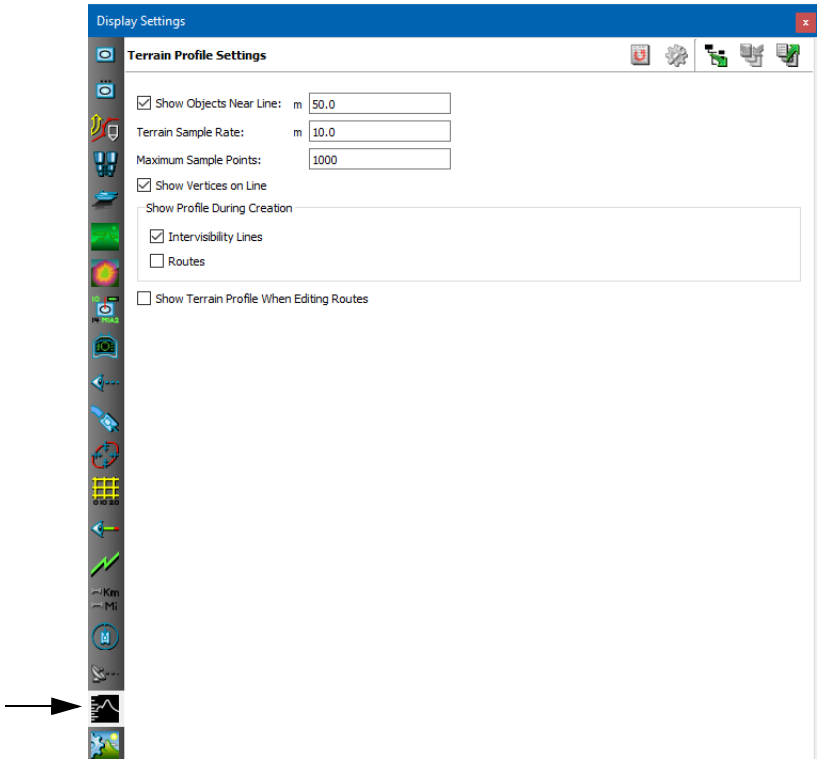


Figure 56-2. Terrain Profile Settings page

3. Change settings as desired. Table 56-1 describes the configuration options:

Table 56-1: Terrain profile parameters

Option	Description
Show Objects Near Line	If enabled, displays icons for simulation objects near the line. The text box specifies the distance from the line within which simulation objects are displayed.
Terrain Sample Rate	Specifies the frequency at which the terrain is sampled along the line.
Maximum Sample Points	Specifies the maximum number of sample points for a line.
Show Vertices On Line	If enabled, the vertices of the line are displayed.
Show Profile During Creation	If enabled, a terrain profile window opens while intervisibility lines, routes, or both are created.
Show Terrain Profile When Editing Routes	If enabled, a terrain profile window opens when you edit a route.

56.2. Analyzing Terrain Using a Terrain Profile Line

A terrain profile line is a special line used to analyze the terrain. It is more versatile than a point-to-point intervisibility line because it can have multiple points, which allows a finer analysis of the terrain. Unlike a route (for which you can also view terrain profiles), a terrain profile line is not published, so it has less overhead and does not clutter up object lists.

To create a terrain profile line:

1. Choose **Intervisibility** → **Create Terrain Profile Line**. A Terrain Profile window opens and the cursor changes to draw mode.
2. Click on the terrain to place the vertices of the line. As you add vertices, the Terrain Profile window updates to show the profile of the line (Figure 56-3).

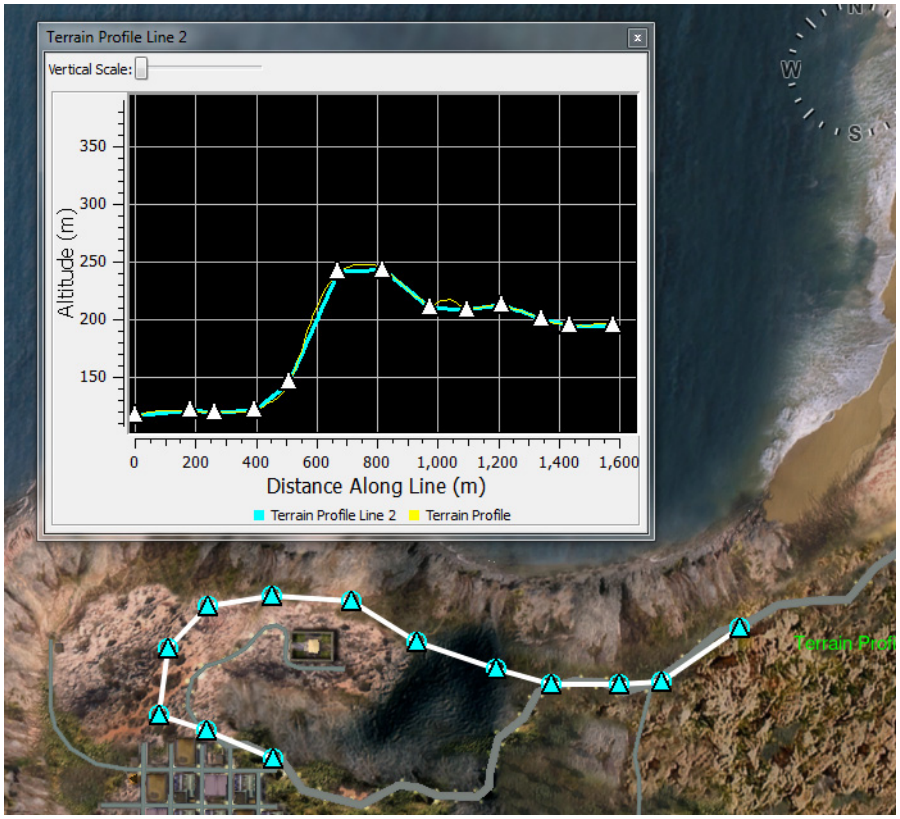


Figure 56-3. Terrain profile line

3. Right click to finish the line.

After you create a terrain profile line, you can edit it just like any other line. For details, please see "Editing Vertices," on page 40-18.



57. Composing Terrains

This chapter explains how to combine various types of data to build a terrain. For basic conceptual information about terrain agility, please see Section XI, "Terrain". For information about paged and streaming terrains, please see Chapter 58, *Paged and Streaming Terrains*.

Creating a Composed Terrain 57-2
Considerations and Limitations for Building Terrains 57-2
Saving a Terrain 57-3
Adding Elevation Data (Terrain Patches) to a Terrain 57-4
Adding Images to a Terrain 57-6
Changing the Display Order of Raster Maps 57-9
Adding a Feature Layer 57-11
Adding a Dynamic Ocean Layer 57-14
Extracting Water Textures and Adding an Ocean Layer 57-15
Adding a Dynamic Ocean Layer Directly 57-16
Setting the Ocean LOD Elevation 57-17
Configuring the Ocean Height Map 57-17
Adding Props to a Terrain 57-20
Extracting Props from a Terrain Patch 57-20
Adding Props from a Feature Layer 57-22
Viewing a List of Props 57-24
Selecting Props 57-24
Setting the Opacity of Props 57-25
Changing a Prop's Type 57-26
Changing a Prop's Position or Orientation 57-26
Changing a Prop's Model Definition 57-26
Exporting Props and SpeedTrees to a Shapefile 57-27
Importing Props and SpeedTrees from Shapefiles 57-27
Specifying Simulation Ocean Settings 57-28

57.1. Creating a Composed Terrain

The most common ways to compose a terrain in VR-Forces are as follows:

- Load individual terrain components through the VR-Forces GUI and then save the terrain as an MTF file.
- Create an earth file that specifies all the terrain components to load. Then load the earth file and save it as an MTF file.
- Load an earth file and add local terrain components. Then save the terrain as an MTF file.

This section describes the first option, creating a terrain by loading individual elevation, imagery, and feature files. (Using earth files is the preferred method. For details about using earth files, please see Chapter 58, *Paged and Streaming Terrains* and Chapter 59, *Loading Data Using Earth Files*.) The general procedure for composing a terrain this way is as follows:

1. Choose **File** → **Terrain** → **New Terrain**, or click the New Terrain button () on the Terrain Toolbar. If any terrain data is loaded, it is removed.
2. Add elevation data, as described in "Adding Elevation Data (Terrain Patches) to a Terrain," on page 57-4.
3. Optionally, add imagery, as described in "Adding Images to a Terrain," on page 57-6.
4. Optionally, add feature data, as described in "Adding a Feature Layer," on page 57-11.
5. Optionally, add a dynamic ocean layer, as described in "Adding a Dynamic Ocean Layer," on page 57-14.
6. Optionally, add props or extract props, as described in "Adding Props from a Feature Layer," on page 57-22 and "Extracting Props from a Terrain," on page 66-8.
7. Save the terrain, as described in "Saving a Terrain," on page 57-3.

For a list of supported terrain and feature formats, please see "Supported File Formats," on page 54-2.

57.1.1. Considerations and Limitations for Building Terrains

As you compose a terrain, consider the following issues:

- When you load DTED as a terrain patch, VR-Forces supports DTED level 0, 1, and 2, as follows:
  - DTED is loaded using a Flat Earth projection.
  - Level 0 DTED files have 1 KM post spacing. All posts are loaded.
  - Level 1 DTED files have 100 M post spacing. Every fifth post is loaded.
  - Level 2 DTED files have 20 M post spacing. Every fifth post is loaded.

  The maximum cell size you can load is 1 degree.
- The larger a terrain database is (as measured by the number of polygons), the more it will affect performance.

Loading DTED

By default, when you load DTED files as terrain patches, the files are loaded with their altitude referenced to WGS84. If you want the altitude to reference a different vertical datum (most likely EGM96), you must provide a geoid grid file for that vertical datum. If you use a geoid grid file, the altitudes in the DTED file are adjusted based on the information in the grid file.

To use a geoid grid file, set the environment variable MAK_GEOID_GRID to the path of the file.

For information about EGM96 and geoid files, please see the following URL:

http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm96/egm96.html

57.1.2. Saving a Terrain

When you save a terrain, VR-Forces saves information about the terrain patches, imagery, and feature layers that you used to create the terrain. Saving a terrain does not save out data to a file format comparable to OpenFlight or GDB that can be loaded into other applications or converted into other formats. If accelerated file loading is enabled, the data gets saved in a format that can be loaded more quickly than from its native format. (For details about accelerated file loading, please see "Configuring File Caching," on page 64-2.) Terrains get saved with the extension *.mtf*.



When you save a terrain to MTF, if an MTD file of the same name exists, a dialog box explains your options for saving the file. (MTD is a legacy format that might be present if you have terrains created with older versions of VR-Forces.)

To save a terrain:

1. Choose **File** → **Terrain** → **Save Terrain**, or click the Save Terrain button () on the Terrain Toolbar. The Save Terrain dialog box opens.
2. Type a name for the terrain.
3. Click Save.

57.2. Adding Elevation Data (Terrain Patches) to a Terrain

To build a terrain database, you load elevation data, imagery, and feature data. Then you save the data collection in MTF format. Elevation data is added as terrain patches. Note the limitations in "Considerations and Limitations for Building Terrains," on page 57-2.

! Each time you add a terrain patch, VR-Forces resets the origin and coordinate system. Therefore, if you add multiple terrain patches, any patches whose origin and coordinate system do not match the patch that is loaded last will not be usable.

To add a terrain patch to a terrain database:

1. Choose **Settings** → **Terrain**, or click the Add Terrain Patch button () on the Terrain Toolbar. The Terrain Settings dialog box opens.
2. Select the Terrain Contents page (Figure 57-1).

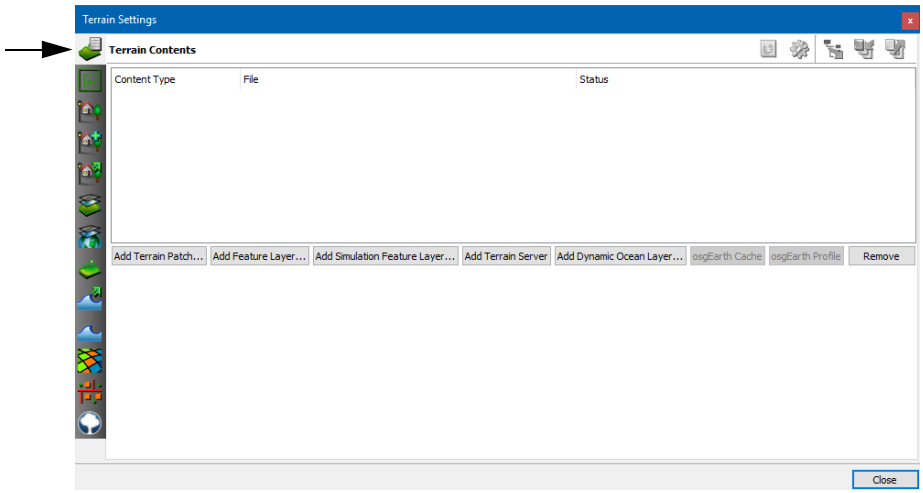


Figure 57-1. Terrain Contents page

3. Click Add Terrain Patch. The Add Terrain Patch dialog box opens and an Open dialog box opens on top of it.
4. Select a file in a supported format.
5. Click Open. The Add Terrain Patch dialog box displays details about the terrain patch (Figure 57-2).

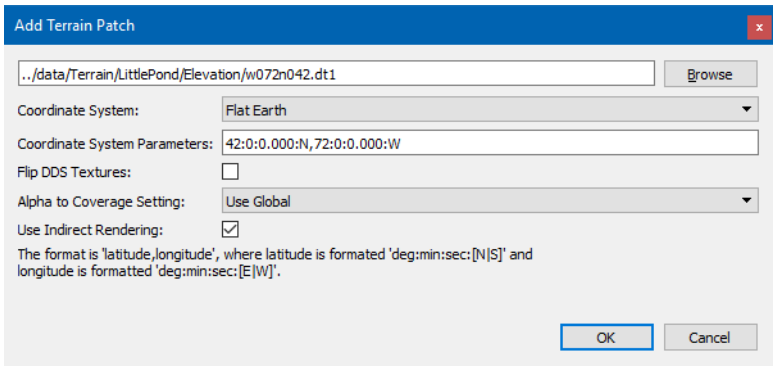


Figure 57-2. Add Terrain Patch dialog box

6. If the terrain patch has DDS textures that are upside down relative to VR-Forces's expectations, select Flip DDS Textures. This setting does not affect paged terrains. (For information about DDS textures, please see "Displaying DDS Textures Correctly," on page 64-17.)
7. Optionally, change the Alpha to Coverage Setting. The options are:
   - Use Global. Use the setting for alpha to coverage from the Display Settings dialog box, Render Settings page.
   - Force Off. Use the regular rendering way of handling transparent objects.
   - Force On. Turn on alpha to coverage for the entire terrain patch (excluding props since they are handled as models).

   For more information about alpha to coverage, please see "Alpha to Coverage," on page 6-17.
8. If you want this terrain patch to support indirect rendering, select the Use Indirect Rendering check box. (Use of indirect rendering improves performance for most terrains. For more information, please see "Indirect Rendering," on page 64-14.)
9. Click OK. The terrain patch is added.
10. Optionally, add additional terrain patches.
11. Save the terrain.

57.3. Adding Images to a Terrain

Depending on your terrain source, you may have terrain polygonal data, but no texture data or feature data to provide a realistic 3D display. You can drape images (raster maps) over the terrain to provide a more pleasing display.

Adding very large images (multi-100 GB) may cause VR-Forces to crash. If you need to load very large images, load them using an earth file and the GDAL driver. The *SanLuisObispo-Imagery.earth* file demonstrates how to load an image using this method. You can also use the GDAL driver to load raster formats that VR-Forces cannot load directly. For a list of formats supported by GDAL, go to http://www.gdal.org/ and select the Supported Formats link.

To add raster maps to the terrain:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Raster Maps page.
3. Click the Add button (+) that is above the list of raster maps. A line is added to the Raster Maps list (Figure 57-3).

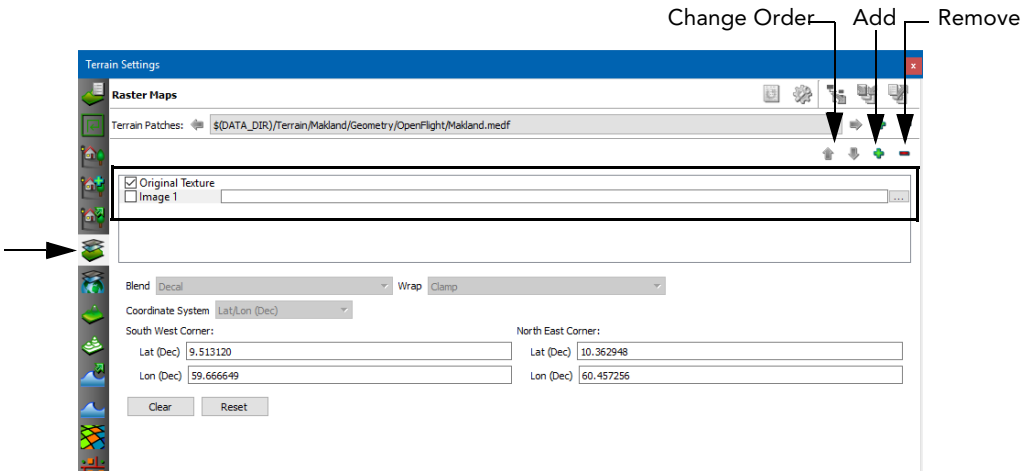


Figure 57-3. Raster Maps page

4. Click the Browse button to the right of the new line. The Image File dialog box opens.
5. Select the raster image that you want to add to the terrain.
6. Click OK. The image is listed in the Raster Maps window. To view it in the main window, select the check box to the left of the entry (Figure 57-4).

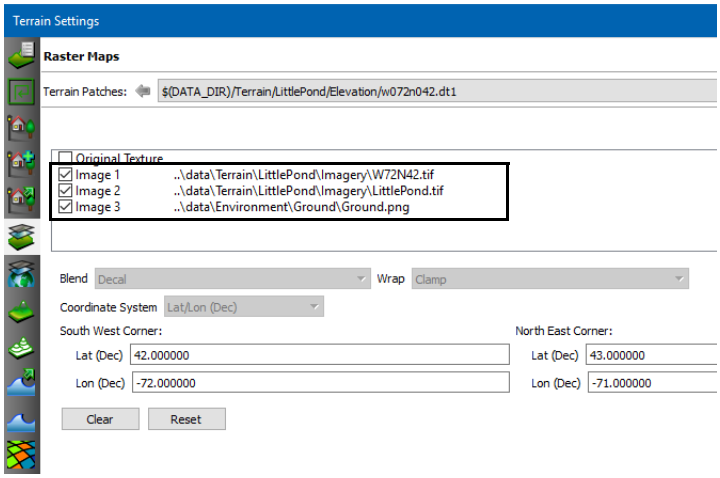


Figure 57-4. Making an image visible

7. Optionally, specify the attributes of the image, as follows:
   – Blend specifies how the image interacts with underlying images, as described in Table 57-1:

Table 57-1: Blend settings

Blend setting	Description
Decal	Covers the polygons with the image. Use it to apply an opaque texture or a texture with transparency. (This is done using an alpha channel.)
Modulate	Color values of the image modify the intensity of the colors of the polygon.
Blend	Blends polygon and texture colors.
Replace	Similar to decal.
Add	Adds the pixel values of the source and base colors.

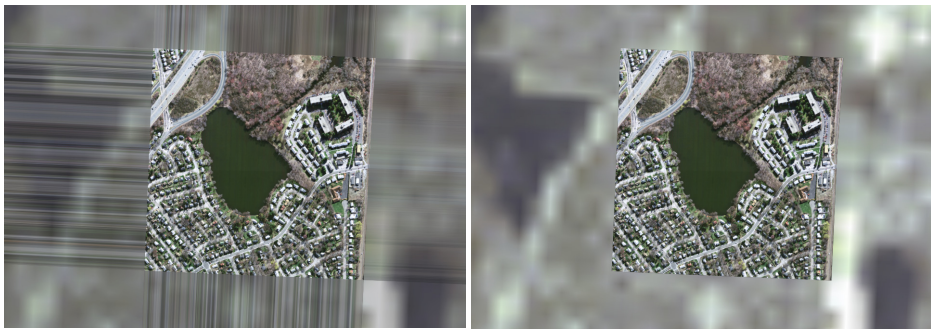
– Wrap specifies whether or not the image is repeated, and if so, how it is repeated, as described in Table 57-2, and as illustrated in Figure 57-5 and Figure 57-6:

Table 57-2: Wrap settings

Wrap setting	Description
Clamp	Extends the border pixels of the image to the edge of the terrain or the next continuous image, blending them with the pixels of other images. It does not obscure underlying images.
Clamp to Edge	Extends the border pixels over the entire terrain, obscuring any other image.
Clamp to Border	Places the image once, at the specified coordinates.
Repeat	Tiles the image onto the terrain.
Mirror	Tiles the image onto the terrain, alternating between the true image and a mirror image.

– Coordinate System specifies the coordinate system to use to place the image. Most options are self-explanatory. The Texture coordinate system specifies UV (X,Y) coordinates for placing textures on the terrain.

– Location specifies the coordinates for the Southwest (SW) and Northeast (NE) corners of the image. If the image is a GeoTIFF, the coordinates are in the image file. If the image file does not have coordinates, VR-Forces stretches it to cover the terrain patch. If you do not want the image to cover the entire terrain, you can specify the coordinates.



Clamp | Clamp to border

Figure 57-5. Raster image wrap effects (Clamp and Clamp to Border)

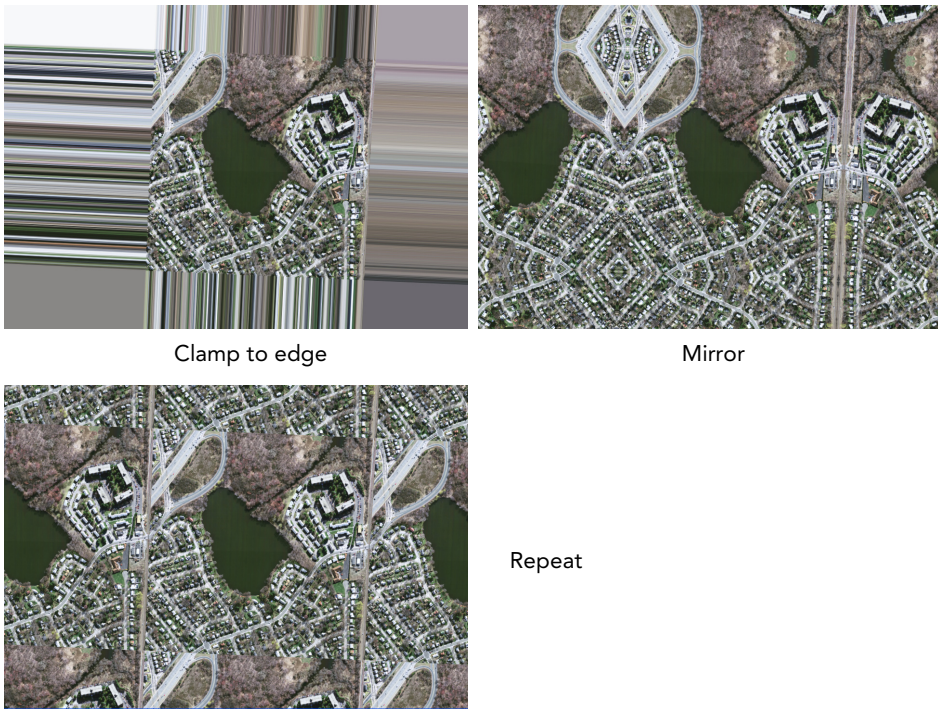


Figure 57-6. Raster image wrap effects (Clamp to Edge, Mirror, Repeat)

57.3.1. Changing the Display Order of Raster Maps

Raster maps are displayed in the order in which they are listed in the Terrain Settings dialog box, Raster Maps page. If they overlay the same portion of the terrain, the topmost image is displayed. You can change the order in which image files are listed, thereby changing which one is displayed.

Figure 57-7 shows two views of the detailed portion of the LittlePond terrain, one with the default image order, and one with the order of raster images reversed.



Figure 57-7. Changing the raster map order

To change the order of a raster map.

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Raster Maps page (Figure 57-3).
3. Select the raster image whose order you want to change.
4. Click the Up Arrow or Down Arrow above the upper right corner of the list of raster maps. The image moves in the list and the window changes as appropriate.

57.4. Adding a Feature Layer

A feature layer is a file that contains data about geographic features on the terrain. This includes linear features, such as roads, and point features, such as trees and buildings. You can create props from point features. A feature layer can be a MAK Terrain Format (GDB) file, a shapefile (SHP), or DFAD data (DFD).

The front-end and back-end use different mechanisms to import feature data and they handle it differently.

When the front-end imports a shapefile, it uses the name of the shapefile to indicate what type of feature it contains. Filenames are mapped to feature types in *./appData/importConfig/filename_shp_map.txt*. For example, the file name *ROADL.shp* is mapped to feature 129, which is a road. If the shapefiles that you need to import are not listed in *filename_shp_map.txt*, you can add them by editing the file.

Loading large feature layers into the front-end can cause it to hang and scenario data can be lost. To avoid this problem, you can choose to load feature layers only in the simulation engine. In this case, you will not be able to visualize the feature data, but the simulation engine will handle it correctly.

When the back-end imports feature data, it configures it based on a configuration file called *./appdata/settings/featureconfig.txt*. For details about this file, please see Chapter 65, *Configuring Feature Data*.

The preferred method for adding feature data in the back-end is via shapefiles. Loading shapefiles in the back-end, as compared to loading feature data from a GDB file, provides better performance. If you have feature data in GDB files and you need to import it into VR-Forces, you can extract it from the GDB to a shapefile using the vectorNetworkToShapeFiles utility. For details, please see "Extracting Feature Data from GDB to Shapefiles," on page 65-7.



- Adding a feature layer does not, by itself, make any visible difference to a terrain. To see the feature data, you must extract the data as props and map the props to model definitions. For details, please see "Adding Props to a Terrain," on page 57-20.
- The number of layers you can add depends on your graphics card. If you add a layer and it does not appear, try reducing the number of layers you have.
- Feature layers in earth files are added automatically.
- VR-Forces supports S-57 data. The recommended way to load S-57 data is to convert it to shapefiles and load the shapefiles. You could load them as feature layers as described in this section. However, since each S-57 file is converted to multiple shapefiles, it may be easier to load them using earth files than as individual shapefiles. VR-Forces includes one terrain that has S-57 data, *Massachusetts S-57 (online).mtf*.

To add a feature layer:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Terrain Contents page (Figure 57-8).

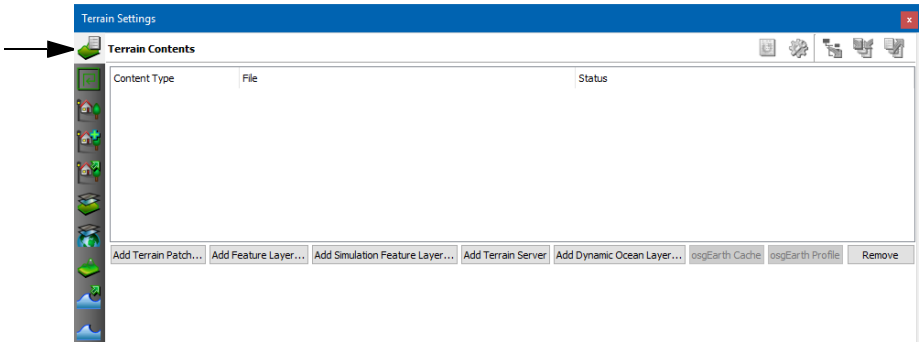


Figure 57-8. Terrain Contents page

3. Click Add Feature Layer to load a feature layer in the front-end and back-end. Click Add Simulation Feature Layer to add a feature layer only in the back-end. The Select Source File for Feature Layer dialog box opens.
4. In the Select Source File for Feature Layer dialog box, select the file that you want to use for this feature layer.
5. Click Open. The source file is added to the list of terrain contents. The Status column indicates whether the feature layer is for visualization and simulation (Visualized) or just simulation (Not Visualized)(Figure 57-9).

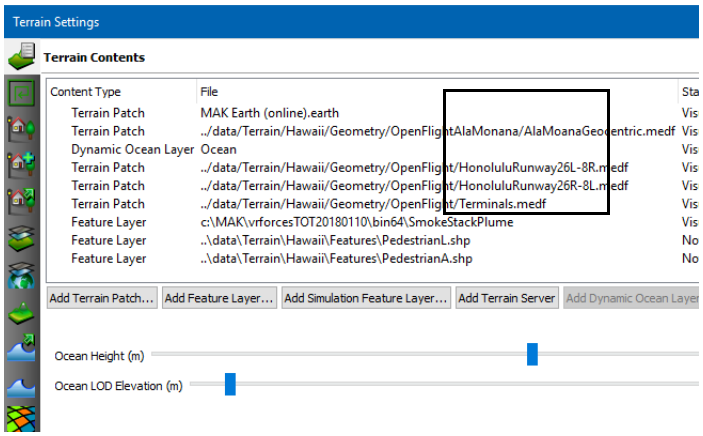


Figure 57-9. Terrain contents status

57.5. Adding a Dynamic Ocean Layer

VR-Forces can simulate dynamic ocean effects, such as waves, swell, and chop. To simulate the ocean, your terrain must have a dynamic ocean layer. You can add an ocean layer directly or you can add one as part of the process of extracting water textures.

If your terrain has textures assigned to water polygons, then you should extract them. They will be replaced by the ocean layer. If you do not extract the water textures and you simply add an ocean layer, the ocean layer and the water texture will interfere with each other in a process called Z-fighting and create an unacceptable display (Figure 57-10). This problem occurs because the ocean layer and the texture are both at the same height (0, or sea level).

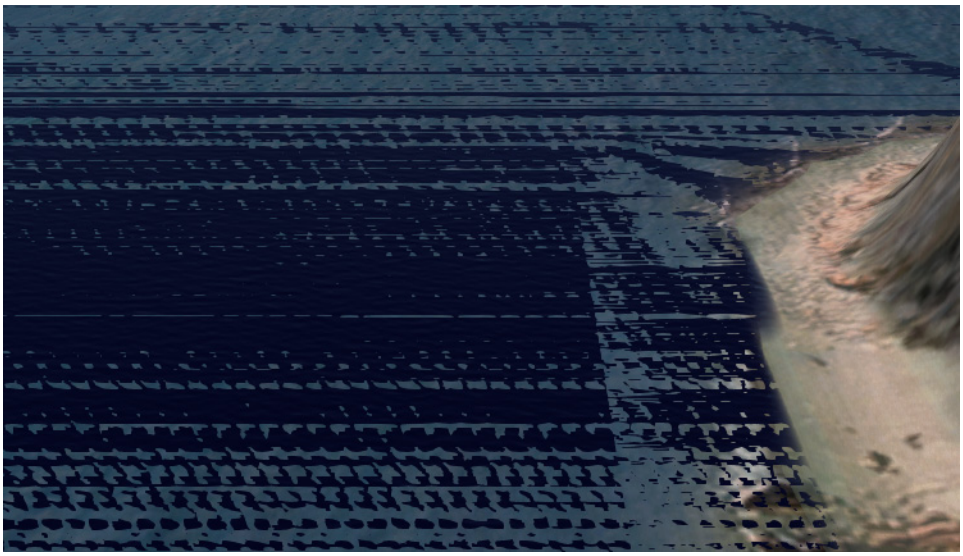


Figure 57-10. Z-fighting

If your terrain does not have a specific water texture assigned to polygons, your only option is to add an ocean layer directly. However, since the ocean layer will be at the same elevation as the terrain texture, you may experience Z-fighting. You may also have to raise the ocean height to actually see the ocean effects.

Figure 57-11 shows the Makland terrain with a water texture and an ocean layer. In the left view, the ocean layer has a height of 0 meters. You cannot see it. In the right view, the ocean layer is set to 1 meter. Now it is visible.

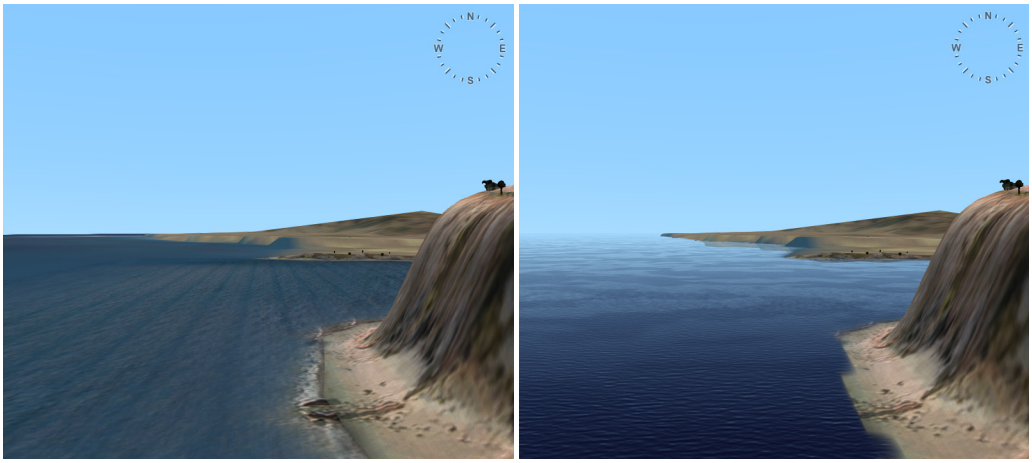


Figure 57-11. Ocean height

57.5.1. Extracting Water Textures and Adding an Ocean Layer

You cannot extract water textures when indirect rendering is enabled.

To extract textures and add an ocean layer:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Extract Ocean page (Figure 57-12).

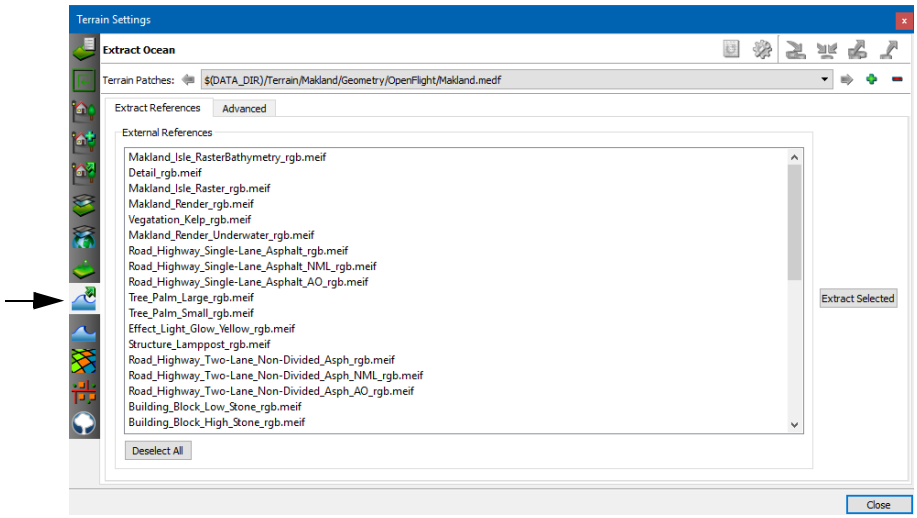


Figure 57-12. Terrain Settings dialog box, Extract Ocean page

3. In the External References list on the Extract References tab, select the water textures that you want to extract.
4. Click Extract Selected. The textures are removed and an ocean layer is added to the Terrain Contents page.

5. Select the Terrain Contents page (Figure 57-13).

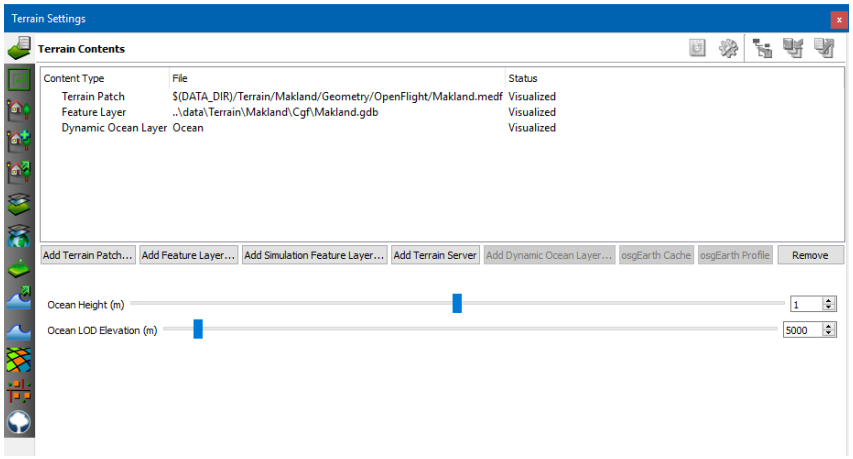


Figure 57-13. Terrain Contents with dynamic ocean layer

6. Adjust the Ocean Height so that the dynamic ocean is visible. This might require raising the ocean above sea level.
7. Optionally, adjust the Ocean LOD Elevation. For information about the Ocean LOD Elevation, please see "Setting the Ocean LOD Elevation," on page 57-17.

57.5.2. Adding a Dynamic Ocean Layer Directly

Ocean layers get created at a height of 0 meters, that is, mean sea level.

To add a dynamic ocean layer:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Terrain Contents page.
3. Click Add Dynamic Ocean Layer. (You can only have one ocean layer.) A layer is added to the Terrain Contents list and two parameters are added to the bottom of the dialog box (Figure 57-13).
4. Adjust the Ocean Height so that the dynamic ocean is visible. This might require raising the ocean above sea level.
5. Optionally, adjust the Ocean LOD Elevation. For information about the Ocean LOD Elevation, please see "Setting the Ocean LOD Elevation," on page 57-17.

57.5.3. Setting the Ocean LOD Elevation

Enabling dynamic ocean affects performance. Since the effects are hard to see once the observer is high above the ocean, VR-Forces stops displaying them when the observer is more than 5000 meters above sea level. You can change this cutoff point by adjusting the Ocean LOD Elevation.

To set the ocean LOD elevation:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Terrain Contents page (Figure 57-13).
3. Adjust the Ocean LOD Elevation slider or change the value in the box.

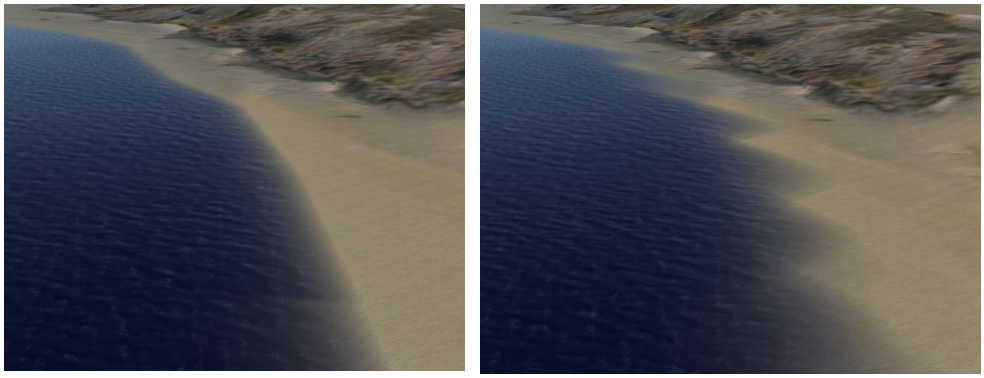
57.5.4. Configuring the Ocean Height Map

The ocean height map is used to blend the dynamic ocean with the coastline. You can configure the height map resolution, its texture size, and the threshold for regenerating the height map. All of these settings interact to affect visual quality and performance.

Height map resolution is specified as meters per pixel. The height map texture size is specified in pixels. Therefore these two settings determine the length of one side of the (square) area that the height map covers. For example, if the value for height map resolution is 10 meters per pixel and the texture size is 4096 pixels, this results in a height map that is 40960 meters on a side.

As height map resolution increases and texture size decreases, visual quality may improve, but performance can degrade. Conversely, decreasing resolution and increasing the texture size can improve performance at a cost of visual quality. You will need to experiment to determine the best values for your simulation.

If you are working with a small terrain or scenes in which the observer is close to sea level, you may want to use a higher resolution. If you are less interested in how the coastline looks, you can use a lower resolution. Figure 57-13 shows a coastline at different levels of resolution.



1.0 15.0

Figure 57-14. Ocean height map resolution

Setting the surface transparency also affects visualization of the coastline. Experiment with these values to achieve the best results. For details about water transparency, please see "Configuring Marine Conditions," on page 11-24.

Setting the Ocean Height Map Resolution

The ocean height map resolution is specified at meters per pixel.

To set the ocean height map resolution:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 57-15).

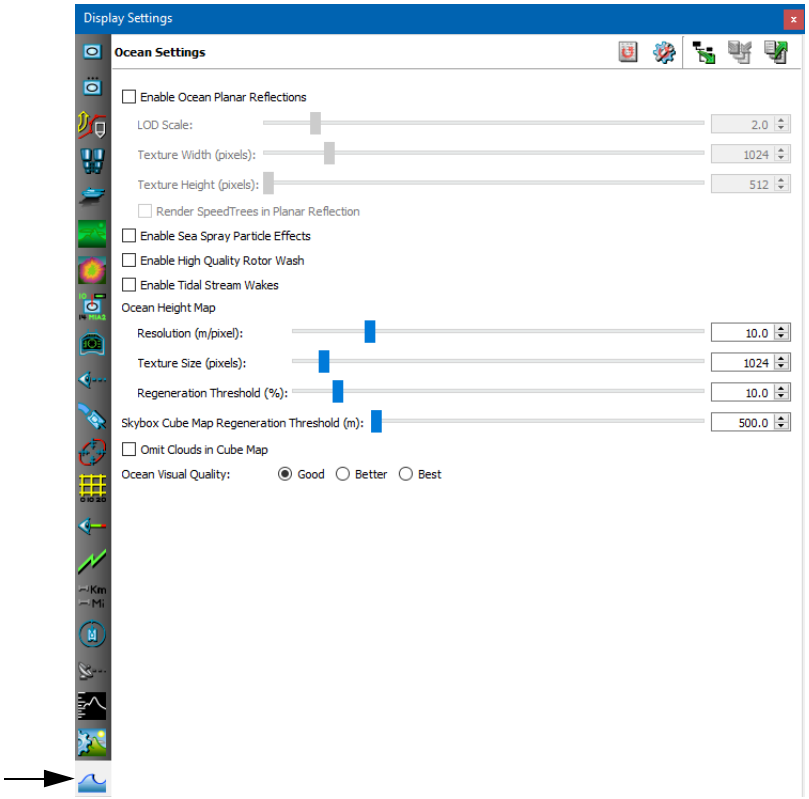


Figure 57-15. Ocean Settings

3. Adjust the Ocean Height Map Resolution slider, or enter a value in the box.

Setting the Ocean Height Map Texture Size

The ocean height map texture size is specified in pixels.

To set the ocean height map texture size:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 57-15).
3. Adjust the Ocean Height Map Texture Size slider, or enter a value in the box.

Setting the Ocean Height Map Regeneration Threshold

The ocean height map is recalculated as the observer moves. Generating the height map is computationally expensive. Therefore, it is done only when the observer moves by more than a specified threshold. You can adjust this threshold if you want to recalculate the height map more frequently. The default is 10% (0.1). This means that the observer must move by 10% of the length of one side of the height map before the height map is regenerated.

To set the ocean height map regeneration threshold:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 57-15).
3. Adjust the Ocean Height Map Regeneration Threshold slider, or enter a value in the box.

57.6. Adding Props to a Terrain

Props are objects like buildings, lampposts, or trees that exist independently of the underlying terrain elevation data. You can extract props from a terrain patch or add them from a feature layer. You can view a list of the props in the terrain, specify their opacity, and enable or disable their visibility. You can change a prop's type and you can change the model definition used to render it. You can also create new prop types to meet your simulation needs.



In VR-Forces you can create buildings and cultural feature objects from the Entity Palette that look like similarly named props in the 3D view. Cultural features are not the same things as props. Props are saved as part of the terrain. Cultural features are simulation objects and are saved as part of the objects list file. They are a mechanism for placing objects into a scenario for visualization or simulation purposes.

57.6.1. Extracting Props from a Terrain Patch

When you extract props from a terrain patch, you have to tell VR-Forces where to find prop data, what type of prop to use, and what model definitions to map the props to, as follows:

- Where to search: A terrain patch, such as an OpenFlight terrain, often references multiple files that define models that are incorporated into the geometry of the terrain. VR-Forces lets you specify the external files from which to extract props or you can specify a search expression that looks in all referenced files. If you select a referenced file, the process is quicker because VR-Forces searches for props only in the specified file.
- What prop type to use: You can select a prop type from a list, specify a type of your choice, or let VR-Forces auto-generate prop types. If you choose Auto Generate, VR-Forces makes the prop type match the name of the referenced file. For example, a prop extracted from the file *Bridge_Arch_Stone.flt* would be given the prop type *Bridge_Arch_Stone.medf*.
- Model definition to use: You can select a model definition from a list, or choose Use Extracted Geometry. If you accept the default value, Use Extracted Geometry, VR-Forces creates props using the geometry in the terrain patch. For example, if you chose *Trees.flt* as the referenced file, VR-Forces creates props using the tree geometry in that file.

VR-Forces extracts props only from the selected terrain patch. If your terrain has multiple terrain patches, you must repeat the procedure for each terrain patch from which you want to extract props.

For an example of how to extract props from a terrain patch, please see "Extracting Props from a Terrain," on page 66-8.

To extract props from a terrain patch:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Extract Props page (Figure 57-16).

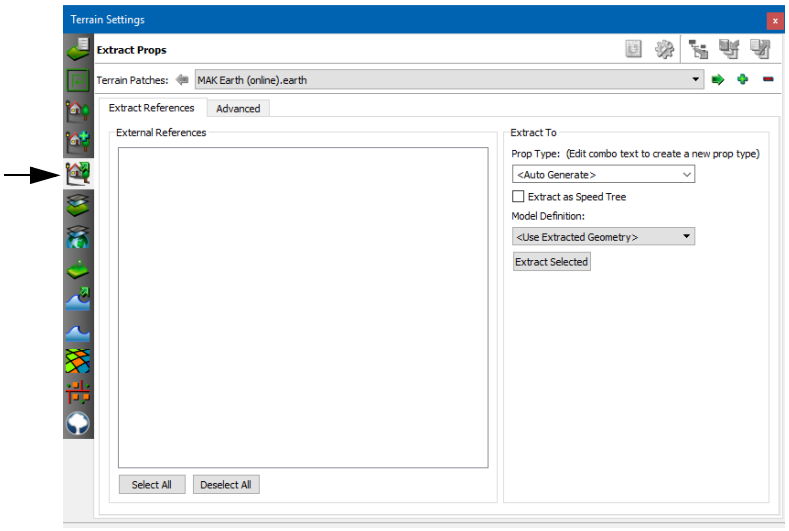


Figure 57-16. Extract Props page

3. In the Terrain Patches box, select the terrain patch from which you want to extract props.
4. To extract props by specifying external files, select the Extract References tab. To extract props using a search expression, select the Advanced tab.
5. If you chose the Extract References tab, in the External References list, select the files from which you want to extract props.

   If you chose the Advanced tab, type the search expression in the Search Expression box. Select the Regular Expression check box if you are using a regular expression.



VR-Forces uses perl regular expression syntax.

6. Optionally, in the Prop Type box, select the type that you want to assign to the extracted props or type a prop type.
7. Optionally, in the Model Definition box, select the model definition that you want to apply to the extracted props.
8. If you are using the Advanced tab, click Extract. If you are using the Extract References tab, click Extract Selected. VR-Forces extracts the props, if any. You can view the list of props in the Props list on the Edit Existing Props page.

57.6.2. Adding Props from a Feature Layer

A feature layer source file contains information about point, linear, or areal features. Feature information is stored in a table in which each row is a feature and each feature is defined by the values in an arbitrary number of columns. To add a prop from a feature layer, you must build a rule that specifies the column headings and values for the feature you want to import. Then you must associate a prop type and model definition with the prop. You also must specify whether or not to ground clamp the prop.

To add a prop from a feature layer:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Add Props page (Figure 57-17).

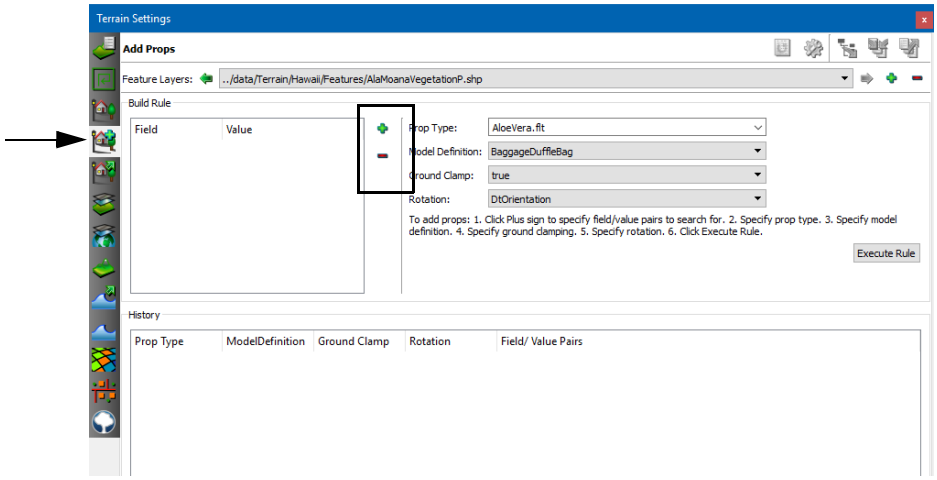


Figure 57-17. Terrain Settings, Add Props page

3. In the Feature Layers list, select the feature layer from which you want to import the props.
4. Click the Add button (+) to the right of the Field/Value window (Figure 57-17). The Build Field/Value Pair dialog box opens (Figure 57-18).

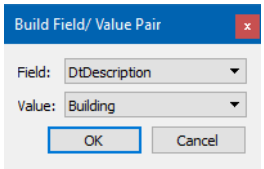


Figure 57-18. Build Field/Value Pair dialog box

5. In the Field list, select a field. Fields are the column headings in the feature definition.
6. In the Value list, select a value. Values are one or more values from the definition table that are available for the column heading.



It is your responsibility to understand the feature definitions for the features that you want to import as props.

7. Click OK. The Field/Value pair is added to the window (Figure 57-19).

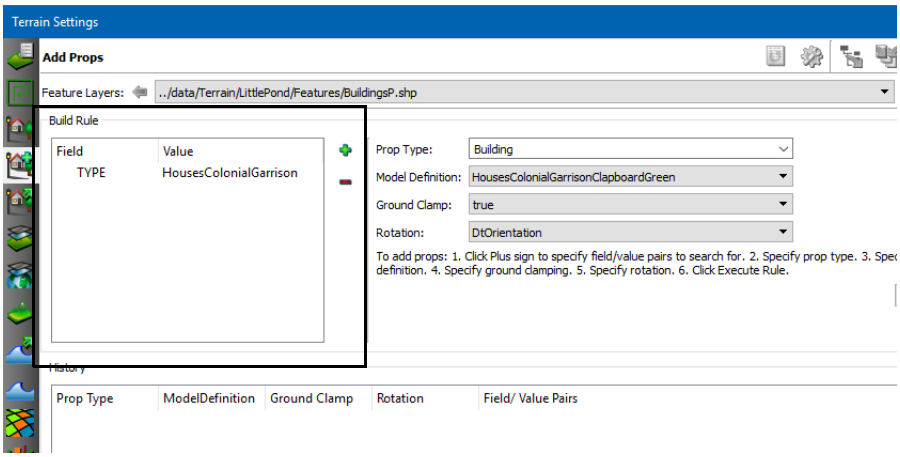


Figure 57-19. New Field/Value pair

8. Add additional Field/Value pairs as needed to define the prop that you want to add.
9. In the Prop Type list, select the prop type to assign to the prop you want to add.
10. In the Model Definition list, select the model definition you want to use for this prop.
11. In the Ground Clamp box, specify whether or not to ground clamp this prop.
12. Optionally, set the rotation.
13. Click Execute Rule. The prop is extracted from the feature layer and is listed in the History window.

57.6.3. Viewing a List of Props

To view a list of the props in the terrain:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 57-20).

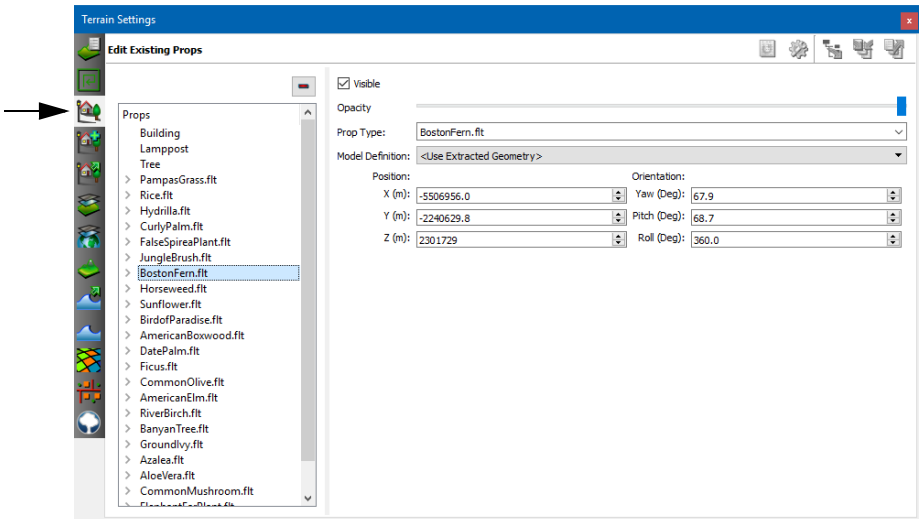


Figure 57-20. Edit Existing Props page

3. Expand the list to view the entries under a particular type of prop.

57.6.4. Selecting Props

➤ To select a prop, click it in the window or select it in the list of props on the Terrain Settings dialog box, Edit Existing Props page.

Press **CTRL**+click to select multiple props.

57.6.5. Setting the Opacity of Props

You can set the opacity of props. This can be useful if you want to see what is inside a building or behind a prop. Figure 57-21 illustrates a building at two levels of opacity.



100% 40%

Figure 57-21. Prop opacity

To set a prop's opacity:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 57-20).
3. In the list of props select the props whose opacity you want to change, or select a prop in the window. The Opacity slider is enabled.
4. Slide the Opacity slider to the right to increase opacity. Slide it to the left to decrease opacity.

Making a Prop Fully Transparent or Opaque

You can quickly set a prop to be fully transparent or opaque.

➤ To make a prop fully transparent, right-click the prop and on the shortcut menu, choose **Make Transparent**.

➤ To make a transparent prop fully opaque, right-click the prop and on the shortcut menu, choose **Make Opaque**.

57.6.6. Changing a Prop's Type

Every prop has a type. Props are organized by type in the list on the Edit Existing Props page.

To change a prop's type:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 57-20).
3. In the list of props, select the props whose type you want to change.
4. Select a prop type from the Prop Type list, or type a new type name. The selected props are moved to the new prop type in the Props window.

57.6.7. Changing a Prop's Position or Orientation

You can change a prop's position and orientation. You can change the orientation of multiple props at the same time without affecting their position. However, if you select multiple props and change position, they are all placed in the same position.

To change a prop's position or orientation:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 57-20).
3. In the list of props, select the prop whose position or orientation you want to change.
4. To change position, type or select new values in the X, Y, or Z boxes.
5. To change the orientation, type or select new values in the Yaw, Pitch, or Roll boxes.

57.6.8. Changing a Prop's Model Definition

You can change the model definition used to render a prop. For information about model definitions, please see Chapter 87, *Model Definitions*.

To change a prop's model definition:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 57-20).
3. In the list of props, select the prop whose model definition you want to change.
4. In the Model Definition list, select a model definition. The selected props are rendered using the new model definition.

57.6.9. Exporting Props and SpeedTrees to a Shapefile

You can export props and SpeedTrees to a shapefile. This is particularly useful if you have added props from the Props Palette and want to include them in a terrain, but do not want to save them as part of the terrain. You can then add the props to a terrain from an earth file.



When you export props, VR-Forces automatically exports props and SpeedTrees as one operation. If you are exporting to a new file, VR-Forces takes the name you enter and appends BuildingsP.shp and VegetationP.shp to create the shape files. For example, if you specify the file name myShapeFiles, VR-Forces saves to *myShapeFilesBuildingsP.shp* and *myShapeFilesVegetationP.shp*.

I you select an existing file as the export target, VR-Forces uses that file name. It still saves both buildings and trees.

To export props or SpeedTrees to a shapefile:

1. Choose **File** → **Terrain** → **Export Props to Shapefile**. The Save Props as Shapefile dialog box opens.
2. Type a name for the shapefile.
3. Click Save.

57.6.10. Importing Props and SpeedTrees from Shapefiles

If you have exported props or SpeedTrees to a shapefile, you can import those shapefiles into VR-Forces.



You can only import shapefiles that were created using the VR-Forces Export Props to Shapefile process.

To import props or SpeedTrees from a shapefile:

1. Choose **File** → **Terrain** → **Import Props From Shapefile**. The Import Props From Shapefile dialog box opens.
2. Select the shapefile you want to import.
3. Click Open.

57.7. Specifying Simulation Ocean Settings

VR-Forces can determine the depth of the ocean floor from:

- Underwater polygons, if present.
- Feature data, such as S-57 data.
- As a fixed depth.

If a terrain has underwater polygons, VR-Forces always automatically uses that data. You can enable and disable use of feature data that sets the ocean depth. If you have both types of data, VR-Forces uses the shallowest depth of the two types of data for any given point.

You can also specify that VR-Forces use a fixed depth for any area that does not have underwater polygons or feature data.



Changing the simulation ocean setting is not a dynamic change. After changing the settings, you must save the terrain to an MTF file and reload the scenario.

To specify ocean depth:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Simulation Ocean Settings page (Figure 57-22).

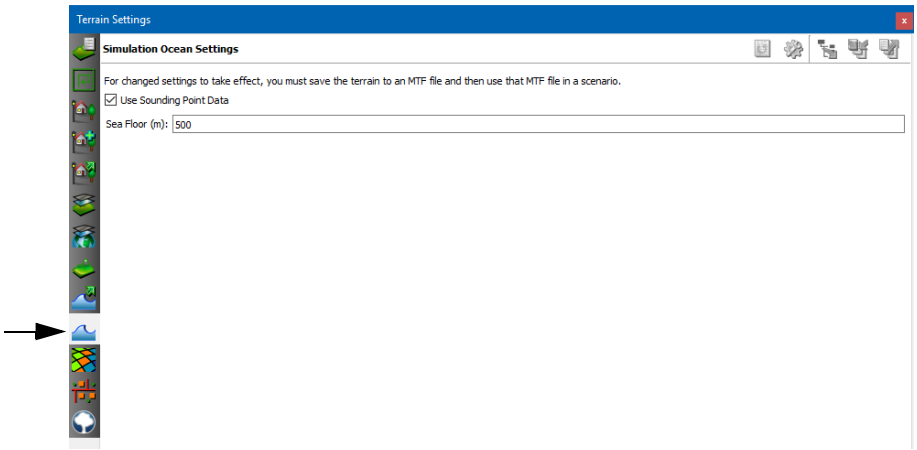


Figure 57-22. Simulation Ocean Settings page

3. To enable or disable use of feature data, select or clear the Use Sounding Point Data check box.
4. To specify the ocean depth for areas that do not have underwater polygons or feature data, type a value in the Sea Floor text box.
5. Save the terrain.
6. Reload the scenario.



58. Paged and Streaming Terrains

This chapter explains how to use terrain servers for paging and streaming terrains. It also provides information about preparing paged terrains for use with VR-Forces.

Connecting to Terrain Servers 58-2
Adding Terrain Server Connections 58-4
Connecting to Terrain Servers through a Proxy 58-5
Loading MetaFlight Terrains 58-5
Preprocessing Paged Terrains 58-6
Manually Paging-In Terrain 58-6

58.1. Connecting to Terrain Servers

VR-Forces supports the osgEarth libraries for loading terrains from remote or local data sources (Figure 58-1). Terrain servers are configured in earth (*.earth*) files. They support streaming of terrain data and are, therefore, ideal for displaying terrains that have large amounts of data.



Figure 58-1. Geocentric terrain

You can manipulate streamed terrains with the keyboard and mouse and can save them as an MTF terrain for quick loading.

VR-Forces includes several terrain server configurations for VR-TheWorld Online, a server hosted by VT MAK. You must have an internet connection to connect to these servers. It also includes server configurations that stream terrain files included with VR-Forces. These servers, such as, *Local World.earth* and *SanLuisObispo-Topo.earth*, are examples of how you can use earth files to load local data.

! Integration of osgEarth into VR-Forces allows you to connect to data sources. However, it does not provide a license to use any data sources. It is your responsibility to comply with the licensing provisions of any site to which you connect.

"Creating an Earth File to Load Terrain Data," on page 66-2, is a tutorial that shows how to create a terrain server connection that serves local data.



When you view a geocentric terrain and move the observer more than 6 million meters or so from the earth's surface, the view is forced to focus toward the center of the earth. The practical effect of this mode is that the left and right movement keys orbit the observer around the earth, rather than moving the entire earth to the left or right.

To connect to a terrain server:

1. Choose **File** → **Terrain** → **New Terrain**. The current terrain closes.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Terrain Contents page (Figure 58-2).

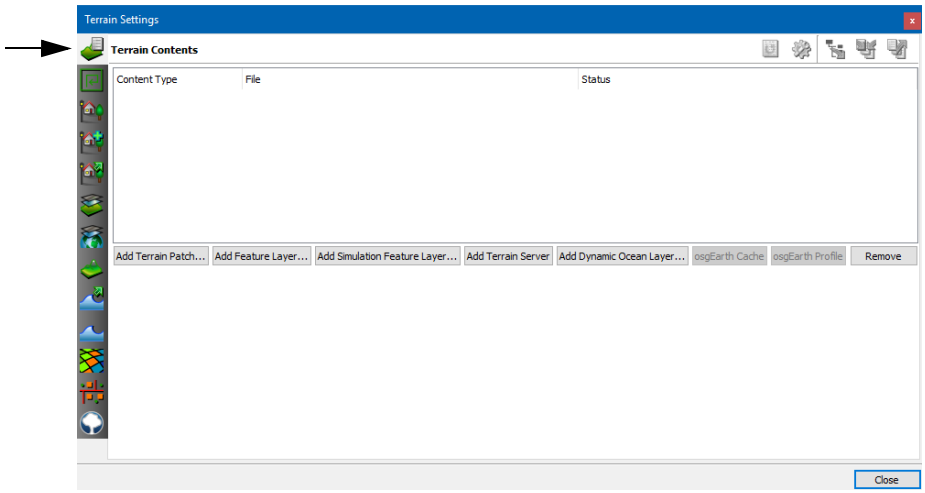


Figure 58-2. Terrain Contents

4. Click Add Terrain Server. The Add Terrain Server dialog box opens (Figure 58-3).

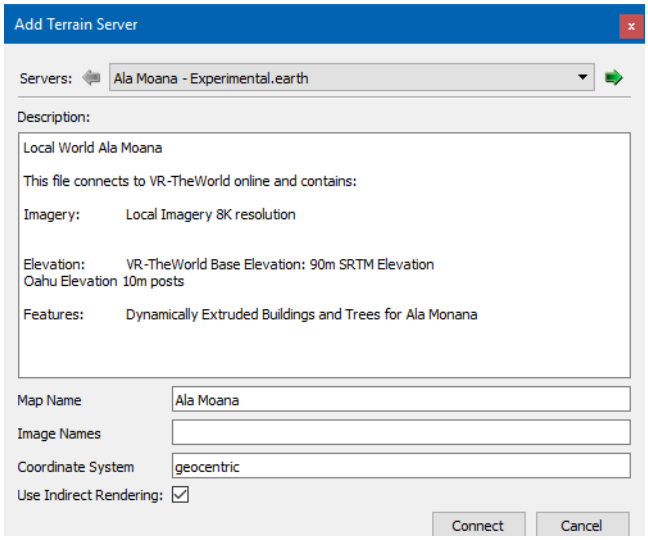


Figure 58-3. Add Terrain Server dialog box

5. Optionally, select the Use Indirect Rendering check box. If the check box is selected, the terrain patch uses the high performance indirect rendering system. If after loading a terrain there are rendering artifacts that were not present in a previous release of VR-Forces, try loading the terrain patch again with Use Indirect Rendering cleared. For more information, please see "Indirect Rendering," on page 64-14.
6. Select a server from the Servers list. The dialog box displays information about the selected terrain server. It is not editable.
7. Click Connect. If the terrain data is available (remote sources might not always be accessible), it is displayed.

You can save a terrain server as part of a terrain. For details, please see "Saving a Terrain," on page 57-3.

58.1.1. Adding Terrain Server Connections

You can add additional terrain server connections to the list provided with VR-Forces. Terrain server connections are configured in XML files according to the requirements of the osgEarth libraries. The server configurations included with VR-Forces are examples of what you can do. We recommend that you add new servers by copying an existing configuration that is similar to what you want and then editing it to point to your data.



Earth files must be placed in *./userData/servers*.

"Creating an Earth File to Load Terrain Data," on page 66-2, is a tutorial that shows how to create a terrain server that serves local data.

58.1.2. Connecting to Terrain Servers through a Proxy

If you connect to the network through a proxy, you may need to set some environmental variables to be able to connect to external terrain servers. VR-Forces has limited proxy capability. It supports basic authentication and may support domain authentication if your proxy allows it to be accessed through basic authentication. To set up authentication, set the following environmental variables:

- OSG_CURL_PROXY. Specifies the proxy machine.
- OSG_CURL_PROXYPORT. Specifies the proxy port.
- OSGEARTH_CURL_PROXYAUTH=username:password. Specifies the user name and password for the proxy.

58.2. Loading MetaFlight Terrains

VR-Forces supports MetaFlight terrain databases. To load a MetaFlight terrain, add a terrain patch, as you would for any other terrain. However, before you can load a MetaFlight file, you must process it using the mftTool and medfTool. For details, please see Chapter 63, *Processing MetaFlight Files*.

When you load a MetaFlight file, VR-Forces is actually loading many individual OpenFlight files. If file caching is enabled, VR-Forces caches the OpenFlight files to disk in a quick-loading, encrypted format so that the files can load more quickly the next time you load the terrain. The cached files are handled the same way that any other cached files are handled. For details, please see "Configuring File Caching," on page 64-2.



Geometry files referenced by a MetaFlight file must be in OpenFlight format, and not in Vega Prime's binary format (*.vsb*). MAK's terrain libraries do not support *.vsb* files.

58.3. Preprocessing Paged Terrains

VR-Forces supports preprocessing external references used by paged terrains. VR-Forces does not require terrain formats such as TerraPage or pageable IVE to be preprocessed. However these terrains may use external references in formats such as OpenFlight, which load slowly. Although VR-Forces can accelerate these files at run-time using the accelerated cache feature, the first time you load one of these terrains you can experience significant performance issues. You can use the medfTool to preprocess these external references and save them in MEDF format. Thereafter, VR-Forces will load the accelerated version at all times.

For example, if a TerraPage terrain references */externals/tree.flt*, preprocessing creates a file called */externals/tree.medf*. When VR-Forces loads the terrain, it loads the MEDF version instead of the OpenFlight version.

We recommend that you preprocess all data to minimize load times and maximize run-time performance. For details about the medfTool, please see "Compressing Model Files," on page 90-35.

58.4. Manually Paging-In Terrain

VR-Forces pages in terrain based on the location of simulation objects and an algorithm that predicts where terrain will be needed. In most cases VR-Forces can predict the need for terrain and page it in without affecting the performance of the exercise. However there may be cases where it cannot, for example if there are many simulation objects over a wide area or many fast-moving simulation objects. Therefore, VR-Forces also lets you specify areas where you want the terrain to be paged in.

Terrain page-in areas are created just like any other tactical graphic area. You can create them as the scenario progresses or you can include them in plans so that the terrain gets paged in when you know the scenario will need it. You can also attach a terrain page-in area to a simulation object to ensure that terrain in front of the simulation object gets paged in.

Terrain page-in areas are supported for paged IVE files and supported versions of the MetaFlight format.



Terrain page-in areas only affect the back-end.

To create a terrain page-in area:

1. Open the Tactical Graphics Palette.
2. In the Tags list, select Control Objects.
3. In the Force list, select the force you want this area to be assigned to.
4. In the list of control objects, select Terrain Page-in Area.
5. Create the area as described in "Creating Objects," on page 14-3.



59. Loading Data Using Earth Files

This chapter provides an introduction to the syntax of Earth files.

Earth Files 59-3
A Simple Earth File 59-3
Including External Files into Earth Files 59-6
Supported Drivers 59-7
Loading Multiple DTED Files 59-7
Processing Streamed Data 59-8
Adding Feature Layers to an Earth File 59-9
Configuring Point Features 59-11
Configuring Linear Features 59-12
Configuring Areal Features 59-13
Extruded Buildings 59-14
Using Cut-in Sites for High Resolution Insets 59-15
Using the Boundary Generation Tool 59-16
Earth File Options 59-17
Referencing an Earth Gravitational Model (EGM) 59-18
Differential Processing of Elements and Attributes 59-18
Differential Processing of model Elements 59-19
Indexing Elevation Layers 59-20
Configuring Elevation Layers for Indexing 59-20
Generating the Elevation Layer Indexes 59-21
Streaming Buoys and Beacons 59-22
Specifying Model Definitions for Streamed Buoys 59-23
Modeling Buoys Using Texture Atlases 59-26
Configuring Streaming Beacons 59-27
Configuring Streamed Navigation Lights 59-28
Configuring Seasonal Buoys and Beacons 59-29

Streaming Daymarks 59-29
Streaming Topmarks 59-30
Generating Signal Sequences 59-30

59.1. Earth Files

An earth file is an XML format configuration file used to load terrain databases using osgEarth. The file format is documented at http://docs.osgearth.org/en/latest/. The information in this chapter supplements the information on the osgEarth web site. In particular, it explains how to configure streaming features, for which there is no GUI in VR-Forces.



VR-Forces only supports geocentric terrains in earth files.

59.2. A Simple Earth File

The following is an example of a simple earth file. It has one image layer and one elevation layer:

```
<map name="Little Pond" type="geocentric" version="2">

  {% include options.terrain-rex-default.xml %}

  <elevation name="World Elevation" driver="gdal">
    <cache_policy usage="none"/>

    <url>../../data/Terrain/World/Elevation/earthnew/w001001.adf</url>
  </elevation >

  <image name="World Imagery" driver="gdal">
    <url>../../data/Terrain/World/Imagery/World4km.tif</url>
  </image>

  <model name="Little Pond Buildings" driver="feature_geom">
    <feature_indexing enabled="false"/> <!-- include this with every
    model section for VR-Vantage. If enabled feature_indexing causes
    performance issues and crashes on Linux -->
    <features name="buildings" driver="ogr">
      <url>../../data/Terrain/LittlePond/Features/BuildingsP.shp</url>
    </features>
    <layout>
      <tile_size_factor>3</tile_size_factor>
    </layout>
    <styles>
      <style type="text/css">
        HousesVictorianMansard {
        model:
"../../data/Structures/HousesVictorianMansardClapboardGray.medf";
        model-heading:       0-[rotation];
        model-scale:        1;
        model-placement: vertex;
        altitude-clamping:   terrain;
        altitude-resolution: 0.00008583068847656250;
        altitude-offset:     0;
        }
        HousesColonialGarrison {
         model:
"../../data/Structures/HousesColonialGarrisonClapboardGreen.medf";
        model-heading:       0-[rotation];
```

```
            model-scale:        1;
            model-placement: vertex;
            altitude-clamping:   terrain;
            altitude-resolution: 0.00008583068847656250;
            altitude-offset:     0;
            }
          </style>
          <selector class="HousesVictorianMansard">
            <query>
              <expr>TYPE = 'HousesVictorianMansard'</expr>
            </query>
          </selector>
          <selector class="HousesColonialGarrison">
            <query>
              <expr>TYPE = 'HousesColonialGarrison'</expr>
            </query>
          </selector>
        </styles>
        <clustering>false</clustering>
        <lighting>true</lighting>
      </model>

  </map>
```

Table 59-1 describes the XML elements in the file and their attributes.

Table 59-1: Earth file elements and attributes

Element	Attribute	Description
<!-- -->		Comment block.
map		Root level element. All other elements must be in the map block.
	name	The name of the map.
	type	The projection. VR-Forces only supports geocentric terrains, which can be specified as `geocentric` or `globe`.
	version	The VR-TheWorld Server version.
image		Specifies an image layer.
	name	The name of the layer.
	driver	The driver to use to fetch tiles, for example, TMS, WMS, or GDAL.
	url	The location of the files. This could be a web address or a local path.
elevation		Specifies an elevation layer.
	name	The name of the layer.
	driver	The driver to use to fetch tiles, for example, TMS, WMS, or GDAL.
	url	The location of the files. This could be a web address or a local path.

Table 59-1: Earth file elements and attributes

Element	Attribute	Description
model		Base element for specifying a feature layer. For more information, please see "Adding Feature Layers to an Earth File," on page 59-9.
	name	Name of the model element. This can be any text.
	driver	Driver to use for creating models.
options		Specifies options that affect the entire map. For more information, please see "Earth File Options," on page 59-17.

59.2.1. Including External Files into Earth Files

Earth files can include external files. This can make management of complex terrains easier. The included files must be XML files. There are two ways to include external files, the `xi:include` element and use of earth file templates.

The following example shows how to use `xi:include`:

```
<map name="readymap.org" type="geocentric">
  <xi:include href="readymap_imagery.xml"/>
  <xi:include href="readymap_elevation.xml"/>
</map>
```

When you reference a document using `xi:include`, it can only contain a single XML node, for example:

```
<image>
  ...
</image>
```

Since using an `xi:include` element for each node that you want to import might require many such elements, you can include multiple nodes in a single file by using earth file templates, an alternative include format. Use the syntax `{% include` *`filename`* `%}`. This syntax is used in many of the earth files shipped with VR-Forces. The following lines are from *MAK Earth (online).earth*:

```
<!-- Boston area models -->
  {% include vrtheworld-streaming.buildings-boston.xml %}

<!-- Hawaii area models -->
  {% include localdata.hawaii-maritime-navigation.xml %}
  {% include vrtheworld-streaming.buildings-oahu.xml %}
  {% include vrtheworld-streaming.airport-hnl.xml %}
```

The included XML files are located in the same directory.

! In an earth file, if you comment out an include line that uses the `{% include` *`filename`* `%}` syntax, the included file must not have any comments or it will not be processed properly. This can result in terrain errors that are not easily diagnosed. To avoid this problem, if you want the earth file to ignore a line, instead of commenting it out, change `include` to `exclude`.

Files included with the `xi:include` syntax do not have this restriction.

59.3. Supported Drivers

Drivers are software components that allow you to load one or more types of terrain data. MAK has not tested all of the drivers supported by osgEarth. VR-Forces supports the following drivers:

- tms
- gdal
- composite
- feature
- ogr
- feature_geom
- feature_custom
- wfs
- aglite

Other drivers supported by osgEarth may work. If you have questions about driver support, please contact support@mak.com.

59.4. Loading Multiple DTED Files

You can load multiple DTED files from an earth file. To do so, in the `elevation` or `heightfield` element, specify a directory instead of a specific file, for example (from *SanLuisObispo-Topo.earth*):

```
<heightfield name="SanLuisObispo Elevation" driver = "gdal">
  <url>../../data/Terrain/SanLuisObispo/Elevation</url>
  <extensions>dt0;dt1;dt2</extensions>
  ...
</heightfield>
```

The `extensions` element lets you specify which files to load based on their file extensions. If you do not include an extensions list, the GDAL driver tries to load every file in the directory.

59.5. Processing Streamed Data

VR-Forces has several software modules, which we generically call "processors," that search through streamed-in data looking for things like SpeedTree trees, objects that need effects textures attached, buoys, beacons, and so on. In order to do this efficiently, processors register feature names that they want to process. The registration data is in *./importConfig/registrant_feature_list.csv*.

For example, in *registrant_feature_list.csv*, it states that the processor for SpeedTree trees, speedTreeManager, is mapped to `Vegetation`. In *MAK Earth (online).earth*, SpeedTree trees are configured as follows:

```
<model name="trees" driver="feature_custom">
  <features name="Vegetation" driver="wfs">
```

The feature name is Vegetation. This means that it will be processed by the speedTreeManager.

Similarly, the featureDataBroker, which is responsible for creating streamed buoys, beacons, and the lights and marks that might be attached to them, is mapped to `s57-data` and `dynamicFeatures`. (Processors can be mapped to more than one feature type.) Examples of using this processor are:

```
<model name="LateralBuoysUS5HA52M" driver="feature_custom"
  enabled="true">
  <features name="s57-data" driver="ogr">
```

and

```
<model name="TaxiwayLights" driver="feature_custom">
  <features name="dynamicFeatures" driver="ogr">
```

Lastly, the indirectGeometryManager, which is responsible for adding models that replace point features for the indirect rendering system, is mapped to buildings and other static streamed geometry. This allows those models to be rendered using the high performance indirect rendering system. An example of using this processor for a model section is:

```
<model name="LargeBuildings" driver="feature_geom">
  <features name="buildings" driver="ogr">
```

registrant_feature_list.csv file is fully commented and provides additional information about this scheme.

59.6. Adding Feature Layers to an Earth File

You can add the following types of features to a terrain:

- Point features, such as trees (point trees), buoys, and buildings.
- Linear features, such as roads and waterways.
- Areal features, such as forests (areal trees), bodies of water, and extruded buildings.

You can add the XML for feature layers to an earth file.

Feature layers are added to an earth file using the `model` element. The following example shows the hierarchical relationships of a `model` element and its sub-elements. Table 59-2 describes the `model` element, its attributes, and some of its sub-elements. (A particular `model` element may not have all of the sub-elements described.)

```
<model>
  <features>
    <url></url>
  </features>
  <layout>
    <tile_size_factor></tile_size_factor>
    <level></level>
  </layout>
  <styles>
    <library>
      <url></url>
    </library>
    <style></style>
    <selector></selector>
  </styles>
  <lighting></lighting>
</model>
```

Table 59-2: `model` element attributes

Element	Attribute	Description
model		Base element for specifying a feature layer.
	name	Name of the model element. This can be any text.
	driver	Driver to use for creating models.
	vrfsim:enabled	Optional. For use in terrains used by VR-Forces if the simulation engine needs a different value for an attribute than that used by the front-end. If present, it is ignored by VR-Vantage and the VR-Forces front-end. For details, please see "Differential Processing of model Elements," on page 59-19.
The `features` and `layout` elements specify how to get and load the feature data. The `features` block is mandatory. `layout` is optional.		
features		Specifies source of data.

Table 59-2: `model` element attributes

Element	Attribute	Description
	name	Name of the element.
	driver	Driver to use to get the data.
url		Location of the feature data.
layout		Specifies how to load the data.
tile_size_factor		The data specified in the features element gets cut into tiles. This attribute specifies how aggressively to tile it.
level	name	The level specifies the level of detail (LOD) beyond which not to display the features.
	max_range	Distance beyond which the feature is not displayed.
selector	class	Specifies the type of feature.
styles		The `styles` element and sub-elements specify how to render the scene. This element is mandatory.
library		A texture library.
	name	Name of the texture library.
	url	The location of the library.
style		A specific style. The name, which precedes the curly brace ({) can be any text you want. For information about the attributes that you can specify in a style, please see http://docs.osgearth.org/en/-latest/references/symbology.html.
lighting		Specifies whether or not to enable lighting. Usually true.

You can add buildings to a terrain using the `model` element or the `buildings` element. They are functionally similar, but the `buildings` element has additional attributes that are specific to buildings. For an example of the `buildings` element, please see *./userData/servers/buildings.world-wide.osm.online.xml*. (The VR-Forces simulation engine does not process some types of buildings when they use this element. If you want to use it with VR-Forces, please contact support@mak.com for assistance.)



When you load a file using the OGR driver (specified in the `feature` element), the layer name is the name of the file (minus the extension). Other files (like S-57) can have multiple layers, and the names are decided by a format-specific algorithm. For example, an S-57 file has an enumerated list of layer names.

59.6.1. Configuring Point Features

Point features, such as trees, use the `model` element. The following code, from *MAK Earth (online).earth*, adds point trees:

```
<model name="Alamoana point trees" driver="feature_custom">
  <!-- streaming point trees -->

  <features name="Vegetation" driver="ogr">


  <url>../../data/Terrain/Hawaii/Features/AlaMoanaVegetationP.shp</url>
    <typename>5</typename>
  </features>
  <layout>
    <tile_size_factor>3</tile_size_factor>
  </layout>

  <styles>
    <style type="text/css">
        Vegetation {
        marker:              "Trees" + [veg] + "ST" ;
        marker-scale:        1;
        marker-placement: vertex;
        altitude-clamping:   terrain;
        altitude-resolution: 0.00008583068847656250;
        altitude-offset:     0;
        }

    </style>
  </styles>
  <clustering>True</clustering>
    <lighting>true</lighting>
</model>
```

59.6.2. Configuring Linear Features

Linear features, such as roads, rivers, or railways, use the `model` element. The following code, from *MAK Earth (online).earth*, configures roads:

```
<model name="Oahu_Roads" driver="feature_geom" overlay="true"
  enabled="false">

  <features name="Roads" driver="wfs">
    <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
    <typename>55</typename>
  </features>

  <!-- Sets the name property on each geometry to the value of the "name"
  attribute -->
  <feature_name>[name]</feature_name>

    <!-- Appearance details -->
    <style type="text/css">
      Oahu_Roads {
      stroke: #ff0000;
      stroke-width: 10.0;
      altitude-offset: 10;
      }
    </style>
</model>
```

59.6.3. Configuring Areal Features

Areal features, such as forests, use the `model` element. The following code, from *MAK Earth (online).earth*, configures a forest with three types of trees:

```
<!-- Forest -->
<model name="trees" driver="feature_custom">
  <features name="Vegetation" driver="wfs">
    <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
    <typename>7</typename>
  </features>
  <layout>
    <tile_size_factor>3</tile_size_factor>
    <crop_features>true</crop_features>
    <level max_range="8000" class="trees-1"> </level>
    <level max_range="2000" class="trees-2"> </level>
    <level max_range="1000" class="trees-3"> </level>
  </layout>
  <styles>
    <style type="text/css">
      trees-1 {
      marker:              "TreesNorwaySpruceST";
      marker-placement:    random;
      marker-density:      800;
      marker-scale:        1;
      altitude-clamping:   terrain;
      altitude-resolution: 0.0001;
      marker-random-seed:  1;
      }
      trees-2 {
      marker:              "TreesColoradoBlueSpruceST";
      marker-placement:    random;
      marker-density:      800;
      marker-scale:        1;
      altitude-clamping:   terrain;
      altitude-resolution: 0.0001;
      marker-random-seed:  2;
      }
      trees-3 {
      marker:              "TreesAmericanElmST";
      marker-placement:    random;
      marker-density:      800;
      marker-scale:        1;
      altitude-clamping:   terrain;
      altitude-resolution: 0.0001;
      marker-random-seed:  3;
      }</style>
    </styles>
    <clustering>true</clustering>
    <lighting>true</lighting>
</model>
```

59.7. Extruded Buildings

If you are simulating in an urban environment, you want your terrain to have realistic 3D buildings. However, modeling hundreds or thousands of buildings by hand would be an arduous and expensive task. osgEarth can extrude buildings from areal features and apply textures to them to create a realistic environment with relatively little work. Publicly available GIS data that has footprint and height data can provide the input.

The following code (from *MAK Earth (online).earth*) adds extruded buildings to the terrain:

```
<!-- Extruded buildings Oahu -->
<model name="buildings" driver="feature_geom">

  <!-- Streamed buildings -->
  <features name="buildings" driver="wfs">
    <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
    <typename>65</typename>
  </features>

  <layout>
    <tile_size_factor>3</tile_size_factor>
  </layout>

  <styles>
    <library name="us_resources">
      <url>http://vr-theworld.com/vr-
theworld/filemanager/download/public/Library/catalog.xml</url>
    </library>

    <style type="text/css">
      buildings {
        extrusion-height:      [hgt2d];
        extrusion-flatten:     true;
        extrusion-wall-style:  building-wall;
        extrusion-roof-style:  building-rooftop;
        extrusion-random-seed: 1;
        altitude-clamping:     terrain;
        altitude-resolution: 0.0001;
      }
      building-wall {
        skin-library: us_resources;
        skin-tags:    building;
        skin-random-seed: 1;
        fill:         #ffffff;
      }
      building-rooftop {
        skin-library: us_resources;
        skin-tags:    rooftop;
        skin-tiled:   true;
        skin-random-seed: 1;
        fill:         #ffffff;
      }
    </style>
  </styles>
```

```
    <lighting>true</lighting>
    <cluster_culling>false</cluster_culling>

  </model>
```

For descriptions of the elements in a `model` block, please see Table 59-2.

59.8. Using Cut-in Sites for High Resolution Insets

Users of terrain servers, such as VR-TheWorld Server, often want to add high resolution sites of particular interest while using the lower resolution terrain provided by the terrain server for the rest of the world. You can add a high resolution terrain patch to a terrain server, but it overlays the terrain provided by the server and this can cause unexpected visual problems. Ideally, you should cut out the relevant areas of terrain provided by the server and add your high resolution terrain to the cut out area. For example, the VR-Forces terrain *MAK Earth (online).mtf* has two terrain patches, *MAK Earth (online).earth*, which provides the base terrain from VR-TheWorld Server, and *./data/Terrain/Hawaii/Geometry/OpenFlightAlaMoana/AlaMoanaModified-GeocentricProxy.osg*, which is a locally stored high resolution inset.

The `mask` element lets you cover the base terrain data with your hand-modeled data. A mask element in an earth file looks similar to the following:

```
<mask name="HonoluluRunway26R-8L-mask" driver="feature" >
  <features driver="ogr">
    <geometry>
      POLYGON((-157.91489143 21.32914026 3.016, -157.91525638
  21.32914026 3.016, -157.91666238 21.32912541 3.016, -157.91697822
  21.32934847 3.016, -157.91709263 21.32968744 3.016, -157.91709263
  21.32968744 7.016, -157.91709263 21.32968744 11.016, -157.91709263
  21.32968744 15.016, -157.91710262 21.32968706 15.016, -157.91697822
  21.32934847 16.016, -157.91709263 21.32968744 16.016, -157.91710262
  21.32968706 15.016))
    </geometry>
  </features>
</mask>
```

VR-Forces includes a tool, the osgEarth Boundary Generation Tool, that can calculate the geometry of an OpenFlight file. To cut out a section of terrain from the terrain server, add a `mask` element to the earth file. Use the output from the boundary tool for the data in the geometry element.

Table 59-3 describes the `mask` element and its sub-elements.

Table 59-3: `mask` element

Element	Attribute	Description
mask		Element for cutting in geometry.
	name	Name of the mask.
	driver	The driver to use.
features		List of features in the mask.

Table 59-3: `mask` element

Element	Attribute	Description
	driver	The driver to use. OGR reads raw vector feature data from shapefiles and other formats.
geometry		Specifies the 3D boundaries of the site. You must know this information.
	geometry_url	The location of the file with the feature data.

59.8.1. Using the Boundary Generation Tool

The Boundary Generation Tool takes an OpenFlight file and calculates the bounding geometry of the file. It returns the coordinates in a text file. The general process of cutting in a site is as follows:

1. Run the Boundary Generation Tool against an OpenFlight file.
2. Insert the output of the boundary generation tool in a `mask` element in your earth file. (For details about `mask` elements, please see "Using Cut-in Sites for High Resolution Insets," on page 59-15.)
3. Load the terrain.
4. Add the FLT file as a terrain patch.
5. Save the terrain.

Running the Boundary Generation Tool

The output from the Boundary Generation Tool tool, in *boundary.txt* or the output file you specify, looks similar to the following:

```
POLYGON((-157.91489143 21.32914026 3.016, -157.91525638 21.32914026
  3.016, -157.91666238 21.32912541 3.016, -157.91697822 21.32934847
  3.016, -157.91709263 21.32968744 3.016, -157.91709263 21.32968744
  7.016, -157.91709263 21.32968744 11.016, -157.91709263 21.32968744
  15.016, -157.91710262 21.32968706 15.016, -157.91697822 21.32934847
  16.016, -157.91709263 21.32968744 16.016, -157.91710262 21.32968706
  15.016))
```

The first two vertices are the latitude and longitude. The third vertex is the height, in meters. The remainder of the vertices map the outline of the geometry.

To run the boundary generation tool:

1. Open a console window.
2. Change directory to *./bin64*.

3. Run the tool:

```
osgearth_boundarygen [--out file_name --no-geocentric
  --convex_hull --verbose --view --precision precision
  --tolerance tolerance] model_file
```

Table 59-4 describes the options.

Table 59-4: osgEarth Boundary Generation Tool command-line options

Option	Description
`--convex_hull`	Calculates a convex hull instead of a full boundary.
`model_file`	The input model file.
`--no-geocentric`	Skips geocentric reprojection. Use for flat databases.
`--out` *`file_name`*	Specifies the output file for boundary geometry. Default: *boundary.txt*.
`--precision` *`precision`*	Specifies the number of significant digits in the output coordinates. Default: 12.
`--tolerance` *`tolerance`*	Specifies the tolerance for combining similar vertices along a boundary. Default: 0.005.
`--verbose`	Prints progress messages to the console.
`--view`	Shows the result in a 3D window.

59.9. Earth File Options

An earth file can have `option` elements that affect the entire map. Table 59-5 describes the option elements.

Table 59-5: `option` element

Element	Attribute	Description
lighting		Enables or disables lighting for the entire map. This is usually true.
elevation_interpolation		Specifies how OSG interprets elevation between grid posts. If you are using VR-Forces or visualizing a VR-Forces simulation, set this to triangulate to ensure correlation.
cache		Specifies caching for terrain data.
	type	The type of caching to use.
	path	The location of the data cache.

59.10. Referencing an Earth Gravitational Model (EGM)

Earth files can support loading elevation data that references an earth gravitational model (EGM). To do so, add the following line to the `elevation` or `heightfield` element:

```
<vdatum>egm_model</vdatum>
```

where *egm_model* can be egm96, egm84, or egm2008. For example:

```
<elevation driver="gdal">
  <url>file.dt0</url>
  <vdatum>egm96</vdatum>
</elevation>
```

59.11. Differential Processing of Elements and Attributes

There are times when you want the front-end and back-end to process terrain differently. You can specify different values for elements and attributes with the `vrfsim:` syntax. Anyplace that VR-Forces sees `vrfsim:` when it processes an earth file, it uses that value for the back-end. For example, if you want different values for the `name` attribute in a model element, specify it as follows:

```
<model name="front_end_name" vrfsim:name="back_end_name">
```

To specify differential elements, the element names must be the same and be the only two elements of that name in the scope of the earth file or an element block. For example, the following code would mean use the `<trees>` element in the front-end and the `<vrfsim:trees>` element in the back-end:

```
<trees>
  ...
</trees>

<vrfsim:trees>
  ...
</vrfsim:trees>
```

59.11.1. Differential Processing of model Elements

You can enable or disable differential processing within `model` elements. By default, all `model` elements except those that use the `simple` driver have differential processing disabled.

To enable differential processing add `vrfsim:enabled="true"` to the `model` element.

The following example of differential feature processing is from *MAK Earth (online).earth*. It is the section used to extrude buildings. `vrfsim:enabled` is set to true. The value for the `typename` attribute is applied to front-end processing and the value for the `vrfsim:typename` attribute is applied to back-end processing.

```
<model name="buildings" driver="feature_geom" vrfsim:enabled="true">
  <features name="buildings" driver="wfs">
    <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
    <typename>105</typename>
    <vrfsim:typename>109</vrfsim:typename>
  </features>
```

If a model element uses the `simple` driver, it uses the value of the `enabled` attribute to determine whether or not to use differential processing (in other words, there is no default setting). If you want to explicitly enable or disable differential processing, add `vrfsim:enabled` with the appropriate value to the `model` element. Then add an attribute element in which the attribute name is prefaced by `vrfsim:` as in the example in this section.

59.12. Indexing Elevation Layers

Calculating terrain intersections over large distances (hundreds of miles), such as in air-to-air and space-to-ground scenarios can result in loading many terrain tiles, which may reduce performance to unacceptable levels. You can reduce the likelihood of this problem by indexing elevation layers and configuring earth files to use the indexes.

When you index elevation layers, the indexes allow the back-end to know the minimum and maximum altitudes of the elevation in each tile so that terrain intersects that are above or below the tile do not cause the tile to load (since they would not intersect with it anyway). The end result is that only the portions of intersections that are near the terrain cause terrain to load and the rest of the intersection query can be efficiently ignored.

59.12.1. Configuring Elevation Layers for Indexing

To configure an index for an elevation layer, you add a `<mak_terrain_index>` element to the `<elevation>` element in the earth file. The following example shows an elevation layer configured for indexing:

```
<elevation
  name="Worldwide 90m Elevation with Bathymetry Collection"
  driver="tms">
  <cache_policy usage="none"/>
  <url>http://vr-theworld.com/vr-theworld/tiles/1.0.0/122/</url>
  <mak_terrain_index name="worldwide90m_bathymetry">
    <url>path/to/worldwide90m_bathymetry.index.bin</url>
    <max_lod>7</max_lod>
    <max_tolerance>500.0</max_tolerance>
    <min_tolerance>-500.0</min_tolerance>
  </mak_terrain_index>
</elevation>
```

If the layer does not require any non-default configuration you do not need to add any configuration to the earth file.

Indexing is done for each leaf elevation layer. If you have a composite elevation layer you must configure an index for each individual elevation layer that makes up the composite, not for the composite as a whole.

If the layer is local (that is, the URL is a local file, not a streaming URL) then the name of the index file will be the same as the file name in the URL, with the extension *.index.bin* added to the end. It is created in the same directory as the local file.

If the layer is not local, then a file name is generated based on the URL and it is put in *./userData/terrainIndices*.



The index files provided with MAK terrains are in the appropriate data directory, not in *./userData/terrainIndices*.

An elevation layer can have multiple indexes. For example, one index may be a high resolution index around a specific site, while another one may be a lower resolution index that covers the whole terrain. To configure multiple indexes for a layer, you add multiple `<mak_terrain_index>` elements.

The attributes of a `<mak_terrain_index>` element are as follows:

- `url`. The location to put the index file if you do not want to put it in the default location. The file must end with *.index.bin* or *.index.txt*. (Text format is human readable, but takes longer to load.)
- `name`. Label for the index layer. This is used so that you can specify specific layers to generate with the terrainIndexTool.
- `max_lod`. The maximum LOD to put in the index. This allows you to generate a lower resolution index for a terrain that is too large to generate the full index, or for which the generated index would take too long to load.
- `max_tolerance`. A value to add to the max elevation value when reading in this index. If an index is generated with a `max_lod` that is less than the actual maximum LOD of the terrain, then the elevation values in the index will be based off of lower resolution elevation data than what will actually be used in the simulation. This means they will not necessarily correlate. This parameter lets allows you to adjust these values.
- `min_tolerance`. Similar to `max_tolerance`, this value is added to the minimum elevation value when reading the index. It should be negative. The minimum altitudes are not particularly important since they really only come into play when something is below the terrain.

There is no good way to compute good values for `max_lod` and the tolerances. They are generally arrived at by trial and error testing. `max_lod` should usually be set as large as it can and still be able to generate and load an index in a reasonable amount of time. The tolerances should be set as small as possible. Setting them too small can result in terrain not being loaded when it should.

59.12.2. Generating the Elevation Layer Indexes

You generate indexes for elevation layers with the terrainIndexTool. The tool reads the earth file specified in a terrain. It generates indexes for the elevation layers in the earth file. The syntax is as follows:

```
terrainIndexTool --terrain terrain [--name name]
```

where:

- `--terrain` specifies the name of the terrain file (*.mtf*) to process.
- *`--name`* specifies a specific elevation layer to process. You can use *`--name`* more than one time on a command line. If you omit this parameter, all layers in the terrain file are processed.

59.13. Streaming Buoys and Beacons

VR-Forces can display models of buoys and beacons specified in shapefiles and S-57 data, which is streamed in using earth files.

Streamed features (including buoys) are organized into layers. In the earth file, the names in the **styles** block specify the layer the streamed features will belong to. By default, VR-Forces defines seven layers for buoys. They correspond to the different buoy layers in an S-57 file. These layers are named LateralBuoys, SafeWaterBuoys, ISDBuoys, RegulatoryBuoys, InstallationBuoys, Cardinal-Buoys, and SpecialPurposeBuoys.

The configuration file *./appData/importConfig/s57_layer_list.csv* specifies which style names in an earth file correspond to recognized streamed feature types. The file has two lines. The first starts with the keyword "Buoys" and is followed by a comma-separated list of all the style names that will be recognized (and streamed in) as buoys. The second line starts with the keyword "Beacons" and is followed by the list of recognized beacon styles. They can be anything you like, but need to correspond with whatever style names you use in the earth file.

The following is an example of a `style` block for a buoy (Special Purpose Buoys):

```
<model name="buoys" driver="feature_custom">
  <features name="Special Purpose Buoys" driver="ogr">
   ...
  </features>
  <layout>
   ...
  </layout>
  <styles>
    <style type="text/css">
        <!-- | | | This is the layer name -->
        <!-- v v v                        -->
      SpecialPurposeBuoys {
      ......
      }
    </style>
  </styles>
```

59.13.1. Specifying Model Definitions for Streamed Buoys

Within each style block, you specify the model definition to use for a streamed buoy or beacon. Since buoys and beacons have different styles and within each style have different colors and markings, VR-Forces can map the S-57 data to buoy and beacon models in the following ways:

- Specify a model definition for each type of buoy. Buoy color and numbering is the same for all buoys.
- Generate a model definition name using a pattern and S57 attributes.
- Specify a model definition for a unique buoy based on the buoy's name.
- Use a texture atlas to model buoys. (A texture atlas is an image that has sub-images that can be referenced by a coordinate system. Those sub-images can be applied to 3D models.)

All of these methods are supported by VR-Forces and are described in this section. However, VR-Forces uses texture atlases for all streaming buoys and beacons.

Specifying a Model Definition for a Type of Buoy

To specify the model definition for a type of buoy, you specify it as the marker name:

```
<styles>
   <style type="text/css">
    SpecialPurposeBuoys {
    marker: "BuoysSpecialPurpose" ;
    ...
```

In this case, the model definition “BuoysSpecialPurpose” is used for all buoys streamed in for that SpecialPurposeBuoys layer.

This approach is the simplest way to specify model definitions for buoys.

Generating a Model Definition Name for a Buoy

Instead of specifying a model definition, you can have VR-Forces build a model definition using the buoy’s attributes. This approach is more complex than specifying one model definition for a type of buoy, but it is more flexible. You can accommodate many variations of a given type of buoy without having to specify many model definitions.

Buoy attributes can be strings or enumerated values (such as shape (BOYSHP), color (COLOUR), and color pattern (COLPAT)).

The following code results in model definitions like “Buoys352”.

```
<styles>
  <style type="text/css">
    SpecialPurpose {
    marker: "Buoys" + [BOYSHP] + [COLOUR] + [COLPAT];
```

Creating and managing model definitions with coded numbers in this way is difficult. Therefore, VR-Forces provides a mechanism to map the numeric attribute values to strings.

The enumerated value-to-string mappings are in *./appData/importConfig/s57_attr_map.csv*. You can change them to any strings that you want to use. Given this information, you can specify your model definition as a pattern and provide the enumerated attribute values. VR-Forces uses them to decode and build the requested model definition. The pattern and attribute values are specified as a colon (:)-separated list of the following form:

```
"pattern:Attribute1=value:Attribute2=value" ...
```

The pattern itself indicates where attribute values should be replaced using '[Attribute]' substrings. For example, consider an input string such as:

```
"Buoys[COLOUR][BOYSHP]:COLOUR=5:BOYSHP=3"
```

The default mappings for COLOUR and BOYSHP in *s57_attr_map.csv* are as follows:

```
BOYSHP,,Nun,Can,Spherical,Pillar,Spar,Barrel,Super,Ice
COLOUR,,White,Black,Red,Green,Blue,Yellow,Grey,Brown,Amber,Violet,Orange,
  Magenta,Pink
COLPAT,,HorizontalStriped,VerticalStriped,DiagonalStriped,Squared,
  Striped,BorderStriped
```

Therefore, the model definition would be BuoysBlueSpherical, since the enumerated COLOUR value '5' (the fifth value in the list) is mapped to "Blue", and the BOYSHP value '3' is mapped to "Spherical". Some of these attributes, such as COLOUR, may be lists of enumerated values. The strings for enumerated lists are concatenated. The following input string:

```
"Buoys[BOYSHP][COLOUR][COLPAT]:COLOUR=3,1:BOYSHP=2:COLPAT=2
```

results in the model definition "BuoysCanRedWhiteVerticalStriped".

To generate the input string like the one above from an earth file, you would use:

```
marker: "Buoys[BOYSHP][COLOUR][COLPAT]"
            + ":COLOUR=" + [COLOUR]
            + ":BOYSHP=" + [BOYSHP]
            + ":COLPAT=" + [COLPAT]
```

The order that the actual attribute/value pairs are encoded (after the pattern) does not matter.

Designators

Many buoys have some form of one to three numbers, letters, or both that appear on the buoy itself. S-57 does not provide an attribute that indicates what this symbol is, but it is usually found at the end of the OBJNAM attribute, such as “Boston South Channel Buoy 13” or “President Roads Junction Lighted Buoy PR”. If the OBJNAM attribute is provided, the model definition builder tries to extract this symbol. It must be 3 characters or less, and be all numerics, all capital letters, or a mix of the them. You can add this to a model definition pattern using the [DESIGNATOR] substring, for example:

```
marker: "Buoys[BOYSHP][COLOUR][COLPAT][DESIGNATOR]"
+ ":COLOUR=" + [COLOUR]
+ ":BOYSHP=" + [BOYSHP]
+ ":COLPAT=" + [COLPAT]
+ ":OBJNAM=" + [OBJNAM]
```

This code, when streaming in “Boston South Channel Buoy 13”, would result in a model definition of “BuoysCanGreen13”, since the BOYSHP attribute for that buoy maps to “Can” and COLOUR attribute to “Green”.

Unique Buoys

There may be cases where you want to use a specific model definition for a particular buoy. The buoy might be a special landmark that needs to be recognizable. The file *appData/importConfig/s57_modeldef_by_name_map.csv* contains a list of buoy names (the OBJNAM attribute) and model definition pairs. If the OBJNAM attribute value is provided in the marker string and if it matches one of the buoy names in *s57_modeldef_by_name_map.csv*, then the specified model definition is used instead of a generated definition.

59.13.2. Modeling Buoys Using Texture Atlases

This approach is more complex than specifying a model definition per type of buoy or building one from attributes. It requires understanding how to use texture atlases. However, this method allows you to have a large number of buoys in a scene with a wide variety of colors, patterns, and designators using a very limited number of models. This dramatically reduces the scene complexity (thereby increasing frame rate) when you use instancing. This is the approach that VR-Forces uses for all buoys and beacons it ships with.

Each buoy shape (can, pillar, nun, and so on) is a separate model. Each model has a texture atlas that has all the different colors and patterns needed for that buoy shape. In the `marker` field (in the `model` statement) a simple model definition pattern is used:

```
Buoy[BOYSHP]
```

The other attributes are passed in the same fashion as described in "Generating a Model Definition Name for a Buoy," on page 59-23 when mapping attributes to model definition substrings:

```
marker: "Buoys[BOYSHP]"
  + ":BOYSHP=" + [BOYSHP]
  + ":COLOUR=" + [COLOUR]
  + ":COLPAT=" + [COLPAT]
  + ":OBJNAM=" + [OBJNAM] ;
```

In this example, all the attributes are mapped to strings, but the COLOUR and COLPAT strings are used to look up the appropriate uv texture coordinate offsets for the buoy. The texture-uv mapping file has the same name as the buoy model plus the extension *dtxc* (for example, *PillarBuoy.dtxc* and *Can.dtxc*).

The *.dtxc* files can support multiple atlases that may be used on the same model. VR-Forces uses multiple atlases for buoys with topmarks, and beacons with daymarks and topmarks. (For information about topmarks and daymarks, please see "Streaming Daymarks," on page 59-29 and "Streaming Topmarks," on page 59-30.) The model definition for the buoy specifies a key (found in the *.dtxc* file) to be used for the main texture (`mainTextureKey`) and for the topmark's texture (`topmarkTexturekey`).

The buoy is also built with special geometry and a texture atlas for displaying the alpha-numeric designator. The geometry includes a switch to select between sites for placing one, two, or three alphanumeric characters. The characters themselves are then mapped from a texture atlas using the designator string found and a uv-coordinate texture mapping file that is shared by all buoys (and daymarks), named *LettersNumbers.ctxc*.

59.13.3. Configuring Streaming Beacons

You can specify beacon feature styles the same way you specify buoys. You can add any number of beacon styles names. The S57 attribute enumeration-to-string mappings for beacons are in *s57_layer_list.csv*.

The following code is an example of a beacon model block:

```
<model name="beacons" driver="feature_custom" enabled="true">
  <features name="Lateral Beacons" driver="ogr">
    <!-- From S-57 data -->
    <ogr_driver>S57</ogr_driver>
    <url>../../data/Terrain/Massachusetts/Features/US5MA10M.000</url>
    <layer>BCNLAT</layer>
  </features>
  <layout>
    <tile_size_factor>3</tile_size_factor>
  </layout>
  <styles>
    <style type="text/css">
      LateralBeacons {
      marker: "Beacons[BCNSHP]"
           + ":BCNSHP=" + [BCNSHP]
           + ":COLOUR=" + [COLOUR]
           + ":COLPAT=" + [COLPAT]
           + ":PERSTA=" + [PERSTA]
           + ":PEREND=" + [PEREND]
           + ":OBJNAM=" + [OBJNAM] ;
           marker-scale:        1;
           marker-placement: vertex;
           altitude-clamping:   terrain;
           altitude-resolution: 0.00008583068847656250;
           altitude-offset:     0;
      }
    </style>
  </styles>
</model>
```

Model definitions for beacons work the same way as for buoys. They are configured in *s57_attr_map.csv*. It can contain enumeration mappings for any enumerated S-57 attribute that you want to use to create model definitions. Default values are in *s57_defaults.csv*. This contains the default name and default model definition for each style group specified in *s57_layer_list.csv*. The delivered *s57_defaults.csv* contains:

```
# If the final model definition generated does not exist, one of these
  model definitions will be used
Buoys,DefaultModelDefinition,BuoysGreenPillar1
Beacons,DefaultModelDefinition,BuoysRedPillar2

# If the feature has no name, use one of these
Buoys,DefaultName,Unnamed Buoy
Beacons,DefaultName,Unnamed Beacon
```

59.13.4. Configuring Streamed Navigation Lights

Navigation lights are managed somewhat differently from other streamed features. These lights do not result in actual objects added to the scene. The information is used to create and attach light controllers to existing objects. The lights themselves need to already exist as part of the buoy or beacon model they appear on.

Each model that has a light should have a comment-tagged switch for choosing a light group of the specified color. Each light in turn has a switch for turning the light itself on or off. We use the tag `@dis light_color` to tag a color switch, with values 0-3 corresponding to white, green, red, and yellow. The switch to turn lights off and on is `@dis navigation_lights`.

VR-Vantage streams in light data using `model` blocks similar to those used for buoys and beacons. You can apply style names. *light_layer_list.csv* controls which styles will be read. *Massachusetts S-57 (online).earth* uses a single style name for all lights, **NavigationLights**. The following example shows a `model` block for a light:

```
<model name="LightsUS5MA10M" driver="feature_custom"  enabled="true">
  <features name="s57-data" driver="ogr">
    <!-- From S-57 data -->
    <ogr_driver>S57</ogr_driver>

    <url>../../data/Terrain/Massachusetts/Features/US5MA10M.000</url>
    <layer>LIGHTS</layer>
<!--  From a Shape file -->
<url>../../data/Terrain/Massachusetts/Features/US5MA10M/LIGHTS.shp</url>
  </features>
  ...
  <styles>
    <style type="text/css">
      NavigationLights {
        marker: "COLOUR=" + [COLOUR] + ":SIGSEQ=" + [SIGSEQ] + ":EXCLIT="
  + [EXCLIT];
        marker-scale:        1;
        marker-placement: vertex;
        altitude-clamping:   terrain;
        altitude-resolution: 0.00008583068847656250;
        altitude-offset:     0;
      }
    </style>
  </styles>
</model>
```

Since we are not building model definitions, we do not need to map enumerated values to strings. However, we do map S-57 light colors to the colors that we support. This mapping is in *light_attr_map.csv*.

VR-Vantage associates light controllers with other streamed features based on the S-57 or shape file locations of the lights and features, as follows:

- If the COLOUR attribute is provided, the attached light controller sets the light color using the light color switch.
- If the SIGSEQ attribute is provided, the light is turned on and off according to the specified sequence.
- If the EXCLIT attribute is provided, the light controller enables or disables the light based on daylight and fog conditions.

The light color switch and light switch values can be changed by specifying different ones through the API, specifically using:

```
de.scene().terrain().navigationLightManager().
     setLightColorSwitchNumber()
```

and

```
de.scene().terrain().navigationLightManager().setLightSwitchNumber()
```

59.13.5. Configuring Seasonal Buoys and Beacons

Seasonal buoys and beacons are supported. Include the PERSTA/PEREND attributes in the earth file as part of the marker field, and buoys and beacons will only appear during the correct season. Example:

```
marker: "Buoys[COLOUR][COLPAT][BOYSHP][DESIGNATOR]"
  + ":BOYSHP=" + [BOYSHP]
  + ":COLOUR=" + [COLOUR]
  + ":COLPAT=" + [COLPAT]
  + ":PERSTA=" + [PERSTA]
  + ":PEREND=" + [PEREND]
  + ":OBJNAM=" + [OBJNAM] ;
```

59.13.6. Streaming Daymarks

VR-Forces supports streamed daymarks for beacons. A limited number of daymark models and textures are provided as examples. *Massachusetts S-57 (online).earth* has a `model` statement to stream daymarks from the Boston Harbor S-57 data. As with buoys and beacons, the `model` entry specifies a style name (Daymarks), which corresponds to a layer specified in *daymark_layer_list.csv*. OSGEarth models that use the feature_custom driver and have a style name that is in the layer list file, are processed as daymarks. S-57 attribute values can be mapped using the *daymark_attr_map.csv* file.

The beacon models provided with VR-Forces contain multiple daymark placard shapes (Diamond, Square, and Triangle), which are selected based on the TOPSHP attribute. The TOPSHP attribute is mapped to '0', '1', or '2', which are the switch values for the diamond, square, or triangle shaped marks. Other attribute values, including TOPSHP, COLOUR, COLPAT, and CATSPM are used to look up UV coordinates in the model's texture atlas using a uv-texture coordinate mapping file similar to the one described for buoys. The mappings are in *./data/Objects/Beacons/daymark.dtxc*. These daymark shapes, plus textures provided, represent all the daymarks attached to beacons in Boston Harbor. The model definition for daymarks includes a parameter that specifies a string key to use in the *.dtxc* file ("daymarkTextureKey").

59.13.7. Streaming Topmarks

Topmarks can be streamed in and attached to buoys and beacons similarly to daymarks. The shape is determined by the TOPSHP attribute. The attribute is mapped to a switch value using the *topmark_attr_map.csv* file. Textures for the topmarks can be selected from a texture atlas similarly to buoys and daymarks. The model definition for a buoy or beacon that needs to map attributes to uv-texture coordinates using that model's *.dtxc* file will have a string key (`topmarkTextureKey`) that identifies the mapping entries in the file that pertain to the topmark texture atlas.

59.13.8. Generating Signal Sequences

The S-57 attribute SIGSEQ (signal sequence) specifies the flashing sequence that a navigational aid light on a buoy or beacon displays. These sequences (along with color) are used to distinguish among different buoys and beacons at night. However, this is not a required S-57 attribute. If the SIGSEQ attribute is not present, VR-Vantage tries to generate a signal sequence using the LITCHR (light characteristic), SIGPER (signal period) and SIGGRP (signal group) attributes for the most commonly seen sequences.

As part of the class responsible for generating these sequences, the VR-Forces API includes a virtual function for each sequence type that is not implemented, so that a developer can implement it. The assumption is that not many of the other sequence types will be used, but it is data-specific. Once we know what sequences are needed for the S-57 data in use, it should be relatively straightforward to implement those additional sequence generation methods.

The class used to generate signal sequences is *DtLightSignalSequenceGenerator*, in the vrvCore library. *DtLightSignalSequenceGenerator.h* provides documentation, including which methods are implemented and which are not. To extend this class, a creator is provided, *DtLightSignalSequenceGeneratorCreator*, an extended instance of which can be registered through a plug-in with the display engine.



60. Configuring CDB Terrains

This chapter explains how to configure earth files to load CDB terrain databases.

Loading CDB Databases 60-2
Configuring CDB earth Files 60-2
Configuring the Elevation Dataset (001) 60-3
Configuring the Imagery Dataset (004) 60-5
Configuring the GeoSpecific (GS) Dataset (100) 60-6
Configuring the GeoTypical (GT) Dataset (101) 60-13
Configuring the Extended Imagery Dataset (900) 60-15
CDB Configuration Files 60-17
Caching CDB Data (cdbtool) 60-19

60.1. Loading CDB Databases

VR-Forces can display multiple CDB datasets (elevation, imagery, geotypical points). MAK products load the datasets using an earth file. The earth file entries point to the CDB root path of the database and specify the CDB driver and the CDB dataset codes.

Because MAK uses osgEarth to load CDB databases, you have the same flexibility with CDB databases that you do with other database types. For example, you can:

- Use streaming data (from VR-TheWorld Server), or other source imagery to build a skirt around a CDB inset.
- Place an OpenFlight inset into a CDB database.
- Add lights, buildings, forests, grass, and other clutter on top of an existing CDB database.

VR-Forces includes an example CDB terrain – *Test-CDB.mtf*. This terrain demonstrates the ability to load data in CDB format. For details about the CDB API, please see https://portal.presagis.com/support/solutions/articles/19000014917-cdb-api-

60.2. Configuring CDB earth Files

This section explains how to configure your CDB earth file to load the different datasets you need. You may need to configure the following datasets:

- Elevation
- Imagery
- Geospecific
- Geotypical
- Extended imagery

60.2.1. Configuring the Elevation Dataset (001)

To load an elevation dataset, use the `elevation` element, as in the following example:

```
<elevation name="cdb_elevation"
              driver="cdbraster"
              cdb_dataset="1"
              cdb_component_selector1="1"
              cdb_component_selector2="1"
              cdb_cache_file="25"
              cacheid="camppend1-cdb-elevation" >
  cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
</elevation>
```

It tells osgEarth to use the CDB raster driver.

Table 60-1 describes the attributes for the `elevation` element.

Table 60-1: Attributes for elevation and imagery datasets

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be `cdbraster`).
cdb_dataset	(Optional) Specifies the CDB dataset to read. For elevation always use "1". Default: 1.
cdb_component_selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 60-2. Default: 1.
cdb_component_selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cdb_cache_file	(Optional) Specifies how many CDB tiles to keep in memory while building an OSG tile. Default: 20.
cdb_max_lod	(Optional) Specifies the maximum LOD of this dataset. Set this value if you have a CDB database that is larger than 500 geocells. For example, if you have the CDB World Wide database, set the value to 0. This will restrain the CDB Driver to pre-parse all of the LOD folder. If the value is not set (default) it reads the CDB to find the maximum LOD of the elevation dataset.
cacheid	Specifies where to save the cache files, if caching is enabled.
cdb_root	Specifies the top folder of the CDB database.

Table 60-2 lists the `elevation` sub-dataset codes.

Table 60-2: `elevation` sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
1	1	1	Primary Elevation.
1	1	2	Primary Elevation Control (deprecated in CDB 3.2). (Not supported.)
1	1	3	Primary Alternate Elevation. (Not supported.)
1	2-99	1	Subordinate Elevation (deprecated in CDB 3.2). (Not supported.)
1	2-99	2	Subordinate Elevation Control (deprecated in CDB 3.2). (Not supported.)
1	100	1	Subordinate Bathymetry.
1	100	2	Subordinate Alternate Bathymetry. (Not supported.)
1	101	1	Subordinate Tide Elevation. (Not supported.)

Configuring the Elevation Sub-dataset (Bathymetry)

Adding the bathymetry allows you to add dynamic ocean in Vantage. The following example shows how to add bathymetry:

```
<elevation name="cdb_bathymetry"
             driver="cdbraster"
             cdb_dataset="1"
             cdb_component_selector1="100"
             cdb_component_selector2="1"
             cdb_cache_file="10"
             cacheid="camppend1-cdb-elevation" >
  <cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
  <offset>true</offset>
</elevation>
```

60.2.2. Configuring the Imagery Dataset (004)

The `image` element tells osgEarth to use the CDB driver to load the imagery dataset, as shown in the following example:

```
<image name="cdb_imagery"
         driver="cdbraster"
         cdb_dataset="4"
         cdb_component_selector1="1"
         cdb_component_selector2="1"
         cdb_cache_file="25"
         cacheid="camppend1-cdb-imagery" >
  <cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
</image>
```

Table 60-1 describes the attributes for the `image` element. They are the same as for the elevation attribute, except that `cdb_dataset` must be 4.

Table 60-3 lists the codes for the imagery sub-dataset.

Table 60-3: Imagery sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Compo-nent Selec-tor 2 (Txxx)	Description
4	1	1	Yearly Visual Spectrum Terrain Imagery (VSTI) representation.
4	2	1	Alternate Spring representation (depre-cated in CDB 3.2).
4	2	2	Alternate Summer representation (deprecated in CDB 3.2).
4	2	3	Alternate Fall representation (depre-cated in CDB 3.2).
4	2	4	Alternate Winter representation (depre-cated in CDB 3.2).
4	3	1-12	Alternate Monthly representation.
4	4	1-4	Quarterly VSTI representation.
4	5	1	Subordinate VSTLM. (Corresponds to the terrain light maps draped (orthographically) over the terrain skin derived from the Primary Terrain Eleva-tion Dataset. It may be single-channel monochrome or 3-channel color image.) Please see "Configuring the Imagery Lightmap Sub-dataset (004-005)," on page 60-6 for more information.

Configuring the Imagery Lightmap Sub-dataset (004-005)

The following example shows how to add a lightmap. This code is different from the primary cdb imagery dataset.

```
<lightmap>
  <image name="cdb_imagery_lightmap"
           driver="cdbraster"
           cdb_dataset="4"
           cdb_component_selector1="5"
           cdb_component_selector2="1"
           cacheid="camppend3-cdb-lightmap"
           cache_enabled="false">
  <cache_policy usage="none"/>
  <cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
   </image>
</lightmap>
```

60.2.3. Configuring the GeoSpecific (GS) Dataset (100)

Geospecific data replicates the real world data. The following example shows how to configure a geospecific dataset:

```
<model driver="cdb"
         name="GS Man Made"
         cdb_dataset="100"
         cdb_component_selector1="1"
         cdb_component_selector2="1"
         cacheid="camppend3-cdb-gsmodels">
  <cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
  <sigsize_file>../userData/servers/
    Sample_SignificantSize_SwitchDistance.xml</sigsize_file>
  <pagingdistance_file>../userData/servers/
    Sample_Model_PagingDistance.xml</pagingdistance_file>
  <lightdefaultvalues_file>../appData/cdb/
    Lights_DefaultValues_3_0.xml</lightdefaultvalues_file>
</model>
```

Table 60-4 describes the attributes for the geospecific dataset.

Table 60-4: Attributes for geospecific dataset

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be `cdb`).
cdb_dataset	Specifies the CDB dataset to read. For geospecific always use “100”.
cdb_component_selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 60-5. Default: 1.

Table 60-4: Attributes for geospecific dataset

Attribute Name	Description
cdb_component_selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cacheid	Specifies where to save the cache files if caching is enabled. If Accelerated File Loading is enabled, the CDB geospecific models are converted to MEDF and the textures are converted to MEIF on the first load. They will be used the next time the CDB database is loaded.
cdb_root	Specifies the top folder of the CDB database.
elevation_cache	(Optional) Creates an elevation binary cache file for clamp models. Default: true.
texture_mipmap_offset	(Optional) Specifies an offset to load lower mipmap textures from all geotypical textures. (Geospecific models can have geotypical textures.) Allowed values: 0-9. 0 means no offset. 1-9 means load a lower numbered LOD by the amount of the offset. For example if a model is using a texture LOD 9, setting the offset to 2 causes it to load LOD 7.
debug	(Optional) Enables (true) or disables (false) debug information on CDB tiles.
debug_modelcolor	(Optional) If set to true (debug also must be set to true), color the models with the same color as the tiles.
range_factor	(Optional) Multiplier of the tile paging values found in *Model_PagingDistance.xml*.
sigsize_file	(Optional) Replace the default significant size file (*./appData/cdb/SignificantSize_SwitchDistance.xml*) with another version.
pagingdistance_file	(Optional) Replace the default paging distance file (*./appData/cdb/default Model_PagingDistance.xml*) with another version.
lightdefaultvalues_file	(Optional) Replace the default light configuration file (*Lights_DefaultValues_3_2.xml*) with another one. Use this parameter only for Component Selector 1 (4 and 5, that is, Airport Light and Environment Light).

Table 60-5 lists geospecific sub-dataset codes.

Table 60-5: Geospecific sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
100	1	1	Man-made point features.
100	1	3	Man-made linear features (not supported).
100	1	5	Man-made areal features (not supported).
100	2	1	Natural point features.
100	2	2	Natural linear features (not supported).
100	2	5	Natural areal features (not supported).
100	3	1	Trees point features.
100	4	1	Airport light point features.
100	4	3	Airport linear features (not supported).
100	4	5	Airport areal features (not supported).
100	5	1	Environmental light point features.

Configuring the Geospecific Airport Lightpoint Sub-dataset (100-004)

The following example code shows how to configure a geospecific airport light-point dataset:

```
<model driver="cdb"
         name="GS Airport Light"
         cdb_dataset="100"
         cdb_component_selector1="4"
         cdb_component_selector2="1"
         cacheid="camppend3-cdb-gsairportlights">

  <cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
</model>
```

You can test the proper functioning of the lightpoints by turning them on and off using debugging tools:

To turn airport light points on or off using the debug window:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 55-5).
3. Select the Show Developer Debug Overlay check box. The Debug window opens (Figure 60-1).



Figure 60-1. Debug window

4. Expand Light Point Settings.
5. Toggle the slider of the runway you would like to change. For example, in the Camp Pendleton terrain, if you want to close the runway KNFG: 21, set the slider to 0.

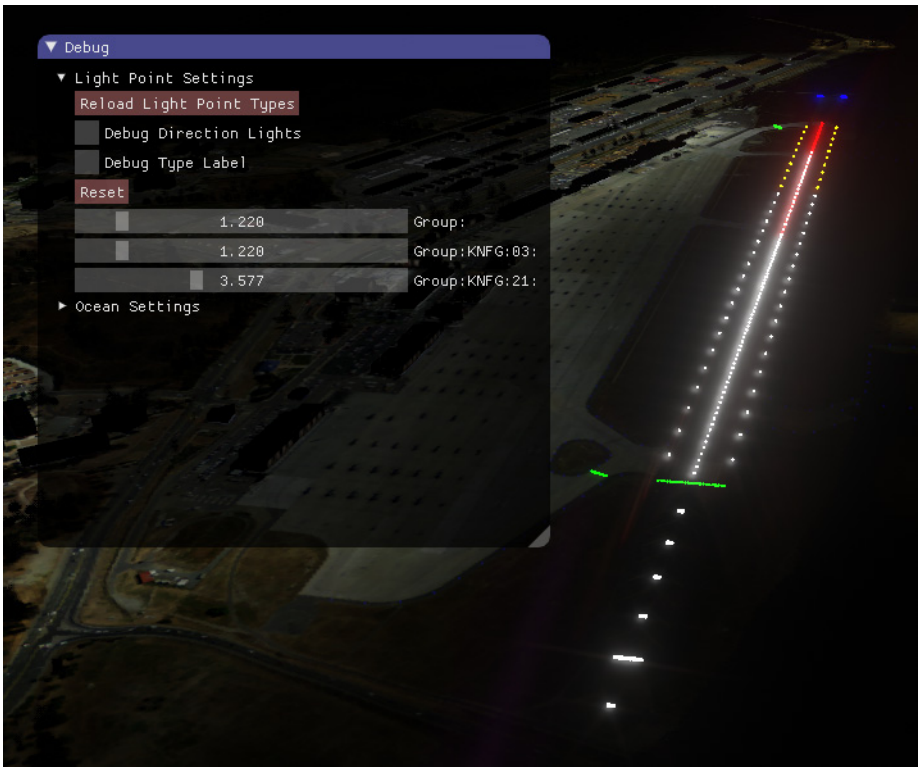


Figure 60-2. Camp Pendleton airport with debug window

Configuring the Geospecific Environmental Lightpoint Sub-dataset (100-005)

The following example shows how to configure geospecific environmental light-points:

```
<model driver="cdb"
           name="GS Environment Light"
           cdb_dataset="100"
           cdb_component_selector1="5"
           cdb_component_selector2="1"
           cacheid="camppend3-cdb-gsenvlights">
  <cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
</model>
```

Configuring Light Point Properties

You can change the behaviors of a type of light point. Light types are configured in *./appData/importConfig/Lights_DefaultValues_3_2.csv*. To change a light point type, you can edit the file in a text editor or spreadsheet. Each light type has the following attributes:

- Code: Light type code (do not edit).
- Description: Light type description (do not edit).
- Intensity: From 0.0 to 1.0.
- Red-Green-Blue: RGB value of the color (from 0.0 - 1.0 for each channel).
- Directionality: Directional, Omnidirectional, Bidirectional.
- Horizontal Lobe: When a light type is not native to the CDB specification, the Horizontal field represents the light point's half-intensity horizontal lobe width for the client-device (range from 0.0 to 360.0). When the light type is native to the CDB specification, the Horizontal field is used as a floating-point modifier that multiplies the horizontal lobe width calculated by the client-device. It applies only to Directional and Bidirectional light types. Default: 1.0.
- Vertical Lobe: When a light type is not native to the CDB specification, the Vertical field represents the light point's half-intensity vertical lobe width for the client-device (range from 0.0 to 360.0). When the light type is native to the CDB specification, the Vertical field is used as a floating-point modifier that multiplies the vertical lobe width calculated by the client-device. It applies only to Directional and Bidirectional light types. Default: 1.0.
- Residual Intensity: When a light type is not native to the CDB specification, Residual Intensity represents the residual intensity of the light. Residual Intensity is the intensity of the light (range normalized from 0.0 to 1.0) outside of the lobe defined by Lobe Width:Horizontal and Lobe Width:Vertical fields. When the light entry is native to the CDB specification, Residual Intensity is used as a floating-point modifier that multiplies the residual intensity calculated by the client-device. It applies only to Directional and Bidirectional light types. Default: 1.0.
- Frequency: A floating-point value greater than or equal to 0.0 that sets the blink or rotating frequency of the light in Hertz (cycles per second). A value of 0.0 disables all blinking and rotating properties. Default: 0.0.
- Duty Cycle: A floating-point value ranging from 0.0 to 1.0 that sets the duty cycle of the light. Duty cycle is defined as the percentage of time the light is turned on over a complete cycle. A value of 0.0 permanently turns the light off. A value of 1.0 turns it on. The value is ignored if Frequency = 0.0. Default: 0.5.
- Min Pixel Size: Minimum number of pixels that represent the light point when you are far from it. (This attribute is specific to VR-Vantage applications.)
- Max Pixel Size: Maximum number of pixels that represent the light point when you are close to it. (This attribute is specific to VR-Vantage applications.)
- Max Visible Distance: Distance in meters when the light point will disappear. (This attribute is specific to VR-Vantage applications.)

To change the behaviors of a light point type, you need to know what light type is being used.

To identify the type of light point:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 55-5).
3. Select the Show Developer Debug Overlay check box. The Debug window opens (Figure 60-1).
4. Expand Light Point Settings.
5. Select the Debug Type Label check box.
6. Zoom in on the light point that you want to edit. It is labeled with a number (Figure 60-3).



Figure 60-3. Lightpoint type labels

60.2.4. Configuring the GeoTypical (GT) Dataset (101)

Geotypical data is data that is typical for a particular type of object but does not necessarily match the object in the real world.

The following example shows how to add a geotypical dataset to an earth file:

```
<model driver="cdb"
         name="GT Trees"
         cdb_dataset="101"
         cdb_component_selector1="2"
         cdb_component_selector2="1"
         cacheid="camppend3-cdb-gtmodels2">
  <cdb_root>../../../../../../../Presagis/CDB/
    CDB_Camp_Pendleton_3_0</cdb_root>
  <sigsize_file>../userData/servers/
    Sample_SignificantSize_SwitchDistance.xml</sigsize_file>
  <pagingdistance_file>../userData/servers/
    Sample_Model_PagingDistance.xml</pagingdistance_file>
  <speedtree_catalog>Camp_Pendleton_3_0_SpeedTrees.xml
    </speedtree_catalog
  <features name="Vegetation"/>
  <styles>
    <style type="text/css">
    Vegetation {
      marker-scale:        1;
      marker-placement: vertex;
      altitude-clamping:   terrain;
      altitude-resolution: 0.00008583068847656250;
      altitude-offset:     0;
    }</style>
  </styles>
</model>
```

Table 60-6 describes the attributes for a geotypical dataset.

Table 60-6: Attributes for geotypical dataset

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be `cdb`).
cdb_dataset	(Optional) Specifies the CDB dataset to read. For geotypical data always use “101”.
cdb_component_selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 60-7. Default: 1.
cdb_component_selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.

Table 60-6: Attributes for geotypical dataset

Attribute Name	Description
cacheid	Specifies where to save the cache files if caching is enabled. If Accelerated File Loading is enabled, the CDB geotypical models are converted to medf and the textures are converted to meif on the first load. They will be used the next time the CDB database is loaded.
cdb_root	Specifies the top folder of the CDB database.
elevation_cache	(Optional) Creates an elevation binary cache file for clamp models. Default: true.
texture_mipmap_offset	(Optional) Specifies an offset to load lower mipmap textures from all geotypical textures. Allowed values: 0-9. 0 means no offset. 1-9 means load a lower numbered LOD by the amount of the offset. For example if a model is using a texture LOD 9, setting the offset to 2 causes it to load LOD 7.
cache_policy	(Optional) Specifies whether or not to cache models.
debug	(Optional) Enables debug information on CDB tiles.
debug_modelcolor	(Optional) If set to true, color the models with the same color as the tiles. The debug attribute must also be set to true.
range_factor	(Optional) Multiplier of the tile paging values found in *Model_PagingDistance.xml*.
sigsize_file	(Optional) Replace the default significant size file (*./appData/cdb/SignificantSize_SwitchDistance.xml*) with another version.
pagingdistance_file	(Optional) Replace the default paging distance file (*./appData/cdb/default Model_PagingDistance.xml*) with another version.
rendering_method	Valid values are OSG_RENDERING or INDIRECT_RENDERING. Default: INDIRECT_RENDERING. Using the indirect rendering value sets this model layer to use the optimized indirect rendering system, which allows the models to be rendered much faster than the OSG rendering method.
speedtree_catalog	(Optional) Specifies a file that lists the CDB geotypical SpeedTree replacement. (For more information, please see "CDB Configuration Files," on page 60-17.) The `features` and `styles` elements must be present as in the example for SpeedTrees to work with CDB.

Table 60-7 lists the codes for the geotypical sub-dataset.

Table 60-7: Codes for geotypical sub-datasets

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
101	1	1	Man-made point features.
101	1	3	Man-made linear features (not supported).
101	1	5	Man-made areal features (not supported).
101	2	1	Tree point features.
101	2	2	Tree linear features (not supported).
101	2	5	Tree areal features (not supported).
101	3	1	Moving model location features (not supported).

60.2.5. Configuring the Extended Imagery Dataset (900)

The following example shows how to add the extended imagery dataset to an earth file.



This dataset needs the CAE API pro license and the configuration file *.\appData\cdb\900_ExtImagery.xml*. These extensions are not part of the CDB specification, but can permit vendors to extend CDB to their needs.

```
<image name="cdb_cartography"
         driver="cdbraster"
         cdb_dataset="900"
         cdb_component_selector1="1"
         cdb_component_selector2="1"
         cdb_cache_file="25"
         cacheid="yemen-cdb-cartography"
  <cdb_root>../../../../../../../Presagis/CDB/CDB_Yemen_3_2</cdb_root>
</image>
```

Table 60-8 describes the attributes for configuring extended imagery.

Table 60-8: Extended imagery attributes

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be `cdbraster`).
cdb_dataset	(Optional) Specifies the CDB dataset to read. For imagery data always use “900”. Default: 1.
cdb_component_selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 60-9. Default: 1.
cdb_component_selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cdb_cache_file	Specifies how many CDB tiles to keep in memory while building an OSG tile. Default: 20.
cacheid	Specifies where to save the cache files if caching is enabled.
cdb_root	Specifies the top folder of the CDB database.
cdb_max_lod	(Optional) Specifies the maximum LOD of this dataset. Set this value if you have a CDB database that is larger than 500 geocells. For example, if you have the CDB World Wide database you can set that value to 0. This will cause the CDB driver to pre-parse all of the LOD folder. If the value is not set (default) it reads the CDB to find the maximum LOD of the imagery dataset.

Table 60-9 lists the sub-dataset code for extended imagery.

Table 60-9: Extended imagery sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
900	1	1	Cartography imagery.

60.3. CDB Configuration Files

VR-Forces uses the following files to configure CDB terrain databases.

- *Datasets.xml*. This file lists the CDB 3.2 Datasets code and folder name. Do not edit this file until a new CDB specification is released or MAK creates an extended CDB dataset. This list is read by the MAK CDB API when the CDB DLLs are loaded.
- *Feature_Data_Dictionary.xml*. This file contains all the FACC, FSC code (Facc sub code) definitions from the CDB 3.2 specification. Do not edit this file.
- *SignificantSize_SwitchDistance.xml*. CDB uses the concept of Significant Size to switch between LODs. Since this concept is not supported currently by makOSG, the makOSG Openflight plug-in was modified to support FLT CDB models. To support the CDB model Significant Size, the plug-in was modified to convert SigSize into normal LOD switch in/out distance. The Switch distance is supported by makOSG. The plug-in contains default values that can be overwritten by this file. As a guideline, each CDB LOD in the file (also associated to a significant size) should have a different switch in distance greater than the next LOD (this will prevent abnormal behavior at runtime).

  Inside this file, switch distance can be defined by CDB datasets.

  You can override this file for each dataset by adding the following code to your earth file:

  ```
  <sigsize_file>.\yourSigSizeFile.xml</sigsize_file>
  ```

- *Model_PagingDistance.xml*. Configures the distance for the view point when each LOD tile is getting loaded. This can be configured by CDB datasets.

  This file can be overwritten in the earth file for each dataset by using:

  ```
  <pagingdistance_file>./AnotherModelPagingDistanceFile.xml</pagingdistance
    _file>
  ```

- *Light_DefaultValues_3_2.xml*. This file contains all the CDB 3.2 light point default values defined in *CDB Specification - Volume 2.pdf*, Appendix E. By default, VR-Vantage uses *./appData/importConfig/Light_DefaultValues_3_2.csv* to configure light properties for CDB databases. You can override it by adding the `<lightdefaultvalues_file>` element to your earth file and specifying *Light_DefaultValues_3_2.xml* (or *Light_DefaultValues_3_0.xml*) as the source of light properties.

  If it is specified, the Model_CDB plug-in reads this file at load time.

  These values complete the information found in your CDB database under *./Metadata/Lights.xml*. Do not edit this file unless an error is found.

You can specify a light properties file for each light dataset by using:

```
<lightdefaultvalues_file>./AnotherLightDefaultValuesFile.xml</lightdefaul
  tvalues_file>
```

For example, if you want to use the CDB 3.0 default values you could add this to the airport lights section:

```
<lightdefaultvalues_file>./appData/cdb/Light_DefaultValues_3_0.xml</light
  defaultvalues_file>
```

- *Light_DefaultValues_3_0.xml*. This file contains all the CDB 3.0 light point default values defined in *CDB Specification - Volume 2.pdf*, Appendix E. Do not edit this file unless you find an error in it except for the following attributes:
  - `<Radius>12.0</Radius>`. Radius in meters of the OSG light point.
  - `<MinPixelSize>2.0</MinPixelSize>`. Minimum pixel size of the OSG light point node.
  - `<MaxPixelSize>4.0</MaxPixelSize>`. Maximum pixel size of the OSG light point node.
  - `<MaxVisibleDistance>12000</MaxVisibleDistance>`. Maximum visible distance in meters.
- *Lights_xxx.xml*. This file shows how to do specific light tuning. Do not change the name of this file. The content must be updated with your device specific tuning and you must put the file in your CDB database's */Metadata* directory.
- *900_ExtImagery.xml*. This file is needed to extend the API to support the 900_ExtImagery dataset in the Yemen CDB database. (You must have a CAE API pro license to support this.)
- *XXX_SpeedTrees.xml*. This file is used to map speedTrees to the CDB geotypical model. For example, you can create a file called *Camp_Pendleton_3_0_SpeedTrees.xml* that will be referenced in the Geotypical tree section of your earth file. (For details, please see "Configuring CDB earth Files," on page 60-2). The file will usually be in the same folder as the earth file.

60.4. Caching CDB Data (cdbtool)

When caching is enabled, VR-Forces automatically caches terrain data as you load it and stores the data for future use.

VR-Forces has multiple level of caching for CDB data:

- Models and textures. For this level of caching, set the `cacheid` field in your earth file. We recommend that you give it an appropriate name, such as `camppend3-cdb-gtmanmademodels`.
- Model clamping elevation. This is done automatically the first time CDB tiles are paged in if it is not disabled in the earth file.
- Height map and normal map. This is done automatically the first time CDB tiles are paged in.
- Elevation and imagery. If you enable osg caching in the earth file, this provides some load performance gain, but may require a lot of disk space.

When you cache data as it loads, initial load times can be lengthy. You can use the cdbtool to compress CDB models (flt) and textures (rgb) offline to speed up initial load times and runtime performance. (You can also run the tool on CDB databases that have been partially cached by automatic caching. In that case the tool only compresses the models and textures that have not yet been compressed.)

The syntax for the CDBTool is as follows:

```
cdbtool -f earth_file [-p threads -c cache_path -- -v -h]
```

Where:

`(-p | --threads)` *`threads`* is the number of threads to use. Too many threads will be slow and run out of memory.

`(-c | --CachePath)` *`cache_path`* is the directory to use to save the information. Default: *./appData/cache*.

`(-f | --EarthFile)` *`earth_file`* is the input earth file name and path containing the CDB datasets information.

`(-- | --ignore_rest)` causes VR-Forces to ignore the rest of the arguments following this flag.

`(-v | --version)` displays version information and exits.

`(-h | --help)` displays usage information and exits.

The following example caches *CampPendleton3-CDB.earth* in the default directory using four threads:

```
cdbtool -f "..\userData\servers\CampPendleton3-CDB.earth"
  -p 4
```



61. Generating Navigation Data

This chapter explains how to generate navigation data for use with VR-Forces.

Introduction 61-2
How Navigation Data is Generated 61-2
Path Finding 61-4
Navigation Areas 61-7
Sectorizing Navigation Areas 61-7
Creating a Navigation Area 61-8
Editing a Navigation Area 61-11
Reverting Edits to a Navigation Area 61-11
Displaying Navigation Areas 61-11
Removing a Navigation Area 61-11
Displaying Information about a Navigation Area 61-12
Generating Navigation Data 61-13
Navigation Profiles 61-13
Generating Navigation Data for a Navigation Area 61-14
Using Newly Generated Navigation Data 61-16
Automatic Regeneration of Navigation Data 61-16
Generating Navigation Data for Entities 61-17
Importing Navigation Areas and Navigation Data 61-19
Displaying Navigation Data 61-20
Opening Navigation Lab from an Information Dialog Box 61-21
Showing Simulation Object Movement in Navigation Lab 61-22

61.1. Introduction

To use advanced navigation, you must use a terrain that has navigation data generated for the areas in which you want to navigate. You generate navigation data in the VR-Forces front-end.



The ability to create navigation areas is an extra-cost feature. VR-Forces includes navigation data for the terrains and entity models it includes. If you want to be able to generate navigation data for your terrains or models, you must purchase an additional license.

When you generate navigation data, the VR-Forces creates separate sets of navigation data, at various levels of detail, for specified profiles. VR-Forces includes profiles for lifeforms and ground vehicles. You can modify the profiles and add new ones. The navigation data is stored in separate directories where it can be accessed by multiple terrains, if needed.

The general procedure for generating navigation data is as follows:

1. Load the terrain on which you want to generate navigation data or load a scenario that uses that terrain.
2. Create a navigation area.
3. Generate the navigation data.
4. Save the terrain.
5. Start a new scenario or save and reload the existing scenario.

61.1.1. How Navigation Data is Generated

Navigation data is generated from the front-end. Before you can use newly generated navigation data, you must close your scenario and reopen it. (This is so it can be loaded by the back-end.) If the terrain is modified, the navigation data must be generated again.

VR-Forces analyzes the terrain and creates a navigation mesh (NavMesh) of the terrain. The NavMesh defines areas in which an entity can move, given its dimensions. A lifeform can move in spaces that a ground vehicle cannot move in. Figure 61-1 shows a NavMesh for a terrain that has several areas in which entities cannot move.

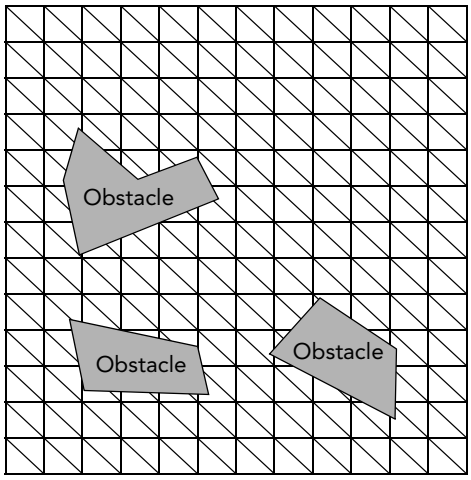


Figure 61-1. NavMesh

Based on the NavMesh, VR-Forces generates a graph of locations in the terrain that the entity type can move to. Each location can be accessed from at least one other location. Figure 61-2 shows a hypothetical graph of the paths that an entity can follow in the NavMesh shown in Figure 61-1.

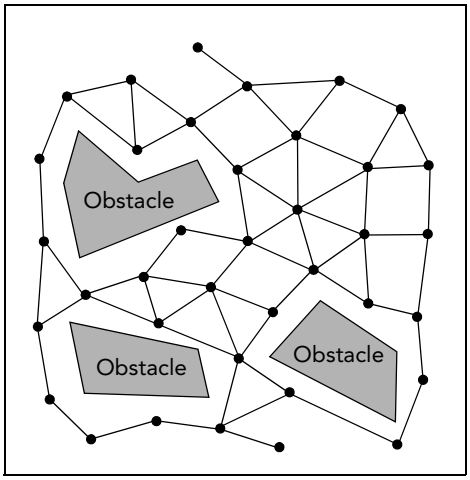


Figure 61-2. Graph

For details about generating navigation data, please see Chapter 61, *Generating Navigation Data*.

61.1.2. Path Finding

Path finding is the process by which an entity plans a path, follows the path, and avoids obstacles along the path.

Planning a Path

Every time an entity decides to move (either through an autonomous decision or a task assignment), it computes its path. Path computation, or path planning consists of running an A* algorithm, with a given heuristic, on the navigation data.

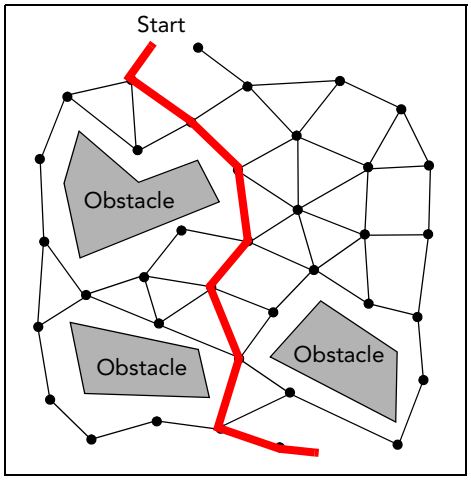


Figure 61-3. Planning a path

A path finding heuristic is a cost estimate for the shortest path to reach a target. This cost can be measured according to various criteria, such as distance, time, and danger. The path finding heuristic is critical for the performance of the A* algorithm used by VR-Forces.

The standard path finding heuristic is the Euclidian Heuristic. In this heuristic, the cost of moving from one vertex to another vertex is the Euclidian length of the connecting edge. VR-Forces also uses a Road Constrained Heuristic, in which a ground vehicle computing its path is forced to move only on roads.

Path Following

Once a path is determined for an entity, it can start moving to its final goal. As illustrated in Figure 61-3, a path is a set of vertices and edges. However, entities do not necessarily follow the exact path. VR-Forces smooths the entity's movement.

The first intermediary destination (or subgoal) of the entity is the nearest vertex. Before moving towards this point, the entity checks to see if it can move directly to the next vertex in its path. This test is performed against the AI mesh. If the next vertex is reachable, it becomes the new subgoal. The entity continues testing successive vertices until it finds a vertex that it cannot reach directly. The final trajectory is a smooth line that connects the various subgoals.

Figures 61-4 and 61-5 illustrate this process. Using the prospective path defined in Figure 61-3, the entity computes its actual path. In Figure 61-4, view 1, it tests the first three vertices (dashed lines) and finds that it can reach all of them directly. In Figure 61-4, view 2, it tests the fourth vertex and finds that it can reach it. It then tests the fifth vertex and finds it cannot reach it directly. Therefore, the fourth vertex becomes the subgoal for this portion of the path. Figure 61-5 shows the final calculated path (dashed line). Notice how the path to the fourth vertex is much smoother than a path that strictly follows the vertices and edges would be, because the entity determined that it could move directly to that vertex.

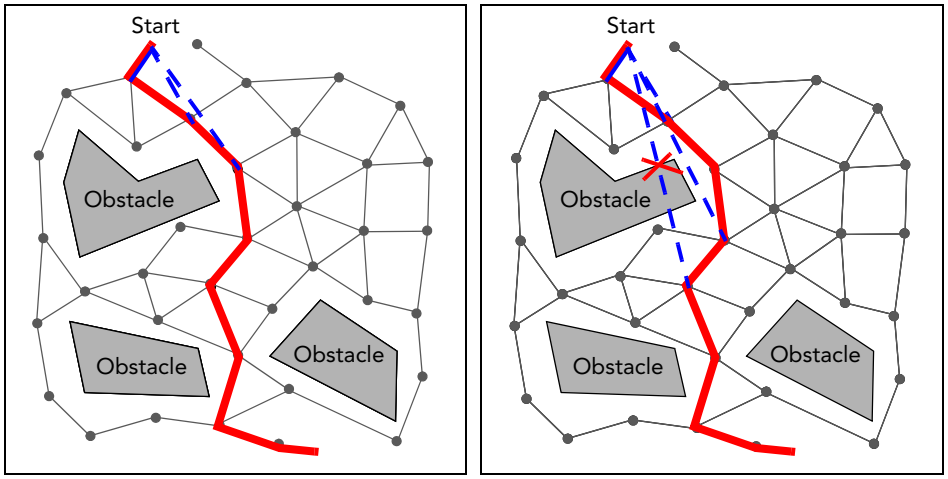


Figure 61-4. Checking next vertex

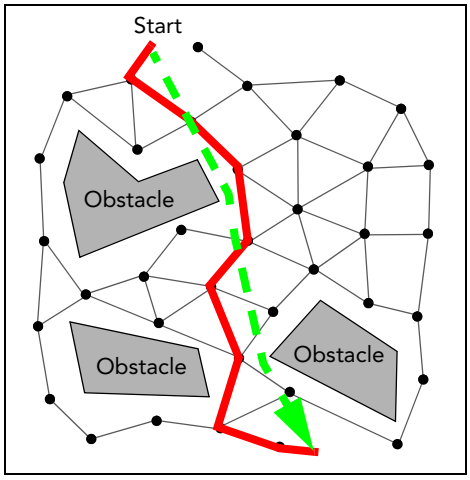


Figure 61-5. Final calculated path

Dynamic Avoidance

While following its path, an entity may encounter other entities. VR-Forces models include collision avoidance. However in complex scenarios, the scenario developer may have to spend a considerable amount of time creating routes and coordinating entities to minimize collisions among entities and obstacles.

Advanced navigation uses a dynamic avoidance algorithm to make entities behave realistically when they are about to collide. Entities detect potential collisions in advance and adjust their trajectories smoothly using the navigation data.

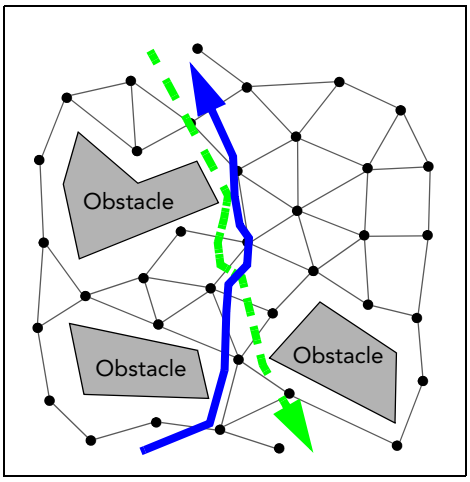


Figure 61-6. Collision avoidance

In Figure 61-6, two entities traveling in opposite directions avoid each other, then resume their routes. Compare the track of the dashed line to that in Figure 61-5.

61.2. Navigation Areas

Navigation areas define areas on the terrain for which you can generate navigation data. Navigation areas are rectangular areas that are aligned to the North/South axis. The maximum size of the area on which you can generate navigation data depends on the raster precision value used to generate it. If you reduce the precision of navigation data, creating a sparser graph, the size of the maximum area increases. The maximum size for a navigation area using the default values is 20 km by 20 km. (Raster precision is configured in *./appData/settings/vrfSim/navigationProfiles.mtl*.)

You can create multiple navigation areas on a terrain.

61.2.1. Sectorizing Navigation Areas

You can generate navigation data for a navigation area as a whole or you can divide the navigation area into sectors and generate navigation data for each sector. VR-Forces can regenerate navigation data at runtime in response to changes in the terrain (dynamic terrain damage and placement of buildings). If you sectorize the navigation area, regeneration is much quicker because VR-Forces can restrict regeneration to the sector that is affected rather than the entire navigation area.

The disadvantages of using sectors are as follows:

- Increasing the number of sectors slows down generation of navigation data.
- The navigation data generation process can fail if any single sector is too large, too complex, or both. However, this is also true if you generate navigation data for a very large terrain without using sectors.
- Path planning over long distances may be slower.

To generate a maximum size navigation area, you must use multiple sectors.

61.2.2. Creating a Navigation Area

Navigation areas get created on the terrain used by the simulation engine. In most cases, scenarios use the same terrain for the GUI and simulation engine and there is no need to think about where navigation areas get created. However, sometimes scenarios use different terrains for the GUI and simulation engine. In those cases note the following:

- The GUI only displays the navigation areas that are in the simulation terrain. If the GUI terrain has navigation areas they are not displayed.
- When you create a navigation area it is displayed in the coordinate system used by the GUI terrain. Any data entry is in the GUI's coordinate system. However, the navigation area configuration files are saved in the coordinate system of the simulation terrain.
- The simulation terrain must be available to the GUI. If the GUI and simulation engine are running on different machines, it is possible that the files for the terrain used by the simulation engine are not on the computer that is running the GUI. However, for navigation areas to get created, the GUI must have access to the simulation terrain.
- Props get created on the GUI terrain. If you want props to be part of the navigation data, you must add them in a scenario that has the same terrain for the GUI and simulation engine.

To create a navigation area:

1. Open a scenario or a terrain on which you want to generate navigation data.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Navigation Areas page (Figure 61-7).

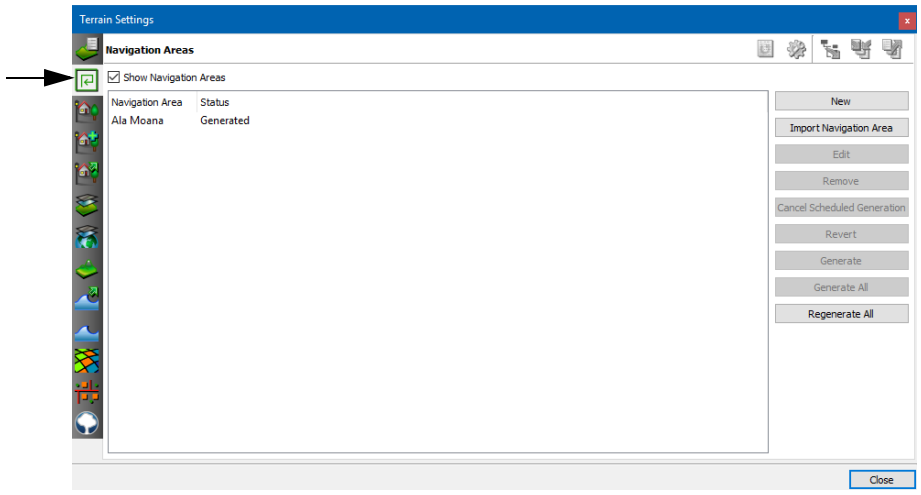


Figure 61-7. Navigation Areas page

4. Click New. The cursor changes to create mode and shows a box that represents the navigation area. The default size is 500 m by 500 m. The Create Navigation Area tab is added to the window.
5. Click to place the lower left corner of the navigation area (Figure 61-8).

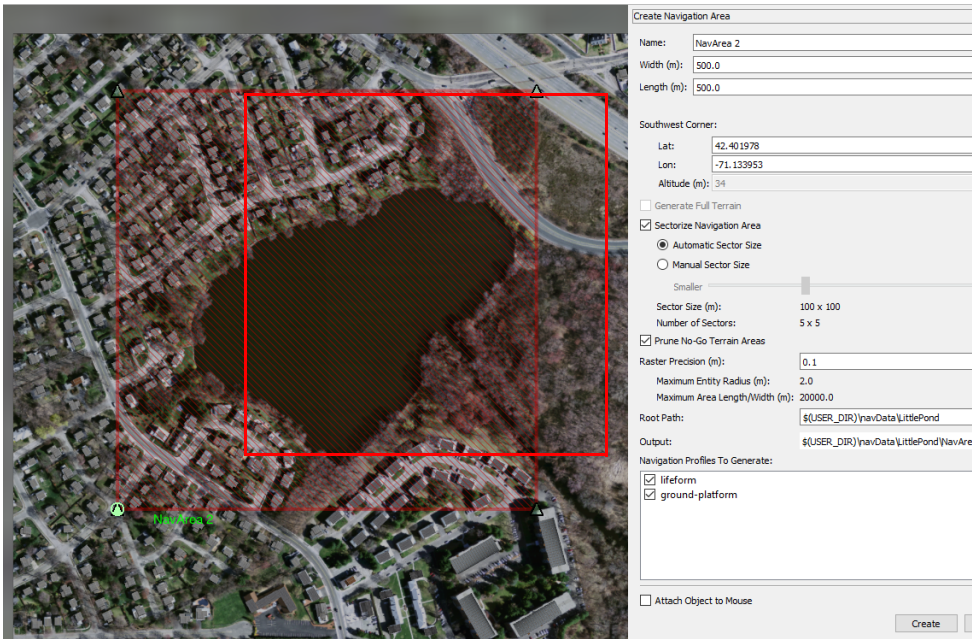


Figure 61-8. New navigation area

6. In the Create Navigation Area dialog box, edit the properties of the navigation area. Table 61-7 describes the fields in the dialog box.

Table 61-1: Navigation area properties

Property	Description
Name	The name of the navigation area. You cannot change the name after you create the navigation area.
Width Length	The length and width of the navigation area. Do not try to create an area larger than 20 km by 20 km unless you have changed the raster precision to support a larger area.
Southwest Corner	The latitude and longitude of the southwest corner of the navigation area. (The altitude value is not used.)
Generate Full Terrain	Generate navigation data for the entire terrain. If the terrain is too large for this option, it is disabled.
Sectorize Navigation Area	Generate navigation data for sectors, rather than one large area.

! If you are creating a large navigation area, we recommend that you always sectorize it. Generation of large unsectorized navigation areas sometimes fails.

Automatic Sector Size	If selected, VR-Forces chooses a sector size based on the size of the navigation area. The sector size and number of sectors is shown below the manual sector size slider.
Manual Sector Size	Allows you to choose a sector size using the slider. As you move the slider, the sector size and number of sectors is shown below the slider.
Prune No-Go Terrain Areas	Do not calculate navigation data for areas in which simulation objects cannot move.
Raster Precision	Determines the density of the navigation graph.
Root Path	The root path of the data for this terrain.
Output Path	The location to save navigation data to.
Navigation Profiles to Generate	Select the check boxes for the platform types for which you want to generate navigation data.

7. Click Create.

61.2.3. Editing a Navigation Area

After you create a navigation area, you can edit any of its attributes except its name.

To edit a navigation area:

1. Open a scenario or a terrain that has the navigation area you want to edit.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Navigation Areas page (Figure 61-7).
4. Select the navigation area you want to edit.
5. Click Edit. The Edit Navigation Area dialog box opens.
6. Make the change you want to make.
7. Click OK.

61.2.4. Reverting Edits to a Navigation Area

If you edit a navigation area, you can revert to the previous version of the area.

To revert an edited navigation area:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Navigation Areas page (Figure 61-7).
3. Select the navigation area you want to revert.
4. Click Revert.

61.2.5. Displaying Navigation Areas

➤ To display navigation areas, choose **Settings** → **Navigation Areas**, click the Navigation Area button () on the Display Settings toolbar, or select the Show Navigation Areas check box on the Terrain Settings dialog box, Navigation Areas page.

61.2.6. Removing a Navigation Area

To remove a navigation area:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Navigation Areas page (Figure 61-7).
3. Select the navigation area you want to remove.
4. Click Remove.

61.2.7. Displaying Information about a Navigation Area

You can display an Information dialog box that has information about the selected navigation area (Figure 61-9). By default, the dialog box does not display the port used by the navigation area. It is marked (Disabled). To display the port, in *vrfSim.mtl*, set enableNavigationLabDebugging to 1.

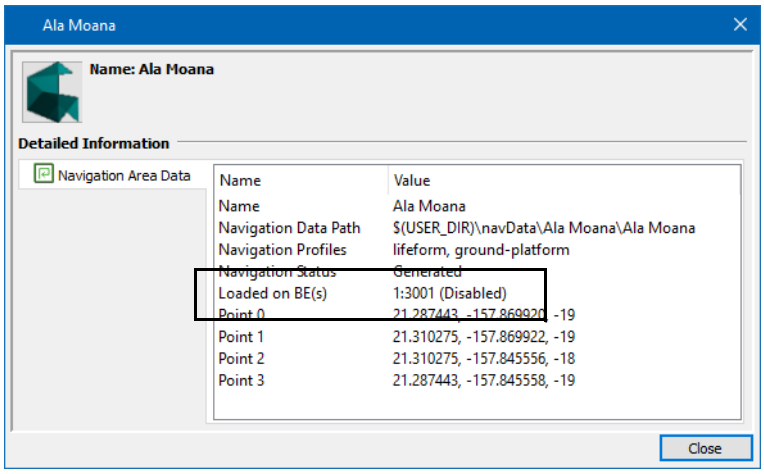


Figure 61-9. Navigation area information

61.3. Generating Navigation Data

To generate navigation data, you create navigation areas and then generate navigation data for them. A terrain can have more than one navigation area. Entities use advanced navigation within the areas and standard navigation if they move between them.

Generating navigation data can last from a few minutes to hours to days depending on the terrain complexity, the level of detail, and your computer's processing power. Navigation data is saved to *./userData/navData* in a directory that matches the name of the terrain on which it was generated. In addition to using sectors ("Sectorizing Navigation Areas," on page 61-7), generating cover points can significantly affect the amount of time it takes to generate navigation data.

By default, when you generate navigation data for lifeforms, VR-Forces calculates cover points. For large and complex terrains, this process can be very slow. MAK has recorded this taking 1 week to finish on a maximum size navigation area. You can disable calculation of cover points in the navigation profile. However, if you disable calculation of cover points, tasks that use cover points will not work in the navigation area

61.3.1. Navigation Profiles

VR-Forces generates navigation data based on parameters in a profile. VR-Forces includes profiles for lifeforms and ground entities. They provide default values for generating navigation data that should work for most terrains. You can change the profiles. You can also add profiles. Profiles are configured in *./appData/settings/vrfSim/navigationProfiles.mtl*. The configuration parameters are explained in the file.

Navigation profiles include parameters that control the priority of navigation data regeneration in response to terrain changes and the types of changes that trigger regeneration. For more information about regenerating navigation data, please see "Automatic Regeneration of Navigation Data," on page 61-16.

The `nav-edges` block in a navigation profile lets you connect points in the terrain that would not normally be connected by the navigation data generation process. This feature, for example, allows a human entity to walk up to a door, open the door, walk through the door, and then close the door. This is an advanced feature. For details, please see the comments in *navigationProfiles.mtl* or contact support@mak.com.



You can dynamically add nav edges to a navigation area using nav edge tactical graphics. For details, please see "Creating Nav Edges," on page 40-12.

The profile that an entity uses for advanced navigation is specified in its OPE file. For example, *Human.ope* specifies use of the lifeform profile:

```
(navigation-parameters
  (DtRwString navigation-profile  "lifeform")
  (DtRwString navigation-method  "shortcut")
)
```

61.3.2. Generating Navigation Data for a Navigation Area

The Navigation Areas page (Figure 61-7) displays the status for each navigation area:

- Generated. Data has been generated.
- Needs Generation. The navigation area needs to have navigation data generated.
- Scheduled Generation. VR-Forces is generating navigation data for multiple navigation areas. This area is in the queue for generation.

You can generate navigation data for:

- Selected navigation areas.
- For all navigation areas that need to have navigation data generated.
- For all navigation areas, including those that already have navigation data.

When you generate navigation data, the generator does not use the version of the terrain that is loaded into the simulation engine or GUI. It uses a separate version. Therefore, if you have made any changes to the terrain, for example by adding props, and you want them to be part of the updated terrain, you must save the terrain before you generate navigation data.

To generate navigation data:

1. Open a scenario or a terrain on which you want to generate navigation data.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Navigation Areas page (Figure 61-7).
4. Select the navigation areas for which you want to generate navigation data.
5. Click Generate.
6. You are prompted to save the terrain before navigation data is generated.
7. Click Yes. The Navigation Area Path Generation dialog box opens (Figure 61-10). You can optionally view details of the generation process or cancel the process. When the navigation data is generated, the dialog box buttons redisplay. You are prompted to reload the scenario.

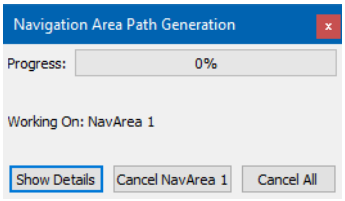


Figure 61-10. Navigation Area Path Generation dialog box

8. Click OK.
9. Click Finish.
10. Save the terrain. (**File** → **Terrain** → **Save Terrain**.)
11. To use the navigation data, reload the scenario.

Removing a Navigation Area from the Generation Queue

If you schedule multiple navigation areas for navigation data generation by clicking Generate All or Regenerate All, you can remove areas from the generation queue. If you remove a navigation area for which you had already generated navigation data, its status returns to Generated. If you remove a navigation area that has not had navigation data generated, its status returns to Need Generation.

To remove a navigation area from the generation queue:

1. Select the navigation area in the Navigation Areas list.
2. Click Cancel Scheduled Generation.

Canceling Generation of Navigation Data

You can cancel generation of navigation data while it is being generated for a navigation area. If you cancel generation of navigation data, the status for the navigation area changes to Needs Generation, even if you previously generated navigation data for this area.

To cancel generation of navigation data while it is being generated, do one of the following:

- Select the navigation area in the Navigation Areas window and click Cancel Scheduled Generation.
- In the Navigation Area Path Generation dialog box, click Cancel NavArea *n*, where *n* is the number of the area currently being generated.

61.3.3. Using Newly Generated Navigation Data

To use newly generated navigation data, create a scenario that uses the terrain or open a scenario that uses that terrain. If you generate navigation data while a scenario is open, you must close the scenario, then open it again before you can use the navigation data.

61.3.4. Automatic Regeneration of Navigation Data

By default, VR-Forces automatically generates navigation data if a dynamic terrain change occurs in a navigation area. Unless your navigation area is quite small, regenerating navigation data will be much quicker if you create the navigation area with sectors.

The speed at which navigation data is regenerated depends on different factors (most notably sector size, number of affected sectors, and the complexity of the terrain), but in all cases it is not instantaneous. It takes at least several seconds, and possibly considerably longer. It is not abnormal for regeneration to take up to 20-30 seconds even in a well sectorized terrain. Therefore it is normal for entities to continue navigating using the original navigation data for a brief period until navigation data is updated.

Regeneration of navigation data is configured in *./appData/settings/vrfSim/navigationProfiles.mtl*.

You can specify the types of dynamic terrain change that trigger regeneration of navigation data and a priority for how quickly regeneration takes place relative to other profiles. The following code configures regeneration of navigation data for the lifeform profile. Please see the file for explanatory comments.

```
(regeneration-priority 1)
(regenerate-for-dynamic-terrain-types
  (dynamic-terrain-type  "entity")
  (dynamic-terrain-type  "damage")
  (dynamic-terrain-type  "damage_structure")
)
```



Regeneration of navigation data in response to terrain changes does not require an additional license. Therefore, terrains that have navigation areas (which require an extra cost license to generate) can be distributed to VR-Forces users and regeneration of navigation data will work even if those users do not have a license for creating navigation areas.

61.4. Generating Navigation Data for Entities

You can generate navigation data for entities that have object geometry. For example, this allows human and ground entities to move around on the deck of an aircraft carrier or some other surface entity. Navigation data gets generated for any surface on the entity that is accessible for the profile. For example, lifeforms can move on steeper surfaces than ground platforms. You generate navigation data for entities in the Simulation Object Editor. For complete details about using the Simulation Object Editor, please see Section XII, "Creating and Editing Simulation Models".

When you generate navigation data for an entity, it gets saved in the *vrfSim/navData* directory for the SMS. It is saved immediately, regardless of whether or not you save changes to the SMS.

To generate navigation data for an entity:

1. Start the Simulation Object Editor, as described in "Starting the Simulation Object Editor," on page 67-4.
2. Load the SMS for the entity you want to edit, for example *EntityLevel.sms*.
3. In the object list, select the entity for which you want to generate navigation data.
4. Select the Movement tab.
5. Select the Use Detailed Geometry in Simulation Engine check box (Figure 61-11).

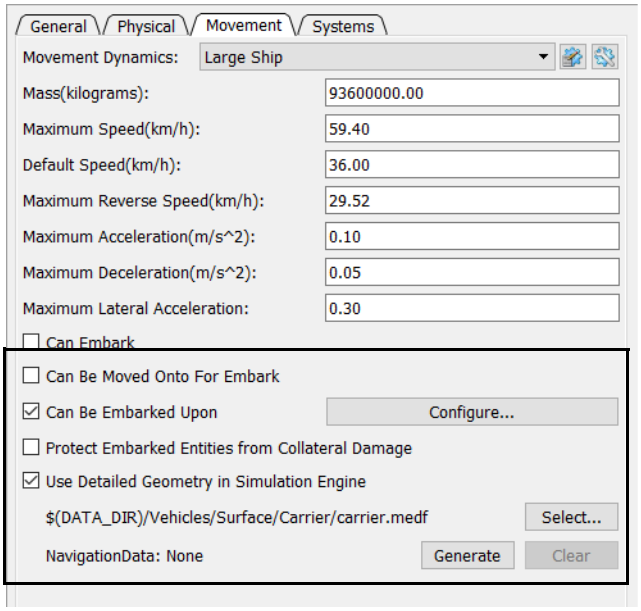


Figure 61-11. Object geometry

6. If you have not specified geometry for this entity, click Select and select the appropriate geometry file. (For complete details, please see "Adding Object Geometry to a Simulation Object," on page 69-7.)
7. Click Generate. The Generating Navigation Data dialog box opens.
8. Specify the platform types for which you want to generate navigation data.
9. Click Generate. VR-Forces generates navigation data (Figure 61-12). When the process finishes, the Cancel button changes to read Complete.

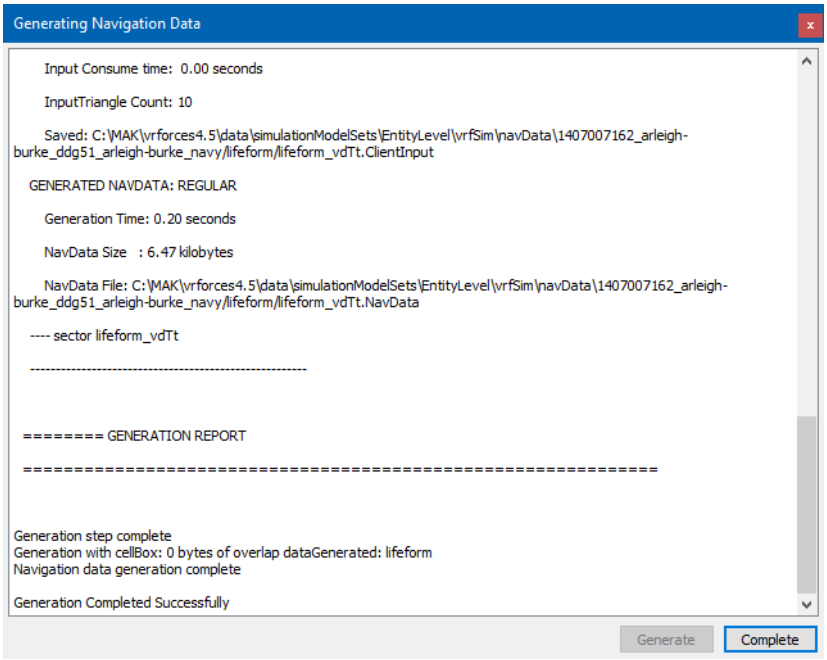


Figure 61-12. Generating Navigation Data dialog box

10. Click Complete.

61.5. Importing Navigation Areas and Navigation Data

When you generate navigation data for a navigation area, navigation area configuration files and the navigation data itself are saved to *./userData/navData*. For example, the configuration files for the Ala Moana navigation area provided with VR-Forces are *Ala Moana.navGenConfig* and *Ala Moana.navRuntimeConfig*.

You can import the navigation area and navigation data into other terrains that have the same coordinate systems and cover the same area of the earth. When you import a navigation area, it is imported with the status Generated.

To import a navigation area and its associated navigation data:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Navigation Areas page (Figure 61-7).
3. Click Import Navigation Area. A file browser window opens.
4. Navigate to the directory for the navigation area that you want to import and select the *.navRuntimeConfig* configuration file for the navigation area.
5. Click Open. The navigation area is added to the terrain.
6. Save the terrain.

61.6. Displaying Navigation Data

VR-Forces includes Autodesk® Gameware Navigation Lab software (Navigation Lab), a tool that lets you view navigation data. It also lets you connect to a simulation and watch entities move about on the terrain.

You can view navigation data for navigation areas and for the object geometry on entities. You can open Navigation Lab and use its menus to select the navigation data to view or you can open it directly from an Information dialog box.

Figure 61-13 displays the navigation data for VR-Village.

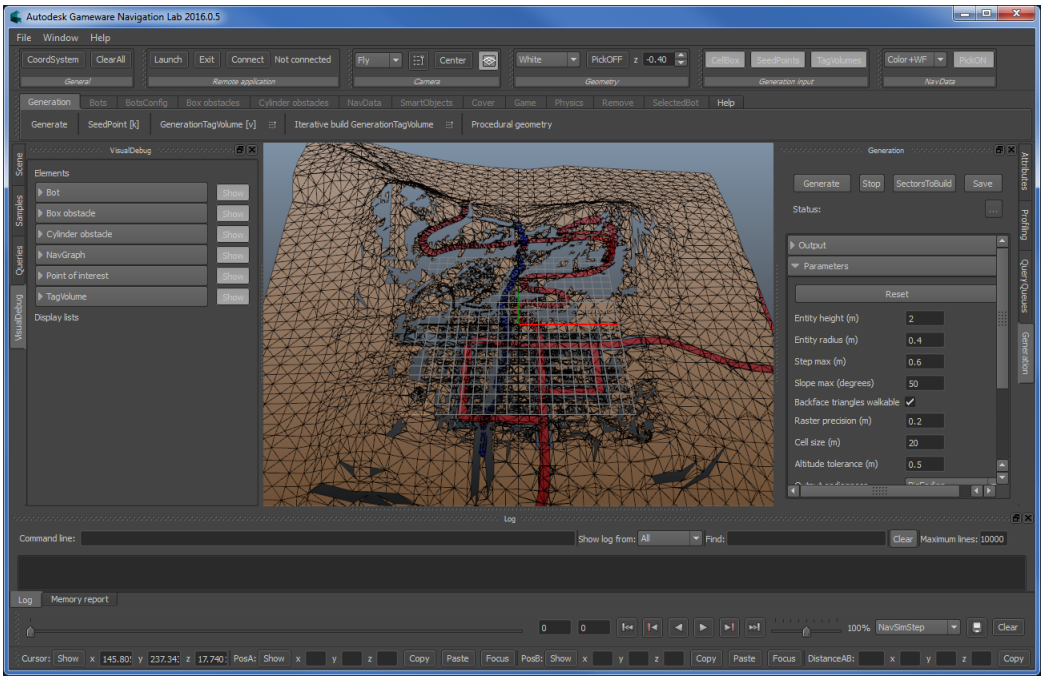


Figure 61-13. Navigation data display

Autodesk® Gameware Navigation Lab is not supported on Linux.

To display navigation data by opening it in Navigation Lab:

1. On the Start menu, choose **MAK Technologies** → **Navigation Lab**.

   Navigation Lab opens.

2. Choose **File** → **Open**. A file browser opens.

3. To view navigation data for a terrain, navigate to a terrain under *./userData/navData*. To view navigation data for an entity, navigate to a directory in *./data/simulationModelSets/EntityLevel/vrfSim/navData*.

4. Select a file with the *.NavData* extension. The navigation data is displayed in the Navigation Lab window.

61.6.1. Opening Navigation Lab from an Information Dialog Box

Navigation areas and entities that have object geometry and navigation data have a button on their Information dialog box that you can use to quickly display their navigation data in Navigation Lab.

To open Navigation Lab from an Information dialog box:

1. Open the Information dialog box for the navigation area or entity for which you want to view navigation data.

2. Click the Navigation Lab button (). The Launch Navigation Lab dialog box opens (Figure 61-14).

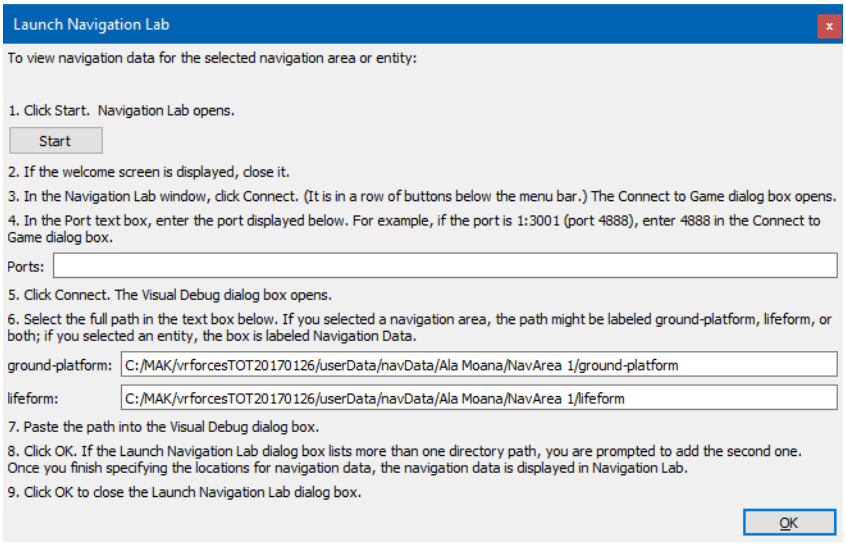


Figure 61-14. Launch Navigation Lab dialog box

3. Follow the directions on the dialog box.

61.6.2. Showing Simulation Object Movement in Navigation Lab

You can view simulation objects in Navigation Lab. This can help you analyze problems in navigation data, such as the inability of an entity to plan a path.

To view simulation objects in Navigation Lab:

1. Connect to Navigation Lab as described in "Opening Navigation Lab from an Information Dialog Box," on page 61-21.
2. Run the scenario.
3. In Navigation Lab, look for volumetric objects that represent the simulation objects (Figure 61-15).

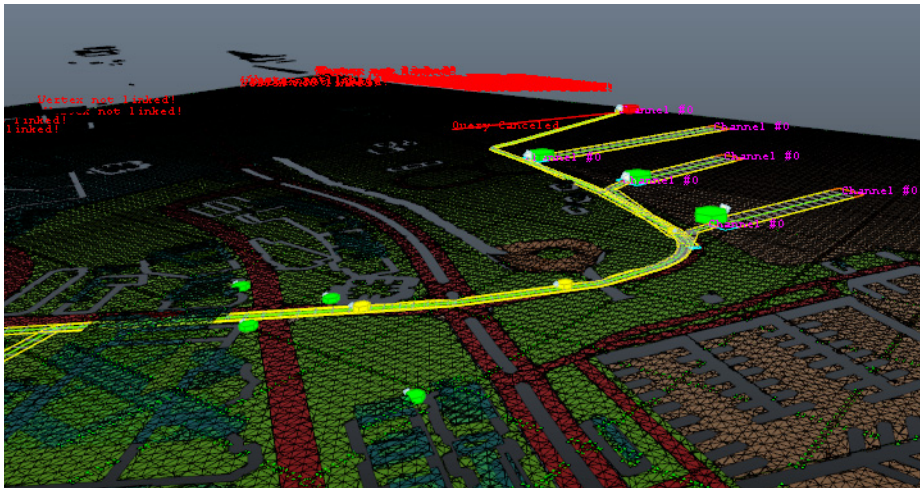


Figure 61-15. Simulation objects represented in Navigation Lab



62. Dynamic Terrain

This chapter describes the dynamic terrain feature.

Introduction to Dynamic Terrain 62-2
Manually Changing the Terrain 62-2
    Resetting Dynamic Terrain Changes 62-3
Removing Dynamic Terrain Damage 62-3
Creating Dynamic Terrain Areas 62-3
Dynamic Terrain Switch Types 62-4

62.1. Introduction to Dynamic Terrain

VR-Forces supports damage to buildings as the result of munition impact. It also supports the ability to open and close doors, windows, and other features. This feature depends on the presence of switch nodes and appropriate model states in the models that can be damaged or changed.

If certain types of terrain damage take place in a navigation area, VR-Forces can automatically regenerate navigation data for the terrain. This lets simulation objects take the damage into account as they move through the terrain. The priority of regeneration and the types of terrain change that trigger it are configured in *./appData/settings/vrfSim/navigationProfiles.mtl*. By default, navigation data is regenerated for damage, and damage_structure switch types. For information about navigation areas and navigation profiles, please see Chapter 61, *Generating Navigation Data*. For information about switch types, please see "Dynamic Terrain Switch Types," on page 62-4.



The speed of navigation data regeneration depends on different factors (most notably sector size, number of affected sectors, and the complexity of the terrain), but in all cases it is not instantaneous. It takes at least several seconds, and possibly considerably longer. It is not abnormal for regeneration to take up to 20-30 seconds even in a well sectorized terrain. Therefore it is normal for entities to continue navigating using the original navigation data for a brief period until navigation data is updated.

For details about how terrain is damaged, please see "Configuring Damage for Dynamic Terrain," on page 77-16.

62.2. Manually Changing the Terrain

Typically, terrain changes occur as the result of munition damage or actions of players in a scenario. For example, a character might open a door so that it can enter a building. However you can manually apply damage states or make other changes.

To manually change the state of a terrain object:

1. Choose **Simulation** → **Dynamic Terrain** → **Add Changes**. The Add Dynamic Terrain Changes dialog box opens. The cursor changes to a green circle that represents the area within which changes can be made.
2. Optionally, change the radius.
3. Click on the terrain or enter the location where you want to make changes. If there are any dynamic terrain models in that area they are listed in the list window.
4. For each option listed that you want to change, select the state you want from the list.
5. Click OK.

62.2.1. Resetting Dynamic Terrain Changes

If you have made changes to the terrain by hand, you can reset those changes. Reset Changes also lets you remove damage created by a munition.

To reset dynamic terrain changes:

1. Choose **Simulation** → **Dynamic Terrain** → **Reset Changes**. The Reset Dynamic Terrain Changes dialog box opens. The cursor changes to a green circle that represents the area within which changes can be made.
2. Optionally, change the radius.
3. Click on the terrain or enter the location where you want to reset changes. If there are any dynamic terrain models in that area they are listed in the list window.
4. For each option listed that you want to change, select Reset in the list.
5. Click OK.

62.3. Removing Dynamic Terrain Damage

You can remove terrain damage caused by munitions or by hand. Remove Damage does not affect changes to doors and windows.

To remove dynamic terrain damage:

1. Choose **Simulation** → **Dynamic Terrain** → **Remove Damage**. The Remove Dynamic Terrain Damage dialog box opens. The cursor changes to a green circle that represents the area within which damage will be removed.
2. Click on the terrain or enter a location in the dialog box.
3. Optionally, specify the radius around the location within which to remove damage.
4. Click OK.

62.4. Creating Dynamic Terrain Areas

If you disable the global dynamic terrain processor when you create a scenario, you can still enable dynamic terrain within dynamic terrain areas. (Global dynamic terrain processor is an option on the Advanced tab of the New Scenario dialog box. For details, please see "Creating a Scenario," on page 7-3.)

Dynamic terrain areas are tactical graphics that are available on the Tactical Graphics Palette. You create them just like any other areal tactical graphic.

62.5. Dynamic Terrain Switch Types

Table 62-1 lists the dynamic terrain switches supported.

Table 62-1: Damage types and values

Type	Allowable Values	Description
damage	none, slight, moderate, destroyed	General terrain damage.
damage_structure	none, slight, moderate, destroyed	Damage specific to houses and buildings.
door	open, closed	The state of a door.
window	open, closed	The state of a window.



63. Processing MetaFlight Files

To display MetaFlight files in VR-Forces, you must first process them. This chapter explains how to do so.

Building Efficient MetaFlight Terrains for VR-Forces 63-2
Processing MetaFlight Files for Use in VR-Forces 63-2
Splitting Datasets into Individual MetaFlight Files 63-3
Converting Virtual Texture Datasets to Tiled Textures 63-4
The mftTool 63-4
Using the mftTool to Process MetaFlight Files 63-6
Convert Geometry Grid Datasets with Tiled Textures 63-7
Convert Source Data into MEDF Format 63-7
Create an MTF File for the MetaFlight Terrain 63-8

63.1. Building Efficient MetaFlight Terrains for VR-Forces

VR-Forces can only load MetaFlight terrains built with certain tools. All MetaFlight files must be preprocessed by VR-Forces tools before they can be loaded. The following are recommendations for constructing MetaFlight terrains that can be processed and loaded into VR-Forces:

- The MetaFlight terrain can contain one or more Geometry Grid Datasets (GGDS).
- The MetaFlight terrain can contain one or more Virtual Texture Datasets.
- The MetaFlight terrain can contain one or more Cultural Feature Datasets.
- The MetaFlight terrain can be built in the Geodetic, Geocentric, UTM, or Flat Earth coordinate systems.
- For the best conversion of Virtual Textures into a tiled texture GGDS, match every Virtual Texture layer with a geometry layer.
- For optimal performance the GGDS should be set up in a Pyramid structure (Level 1: 1x1, Level 2: 2x2, Level 3: 4x4, and so on).
- For optimal performance levels 1-4 should be made of FILE LOD GEOM tiles only. Do not put any children LOD GEOM tiles in these levels.
- For optimal performance the terrain should be balanced so that any single FILE LOD GEOM is not too large and can be paged in real-time (based on machine speed and video card). In other words, keep any single file small.
- For optimal performance any single FILE LOD GEOM should not contain too many LOD GEOM nodes as children.
- Geometry Grid Tile size (in meters) should match Virtual Texture Tile coverage (in meters).
- For optimal performance limit the number of external references. This reduces cull times. (For example, instead of having 10 individual buildings, externally reference one city block with all 10 buildings.)

63.2. Processing MetaFlight Files for Use in VR-Forces

Before you can load a MetaFlight file in VR-Forces, you must process it as described in this chapter. If you try to load an unprocessed MetaFlight file, VR-Forces displays an error message. If you run the MAK processing tools on a MetaFlight database that does not conform, an error message is sent to the console window.

To process a MetaFlight file, you must:

1. Convert source data into MEDF and MEIF format.
2. Split each dataset into its own MetaFlight file.
3. Convert geometry from using virtual texture datasets into using tiled textures. (If the terrain does not use virtual textures, you can skip this step.)
4. Use the VR-Forces terrain composition process to combine the MetaFlight terrain files and save them in MTF format.



Geometry files referenced by a MetaFlight file must be in OpenFlight format, and not in Vega Prime's binary format (*.vsb*). MAK's terrain libraries do not support VSB files.

63.3. Splitting Datasets into Individual MetaFlight Files

MAK tools work on one Geometry Grid Dataset (GGDS) at a time. Some MetaFlight files contain multiple GGDSs, so each one has to be moved into its own file. It is your responsibility to understand the syntax of the MetaFlight file and to know whether or not you need to split up datasets.

To split datasets:

1. Create a copy of the original file for each dataset. Give each a name that identifies the dataset it contains.
2. Open each copy of the original file in a text editor.
3. In each file, delete all of the <GeometryGridDataset> XML element blocks except the one for the dataset you want to have in this file.

When you are finished, each MetaFlight file will have only one GGDS and possibly one or more Virtual Texture Datasets (VTDS).

63.4. Converting Virtual Texture Datasets to Tiled Textures

The display engine does not handle virtual textures at runtime. Therefore you must convert the virtual texture datasets into tiled textures.

63.4.1. The mftTool

The mftTool is a command-line tool for converting virtual texture datasets into tiled textures so that they can be loaded by VR-Forces. It creates the highest resolution textures possible from the virtual texture dataset and binds them to individual tiles.

The syntax for mftTool is as follows:

```
mftTool -f filename [-c index -w index -l index -m string
   -r path -x size -s directory -t filename -A -g -e -u
   -p threads --extendLevelZero distance -v -h --]
```

Table 63-1 describes the command-line options.

Table 63-1: mftTool command-line options

Option	Description
(-- | --ignore_rest)	Ignore all arguments after this one.
(-A | --regenerateAll)	Forces the regeneration of all tiled textures. This can be much slower and is needed less frequently if the virtual texture dataset does not change.
(-c | --col) *index*	The column to convert, using a zero-based index.
(-e | --encryptImages)	The tiled textures are generated in the MEIF file format. If not specified, the file format will be DDS.
--extendLevelZero *distance*	Specifies the page-in range (in meters) of grid level zero in the MEDF and MetaFlight files that the mftTool creates. Use this option if the terrain disappears when you zoom out, particularly in Plan View mode. Increasing the range causes the terrain to page in at a greater distance.
(-f | --file) *filename*	The name of the MetaFlight file that contains the GGDS.

Table 63-1: mftTool command-line options

Option	Description
(-g | --generateFiles)	Generate tiled textures from the virtual texture and generate tiles in MEDF format from the FLT tiles that use the new textures. An MFT file is published. The output name will be equal to the MetaFlight file specified with the -f option with the string _medf appended to the end before the extension. For example, converting a file named *yourTerrain.mft* creates a new MetaFlight file named *yourTerrain_medf.mft* in the same directory as *yourTerrain.mft*. If the generated tiled texture already exists, the file is not regenerated. The -u and -g arguments are mutually exclusive.
(-h | --help)	Displays usage information.
(-l | --level) *index*	The level to convert, using a zero-based index.
(-m | --missing) *string*	The missing texture placeholder.
(-p | --threads) *threads*	Specifies the number of threads to run. Specifying zero results in the number of threads equaling the number of logical processors the host machine has.
(-r | --searchDir) *directory*	Adds a directory to the search path when resolving external references. During processing any external references not found are removed from the generated files.
(-s | --subDir) *directory*	The path to the substitution texture specified by the VTDS. The arguments -s and -t are mutually exclusive.
(-t | --vtFile) *file*	The name of the MetaFlight file that contains the VTDS. The arguments -s and -t are mutually exclusive.
(-u | --publishMftOnly)	Publish a new MetaFlight file that can be loaded by VR-Forces. The output name will be equal to the MetaFlight file specified with the -f option with the string _medf appended to the end before the extension. Converting a file named *yourTerrain.mft* creates a new MetaFlight file named *yourTerrain_medf.mft* in the same directory as *yourTerrain.mft*. The -u and -g arguments are mutually exclusive.
(-v | --version)	Display version information and exit.

Table 63-1: mftTool command-line options

Option	Description
(-w | --row) *index*	The row to convert, using a zero-based index.
(-x | --textureSize) *size*	Specifies the maximum texture size to generate in pixels per dimension. Larger textures use more memory and are slower to page in, but allow for greater virtual texture resolution.

If neither the generate (-g) or the publish only (-u) argument is specified, mftTool prints out information regarding the MetaFlight terrain.

63.4.2. Using the mftTool to Process MetaFlight Files

To process a terrain with a single MetaFlight terrain file that has one GGDS and one VTDS, run the mftTool as follows:

```
mftTool -f C:\yourTerrain\yourTerrain.mft
   -s C:\yourTerrain\ds200-vt-otw-virtual-texture -g -A -e
```

The following example shows how to process a terrain with two MetaFlight terrain files, one with the GGDS and one with the VTDS. In this case, the tiles also contain external references to other OpenFlight models.

```
mftTool
   -f C:\yourTerrain\ds100-flight-terrain\
   ds100-flight.mft
   -t C:\yourTerrain\ds200-vt-otw-virtual-texture\
   ds200-vt.mft
   -r C:\yourTerrain\textures
   -r C:\yourTerrain\objects
   -m C:\yourTerrain\MissingTexture.png -x 4096 -g
```

This creates a new MetaFlight file that references the new geometry grid dataset with virtual textures bound to it.

63.4.3. Convert Geometry Grid Datasets with Tiled Textures

The mftTool only needs to publish the MFT when the GGDS uses tiled textures. Cultural feature datasets are often built this way. These can be converted using the medfTool like all other files. The command syntax for the medfTool is described in "Compressing Model Files," on page 90-35.

To only publish a new MetaFlight for the cultural dataset, do the following:

```
mftTool -f C:\yourTerrain\flt\ds7_culture_db5_build-
   ing_area_new-hier_cult\ds7_culture_db5_building.mft -u
   -p 0
```

To convert a MetaFlight with tiled textures into a compressed format, use the medfTool, as follows:

```
medfTool -s C:\yourTerrain\flt\textures
   -s C:\yourTerrain\flt\objects
   --directory C:\yourTerrain\flt\
   ds7_culture_db5_building_area_new-hier_cult -x flt
   -m rgb -m rgba -m jpg -m bmp -p 0 -z 1
```

! The medfTool should never be run on a directory that contains GGDS tiles that use Virtual Textures.

63.5. Convert Source Data into MEDF Format

Many file formats, including OpenFlight files, are often slow to load, so it is best to convert them to the fast loading MEDF (MAK Encrypted Data Format) format. Do this for terrains that do not have virtual textures (it is done automatically when creating geometry grid datasets from virtual texture datasets) and on cultural feature datasets. This should be done on all data files being loaded, but is required for all external references used by MetaFlight terrains.

! You must convert the files to MEDF and MEIF before you run the mftTool.

To convert external references in separate directories:

```
medfTool -s C:\yourTerrain\flt\textures --directory
   C:\yourTerrain\flt\objects -x flt -m rgb -m rgba -m jpg
   -m bmp -p 0 -z 1
```

To convert external reference textures that are in the same directory:

```
medfTool --directory C:\yourTerrain\flt\textures -m rgb
   -m rgba -m jpg -m bmp -p 0 -z 1
```

63.6. Create an MTF File for the MetaFlight Terrain

Once you have processed all of the MetaFlight files, create an MTF file for the terrain.



Any missing models or white objects usually indicate missing external references. Fixing any missing search paths passed to the tools should resolve most problems.

To create an MTF file for the MetaFlight terrain:

1. Start VR-Forces.
2. Add each MetaFlight terrain as a terrain patch.
3. Save the terrain as an MFT terrain.



64. Terrain Performance and Configuration

This chapter covers miscellaneous issues that affect terrain performance and functioning.

Configuring File Caching 64-2
- Specifying the Location to Cache Terrain Server Data 64-3
- Caching earth File Terrain Data 64-3
- Caching earth File Terrain Data Offline 64-5
- Enabling Texture Compression 64-6
- Clearing the File Cache 64-6

Configuring Paged and Streaming Terrains 64-7

Debugging Earth Files 64-7

Using Shader-based Effect Maps 64-10
- Composite Effects Maps 64-11
- Debugging Shaders 64-12
- Reloading Shaders 64-13

Indirect Rendering 64-14
- Scaling the LOD Distance for Indirect Rendering 64-15
- Setting the LOD Transition Percentage 64-15

Displaying Terrain Intersection Lines 64-16

Displaying DDS Textures Correctly 64-17
- Flipping DDS Textures Globally 64-17

Enabling Occlusion Queries 64-19

Converting OpenFlight File Texture Data 64-20

Drawing Bounding Volumes for Scene Graph Nodes 64-22

Simulation Terrain Debugging 64-23

64.1. Configuring File Caching

When file caching is enabled, VR-Forces stores information about the terrain and models in a format that enables it to be loaded quickly. This improves loading time the next time the cached objects are loaded. If caching is disabled, terrains and models must be loaded from their native formats, which may take longer.



When VR-Forces processes a request to load a file, such as an external reference in a terrain, it checks the cache and the directory of the requested file for an MEDF version of the file. If it does not exist, it loads the file in the original format.

To enable or disable file caching:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 64-1).

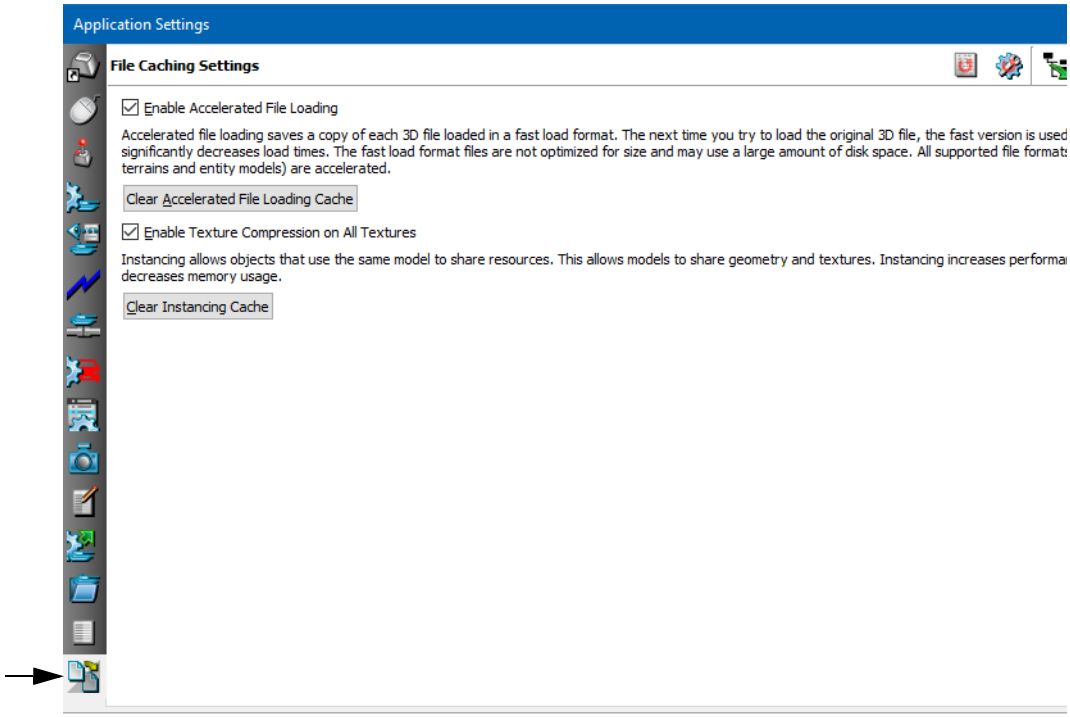


Figure 64-1. File Caching Settings page

3. Select or clear the Enable Accelerated File Loading check box.

64.1.1. Specifying the Location to Cache Terrain Server Data

When file caching is enabled, VR-Forces caches data from terrain servers in the default locations. You can override this by specifying alternative caching locations using the OSGEARTH_CACHE_PATH and VRFSIM_OSGEARTH_-CACHE_PATH environment variables. These environment variables interact as follows:

- If you just set OSGEARTH_CACHE_PATH, then the front-end and back-end both use it. It overrides the `cache` element in the earth file.
- If you set OSGEARTH_CACHE_PATH and VRFSIM_OSGEARTH_-CACHE_PATH then the front-end uses OSGEARTH_CACHE_PATH and the back-end uses VRFSIM_OSGEARTH_CACHE_PATH.
- If you just set VRFSIM_OSGEARTH_CACHE_PATH the back-end uses that value, and the front-end uses the default cache location or the location specified in the earth file.

64.1.2. Caching earth File Terrain Data

You can cache the terrain data in earth files to speed up loading terrain and to allow you to load terrains without connecting to a terrain server, either locally or over the internet. The osgearth_cache tool lets you cache imagery and elevation data offline. (For details, please see "Caching earth File Terrain Data Offline," on page 64-5.) You can also generate cached data through the GUI. The advantage of caching data from the GUI is that you can cache imagery, elevation, and feature data and can use the mouse to specify the area to cache rather than needing to calculate the latitude and longitude, as required by osgearth_-cache.

It may take a very long time to generate the cache for large areas.

To generate cached data:

1. Load a terrain that uses an earth file.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Terrain Contents page.
4. Click the osgEarth Cache button. The osgEarth Cache dialog box opens (Figure 64-2).

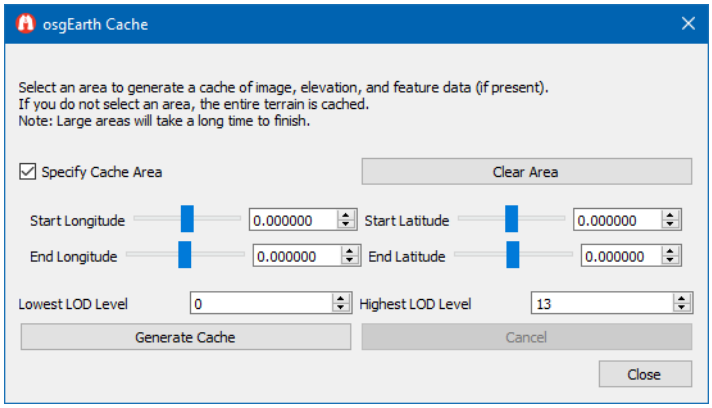


Figure 64-2. osgEarth Cache dialog box

5. If you want to generate cached data for the entire terrain, make sure that the Specify Cache Area check box is cleared.
6. If you want to generate cached data for a portion of the terrain:
   a. Select the Specify Cache Area check box.
   b. Specify the latitude and longitude of a rectangle that defines the area you want to cache. You can do this by adjusting the sliders or entering values for the four values or you can draw a rectangle on the terrain.

      To draw a rectangle on the terrain, click the left mouse button and release it to specify the upper left corner of the rectangle. Drag the mouse to define the cache area. Left-click again to specify the lower right corner.
7. Optionally specify the lowest LOD level and the highest LOD level.
8. Click Generate Cache.

64.1.3. Caching earth File Terrain Data Offline

VR-Forces includes a tool, *./bin64/osgearth_cache*, that you can use to cache terrain data offline. When you need to use the terrain, the data will have already been cached. osgearth_cache caches imagery and elevation. It does not cache feature data. The syntax is as follows:

```
osgearth_cache --list file.earth | --seed file.earth
   [--estimate --min-level level --max-level level
   --bounds xmin ymin xmax ymax* --index shapefile --mp
   --mt --concurrency --verbose]| --purge file.earth
```

where:

- `--list` *`file.earth`* lists information about the cache in an earth file. If the earth file uses template syntax to include other files, you must append *.template* to the name of the file, for example, *myfile.earth.template*. If the file name has spaces, enclose it in parentheses.
- `--seed` *`file.earth`* seeds the cache for an earth file. If the earth file uses template syntax to include other files, you must append *.template* to the name of the file, for example, *myfile.earth.template*. If the file name has spaces, enclose it in parentheses.
- `--estimate` prints out an estimation of the number of tiles, disk space, and time it will take to perform the seed operation.
- `--min-level` *`level`* specifies the lowest LOD level to seed. Default: 0.
- `--max-level` *`level`* specifies the highest LOD level to seed. Default: highest available.

! We strongly recommend that you specify `--max-level` and seed just the areas that you want.

- `--bounds` *`xmin ymin xmax ymax`* specifies the geospatial bounding box to see, in map coordinates. You can include multiple instances of this option. Default: entire map.
- `--index` *`shapefile`* tells osgearth_cache to use the feature extents in the specified shapefile to set the bounding boxes for seeding.
- `--mp` specifies use of multiprocessing to process the tiles. This is useful for GDAL sources because it avoids the global GDAL lock.
- `--mt` specifies use of multithreading to process the tiles.
- `--concurrency` specifies the number of threads or processes to use if `--mp` or `--mt` are used.
- `--verbose` displays the progress of the seed operation.
- `--purge` *`file.earth`* purges a layer cache in an earth file. This command is interactive.

The following example caches a file named *VR-TheWorldOnline.earth*:

```
osgearth_cache.exe --seed ../userData/servers/VR-TheWorld-
  Online.earth --max-level 2
```

The following example caches a portion of a file named *HawaiiLocalServer-mak.earth*:

```
osgearth_cache.exe --seed ..\userData\servers\Local World
  Ala Moana.earth --max-level 18 --bounds
  -162.0013879999999915 14.9985579999999992
  -149.9985580000000027 25.0013879999999986
```

If you preprocess a CDB terrain, you must copy all of the directories created for the terrain from *./appData/cache* to *./appData/cache/vrfsim*.

64.1.4. Enabling Texture Compression

When texture compression is enabled, textures are converted internally to a compressed format (DXT). Compressing textures decreases memory use and increases rendering performance.



The textures provided with VR-Forces are already compressed. This setting is useful if you are adding your own data.

To enable or disable texture compression:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 64-1).
3. Select or clear the Enable Texture Compression On All Textures check box.

64.1.5. Clearing the File Cache

You can clear the file cache if you need to free disk space.

The cache is normally configured to be in *./appData/cache*. You can delete files from this directory if you want to. The most likely reason to do this is if you load a terrain and realize that it should have had its DDS textures flipped, but has now been saved with them oriented the wrong way. You could delete the cache, then reload the terrain and toggle DDS Textures to the opposite setting.

To clear the file cache:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 64-1).
3. Click Clear Accelerated File Loading Cache.

64.2. Configuring Paged and Streaming Terrains

You can improve performance for some terrains by configuring the following environment variables:

- OSG_MAX_PAGEDLOD. OSG_MAX_PAGEDLOD specifies the target maximum number of paged LODs to maintain. Default: 200. Set it to a lower value if you need to reduce memory consumption when using paged terrains. (This variable only affects the front-end.)
- GDAL_CACHEMAX. GDAL_CACHEMAX specifies the maximum cache size for streaming terrain. The default value is 200 MB, which is often adequate. However, if you are using a highly detailed terrain, increasing the cache size to a larger value, such as 2000, can significantly improve performance.

64.3. Debugging Earth Files

Earth files can contain many layers of elevation, imagery, and feature data. When you are debugging performance problems for a terrain that uses an earth file it can be helpful to turn off individual layers and see how that affects performance. You can do that with the osgEarth Profile dialog box.

In addition to seeing how enabling and disabling a layer affects performance, you change test the effect of changing other properties. The properties you can change for all layers are as follows:

- Visible. Enable or disable display of the layer.
- Opacity. The opacity of the layer. The range of values is 0.0 through 1.0.
- Minimum Visible Range. The minimum camera range at which this layer is visible.
- Maximum Visible Range. The maximum camera range at which this layer is visible.
- Lighting. Enable or disable OpenGL lighting on the model. This setting only applies to model layers.

For ground cover layers, the properties described in the previous list apply. You can also set the following properties for each type of ground cover in a layer:

- Wind Factor. Affects the extent to which some types of ground cover respond to wind. For example, trees that sway in the wind.
- Density. Specifies the number of land cover instances that will render in a given area. Range 0.0 through 1.0.
- Fill. Percentage of land that this layer's instances cover. Lower values result in a patchy appearance.
- Maximum Distance. The maximum viewing distance from the camera.
- Brightness. The level of brightness to apply. Default 1.0. (Some terrains provided with VR-Forces override the default.)
- Contrast. The level of contrast to apply. Default: 0.5.

You can only disable lighting for model layers (features area specified using the `model` element).

- This feature is for use in debugging only. You cannot use it to configure different views of the terrain. Changes are not saved. The earth file is not edited.
- If you want to filter out earth file layers and have the changes persist, use the Earth Layers page. For details, please see "Filtering Earth File Layers," on page 55-10.

To turn earth file layers on and off:

1. Load an earth file.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Terrain Contents page (Figure 64-3).

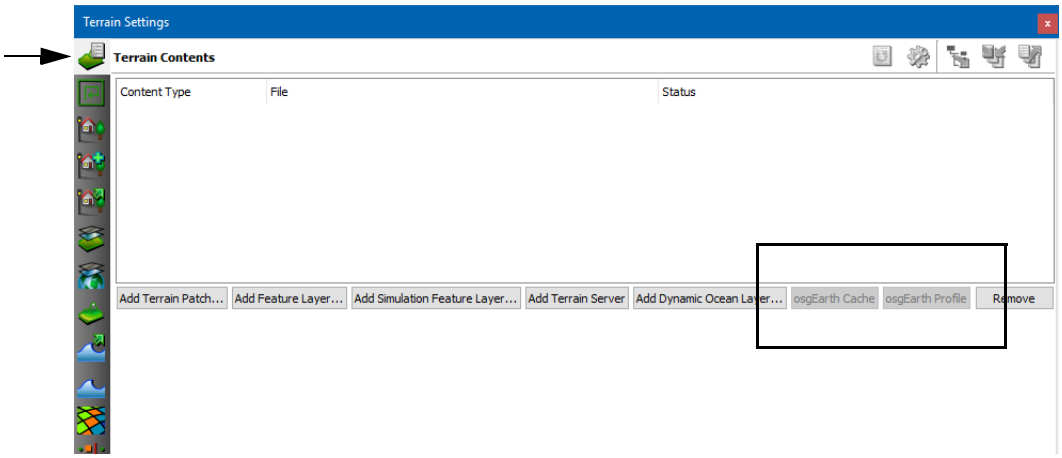


Figure 64-3. Terrain Contents page

4. In the terrain contents window, select the earth file that you want to debug.
5. Click osgEarth Profile. The osgEarth Profile dialog box opens (Figure 64-4).

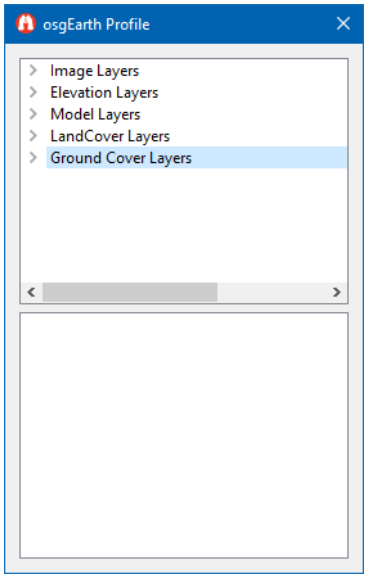


Figure 64-4. osgEarth Profile dialog box

6. Expand the list of the layers that you want to check.
7. Select a layer in the list. The dialog box displays the editable properties for the layer.

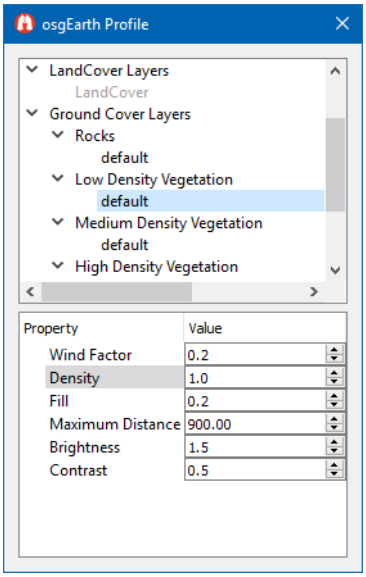


Figure 64-5. osgEarth Profile dialog box

8. Change properties as desired.
9. Repeat this process for each layer that you want to check.

64.4. Using Shader-based Effect Maps

VR-Forces supports several types of shader-based effect texture maps. Shaders are computer programs that run on the graphics processing unit (GPU). VR-Forces makes extensive use of them for lighting effects.

Effect maps are raster images that apply highly realistic textures to the terrain and models. By applying different types of effect maps to terrain and models, you can improve the visual qualities of your simulation without the overhead of high polygon counts.

VR-Forces supports the following types of shader-based effect maps:

- Normal, or bump, maps. 3 Channel. Normal maps give terrains the appearance of relief, such as a rocky landscape.
- Specular maps. 3 Channel. Specular maps affect the highlight color of objects.
- Ambient occlusion maps. 1 Channel. Ambient occlusion maps model areas that do not receive direct light, such as cracks and crevices and shaded areas of terrain and models. These areas are lit only by ambient light.
- Reflection maps. 4 Channel. Reflection maps affect the reflectivity of surfaces, such as windows. Reflection maps reflect objects in the sky, not the terrain.
- Emissive maps. 4 Channel. Emissive maps control the emissivity of whatever they are applied to based on the current ambient light values. The alpha channel is an ID that is set from 0-255. All the pixels that should light together (like a window, and perhaps a surrounding sill) should have the same ID. When the ambient light level drops below a given ID level, those pixels are emissive. Channel 2 (green) is used to indicate the intensity of the emissivity. These textures need to use a lossless compression scheme, such as PNG. *LittlePond.mtf* has sample emissivity textures on many of the houses.
- Nightmaps maps. 3 Channel. Used when ambient light is low. Used in conjunction with emissive maps to control the brightness of pixels.
- Emissive Nightmap maps. 3 Channel. Used when ambient light is low. Makes all pixels glow 100%. An emissive map is not needed, but you cannot control how bright or dark the pixels are.
- Lightmap maps. 3 Channels. Lightmaps store precomputed lighting for surfaces in a scene.
- Gloss maps. 1 Channel. Gloss maps control how wide or narrow the specular highlight appears.
- Reflection maps. 4 Channel. Reflects the environment onto an object.
- Flow maps. 3 Channel. Flow maps define direction-based distortion, such as water flow.

You can include effect maps as part of an OpenFlight model or terrain or as raster maps (on the Raster Maps page of the Terrain Settings dialog box). VR-Forces uses a naming convention for effect map files that lets it process the files properly without any additional configuration. To specify the type of map file, use the following naming conventions:

- Normal map: *filename_NML.png*
- Ambient Occlusion map: *filename_AO.png*
- Specular map: *filename_SPC.png*
- Reflection map: *filename_RFL.png*
- Emissive map: *filename_EMM.png*
- Emissive Nightmap: *filename_NGT.png*
- Lightmap: *filename_LMM.png*
- Gloss maps: *filename_GLS.png*
- Reflection map: *filename_RFL.png*
- Nightmap: *filename_NTE.png*
- Flow map: *filename_FLO.png*



You can use up to seven effect maps per texture. If you need more than that, you must use a composite effects map.

The map code must be uppercase on Linux.

To see the effect of effect maps, you must enable lighting. For details, please see Chapter 45, *Lighting Effects*.

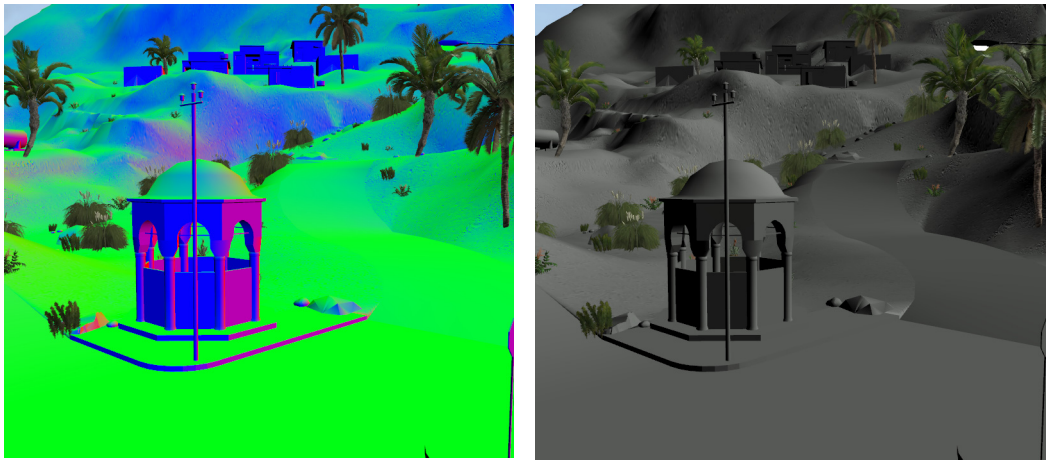
64.4.1. Composite Effects Maps

Effects maps can be composited into a single file using channels. The name of the file represents each effects map in its respective channel, Red, Green or Blue (for example, House_GLS_SPR_AO_REF.png - Glossy, Specular, Ambient Occlusion and Reflection composite map). Only one of the textures can be a composite texture, but all files have to use the Composite Effects Maps naming convention.

- NML: 3 Channel Normal map.
- SPR: 1 Channel Specular map.
- EMS: 1 Channel Emissive map.
- LMM: 3 Channel Lightmap (Supported for CDB).
- GLS: 1 Channel Gloss map.
- AO: 1 Channel Ambient Occlusion map.
- REF: 1 Channel Reflection map.
- NGT: 3 Channel Emissive Nightmap map.
- NTE: 3 Channel Nightmap map.
- FLO: 1 Channel Flow map.

64.4.2. Debugging Shaders

VR-Vantage provides some shader debugging capabilities within its lighting shaders. The shaders are instrumented such that setting different flags (through uniforms) changes the behavior of the shader. Typically it changes fragment colors based on some parameter or parameters. Figure 64-6 shows a view of VR-Village with two different shader debug settings.



Bumpmap Normals | Diffuse Light

Figure 64-6. Shader component visualization

To debug shaders:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 64-7).

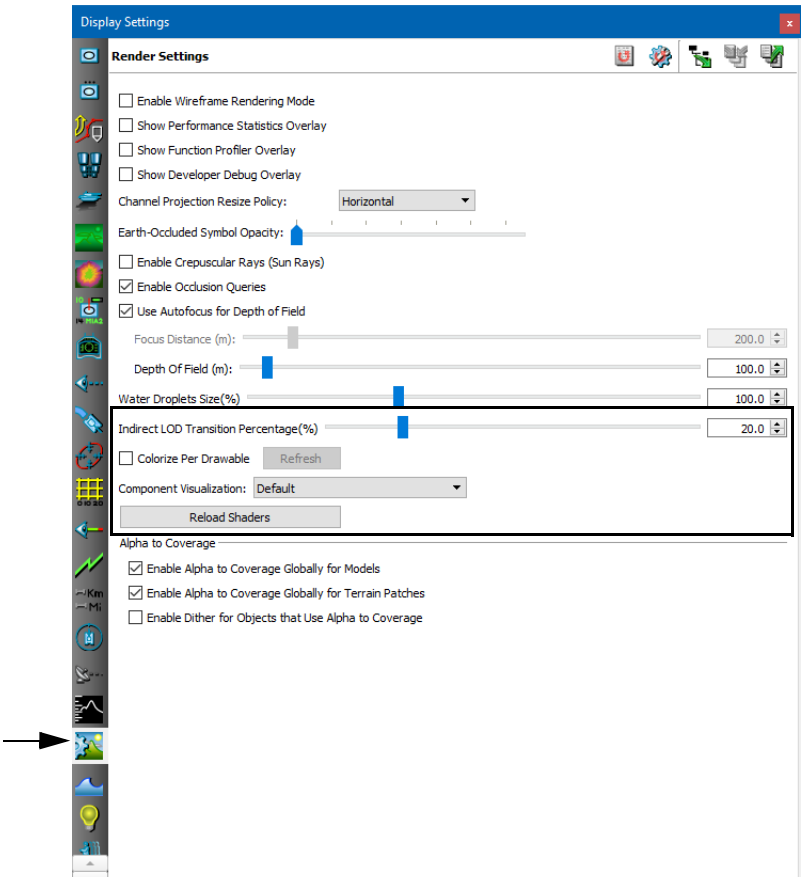


Figure 64-7. Render Settings page

3. Select an option from the Component Visualization list.

64.4.3. Reloading Shaders

Developers who are fine tuning shaders for terrain and object models do so by editing text files that configure the shaders. They can load their new shaders without shutting down VR-Forces. Compile failures are reported in the console. We recommend that you display it while debugging.

To dynamically reload shaders:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 64-7).
3. Click Reload Shaders.

64.5. Indirect Rendering

Indirect rendering is supported for simple OpenFlight terrains loaded as a terrain patch. Indirect rendering improves rendering performance by combining geometry into fewer draw calls. It can combine geometry better than instancing or clustering because it does not require textures to be combined in an atlas or the geometry to have the same vertex data. It is enabled by default for the terrains provided with VR-Vantage that use indirect supported features. For details about enabling, disabling, and configuring indirect rendering, please see:

- "Adding Elevation Data (Terrain Patches) to a Terrain," on page 57-4.
- "Scaling the LOD Distance for Indirect Rendering," on page 64-15.
- "Disabling Indirect Rendering for an OpenFlight Model," on page 90-35.
- "Processing Streamed Data," on page 59-8.
- "Connecting to Terrain Servers," on page 58-2.

Indirect rendering does not effect the rendering of:

- Light Points.
- Light Sources.
- Any nodes below a degree of freedom node.
- Extracted external references.
- Props.
- Speedtrees.
- Triton dynamic ocean.
- Silver-lining sky.
- osgEarth terrain skin.
- Tactical graphics.
- Entities.
- Di-Guy characters.
- TerraPage terrains.
- GDB terrains.

64.5.1. Scaling the LOD Distance for Indirect Rendering

When you use indirect rendering, you can dynamically adjust the distance at which LODs get paged in. If a terrain contains high LODs for objects at distances from the eyepoint at which the details are not visible, you can pay the performance price for the LODs without the visual benefit of high resolution. By scaling the LOD distance, you can cause VR-Vantage to display objects at lower levels of detail without needing to make any changes to the terrain database.

This feature can also help terrain developers to quickly assess the LOD settings for the terrains that they are developing.

To scale the LOD distance for indirectly rendered objects:

1. Choose **Settings** → **Experimental Features**. The Experimental Features dialog box opens (Figure 11-13).
2. Select the Use Indirect Rendering LOD Distance Scale check box. The Scale LOD Distance Calculation in Indirect Rendering slider becomes active.
3. Adjust the slider as desired or specify a value in the spin box.
4. Click Close.

64.5.2. Setting the LOD Transition Percentage

When VR-Vantage transitions between LODs, it blends them so that the change in resolution is not visually jarring. You can set the LOD transition percentage to get the most pleasing effects for your terrain. A setting of zero means there is no transition blending.

To set the LOD transition percentage for indirect rendering:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 64-7).
3. Adjust the Indirect LOD transition percentage slider.

64.6. Displaying Terrain Intersection Lines

You can display terrain intersector lines to help you debug problems with terrain performance.

To display terrain intersection Lines

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Intersector Settings page (Figure 64-8).

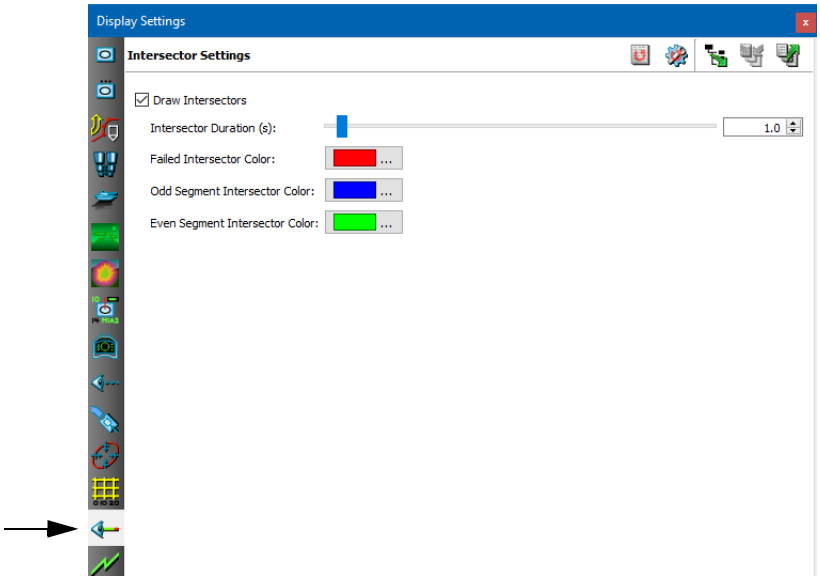


Figure 64-8. Intersector Settings page

3. Select the Draw Intersectors check box.
4. Optionally, set the duration of the intersector lines.
5. Optionally change the colors used for intersector lines.

64.7. Displaying DDS Textures Correctly

DDS is a DirectX bitmap format that is often used for textures. DirectX expects the origin of the image to be at the top left corner. OpenGL expects the image origin to be at the bottom left corner. By default, VR-Forces assumes that images have the OpenGL orientation. However, sometimes images are not oriented properly and therefore you may have to tell VR-Forces to flip them.

You can change the DDS textures setting as follows:

- On a per-model basis. (For details, please see "Flipping DDS Textures for a Model," on page 90-29.)
- When you add a terrain patch. (For details, please see "Adding Elevation Data (Terrain Patches) to a Terrain," on page 57-4.)
- As a global setting. (For details, please see "Flipping DDS Textures Globally," on page 64-17.)

64.7.1. Flipping DDS Textures Globally

You can specify that the DDS textures setting apply to all terrains. This setting is primarily beneficial for paged terrains because you cannot know ahead of time if textures need to be flipped, as you could for a model or terrain patch. If a paged terrain has DDS textures that are upside down relative to VR-Forces's expectations, the textures can be flipped when the terrain gets paged in.

To enable or disable DDS texture flipping for all terrains:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Loader Settings page (Figure 64-9).

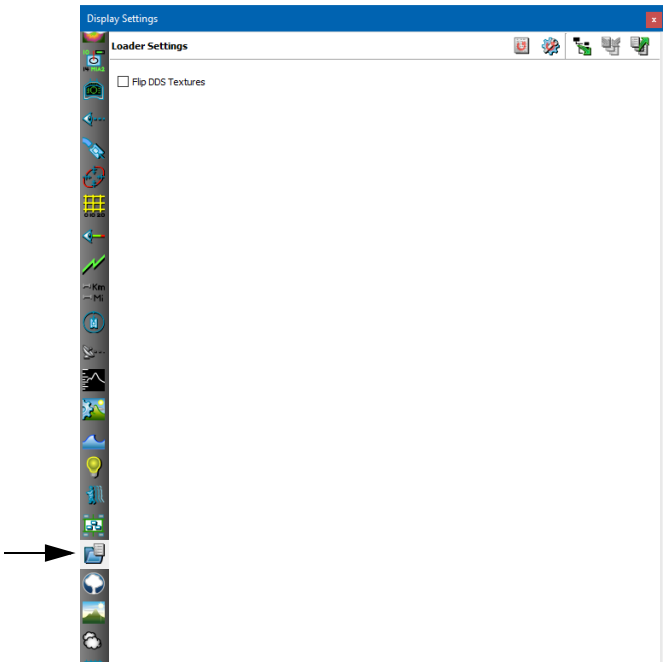


Figure 64-9. Loader Settings page

3. Select the Flip DDS Textures check box.
4. Click Close.

64.8. Enabling Occlusion Queries

If you are simulating on a terrain with many buildings, such as in a city, enabling occlusion queries can improve the frame rate. When they are enabled, VR-Forces draws bounding boxes around some objects and checks to see if they are being rendered or are blocked visually (Figure 64-10). You can view the bounding boxes by enabling occlusion nodes and OC Draw Debug in the Developer Debug Overlay Terrain Patch Object Settings. (For information about Developer Debug Overlay, please see "Using the Developer Debug Overlay," on page 6-27.)



Figure 64-10. Occlusion query bounding boxes

To enable or disable occlusion queries:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 64-7).
3. Select or clear the Enable Occlusion Queries check box.

64.9. Converting OpenFlight File Texture Data

OpenFlight files do not contain information about soil characteristics, which VR-Forces models need to calculate entity mobility.

To provide soil characteristic data for VR-Forces, when VR-Forces loads an OpenFlight terrain database, it tries to deduce soil type information from the names of the texture files that are usually associated with OpenFlight files.

By default, VR-Forces tries to determine the soil types by looking for keywords, such as road, street, grass, water, and so on, in the name of the texture file. If this method does not adequately capture soil type information in the file you are importing, you can map the texture files to soil types in the *surfChar.map* file. This file is specific to each terrain. For an example, see *./data/terrain/Makland/Geometry/OpenFlight/surfChar.map*.

Table 64-1 lists the MAK soil types to which you can map the OpenFlight texture files.

Table 64-1: MAK soil type

asphalt	dryGround	orchards	shallowRiver
boulder	forest	pavedRoad	shallowStream
building	grass	rock	softSoil
cultivatedFields	gravelRoad	sand	swamp
deepLake	mud	shallowLake	tree
deepRiver	ocean	shallowPond	USRailroad
dirtRoad			

The *surfChar.map* file supports two methods of matching textures to soil types, substring match and regular expression. For a substring match, use the `Match` keyword and provide a string to match, as in:

```
Match      HL_water        ocean
```

VR-Forces will match textures with the string `HL_water` to the ocean soil type.

To use a regular expression, use the `RegExprMatch` keyword and provide a regular expression, as in:

```
RegExprMatch ^water.*      ocean
```

VR-Forces will match textures that start with the string `water` to the ocean soil type.

The following is an example *surfChar.map* file:

```
SurfaceCharacteristicMap
{
   Match      HL_water                        ocean
   Match      Water_Ocean_Deep_Blue           ocean
   Match      Tree_Evergreen                  tree
   Match      Tree_Canopy_Evergreen           forest
   Match      Road_Highway_Two-Lane_Asphalt   pavedRoad
   Match      Road_Street_Single-Lane         pavedRoad
}
```

The center column in each entry is an OpenFlight texture. The right column is a MAK soil type.



The list of supported soil types is fixed. Therefore, you cannot add new soil types.

64.10. Drawing Bounding Volumes for Scene Graph Nodes

You can draw bounding volumes for the nodes in the scene graph. This lets a terrain artist or developer visualize the size of the bounding volume for a portion of the terrain so that they know when it will be culled from the scene. It is also useful for finding out if there is something wrong with the size of the bounding volume.

For any given node, the bounding volume drawn is a superset of all the bounding volumes of the children of the node.

Figure 64-11 shows a bounding volume for the terminal area of the airport in Honolulu, Hawaii, USA in the MAK Earth (online) terrain.

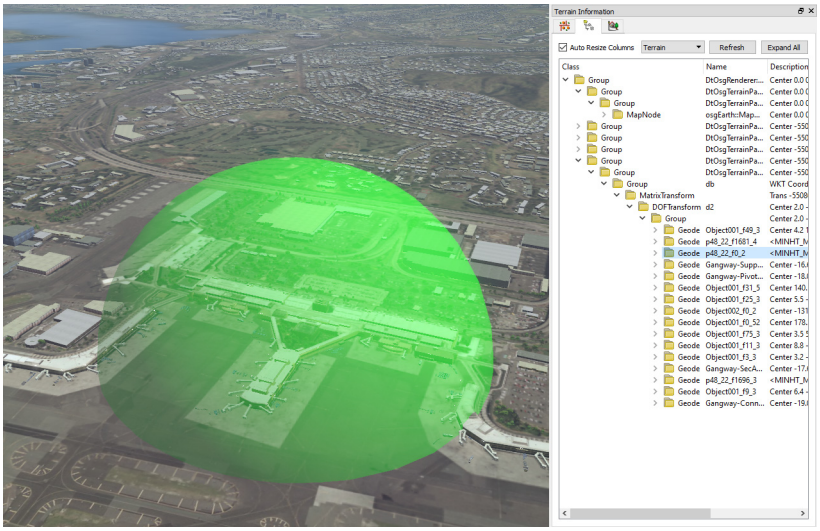


Figure 64-11. Terrain bounding volume

To display terrain bounding volumes:

1. Choose **View** → **Terrain Information Panel**. The Terrain Information Panel opens.
2. Select the Scene Graph tab.
3. In the drop-down list at the top of the tab, select Terrain.
4. Click Refresh.
5. Expand the scene graph to display the node whose bounding volume you want to display.
6. Right-click the node and on the shortcut menu, select **Draw Bounding Volume**.
7. Move the observer until you see the bounding volume.

64.11. Simulation Terrain Debugging

You can display additional terrain paging information by entering commands on the back-end console. You can also display information about feature sets if they are present. Table 64-2 lists the commands and their results.

Table 64-2: Terrain paging information commands

Command	Result
`features config`	Print queries and transforms from *featureconfig.txt*. For details about feature configuration, please see Chapter 65, *Configuring Feature Data*.
`features help`	Prints the syntax of feature commands.
`features info`	Prints generic information about the feature sets, such as the number of paged in features and the coordinate system.
`features list`	Lists all feature sets. The index of each set is used on the commands that take lists of feature sets.
`features pageout`	Pages out all unused feature tiles.
`features print` *`index [number]`*	Prints the feature sets as text. Optionally, specify the maximum number of features to print.
`terrain PSG`	Prints the contents of the scene graph to the console. You can display the same information in the front-end by opening the Scene Graph Nodes Panel.
`pageout`	Pages out all tiles. Tiles that are needed are then paged back in.
`terrain save` *`filename`*	Saves the paged-in tiles to an MTF file. This is useful for quickly and easily saving a portion of a large terrain to a smaller database. The default filename is *simTerrain.mtf*.
`pbv`	Prints page bounding information to the console.
`terrain addlayer` *`layer`*	Displays information in the layer specified by *`layer`*. Developers can write plug-ins to create new layers and add keywords to display them. The timer layer, included with VR-Forces causes the console to display how long it takes to perform terrain intersections. This information can be useful when diagnosing performance issues.

To execute a console command:

1. Open the back-end console.
2. Type the command. You can type commands even if the console is displaying status messages. It remembers what you typed.
3. Press **ENTER**.



65. Configuring Feature Data

This chapter explains how to update the feature data configuration file, *feature-config.txt*, so that the simulation engine can understand feature data.

Introduction 65-2
Specifying Different Feature Names for the Front-End and Back-End 65-2
Named Queries 65-3
Feature Configuration Attributes 65-4
Attribute Transformations 65-4
- Transformations as Aliases 65-4
- Transformations as Derived Attributes 65-5
- Transformations as New Attributes 65-5
- Writing Multiple Transforms to Account for Possible Failures 65-5

Source-Specific Configuration 65-6
- Loading Source-Specific Configuration Files 65-6
- Redefining Queries 65-6

Extracting Feature Data from GDB to Shapefiles 65-7
Converting S-57 Feature Data to Shapefiles 65-7
Configuring Aggregate-Level Movement Restrictions 65-9
Dynamic Features 65-10

65.1. Introduction

VR-Forces can import feature data, such as shapefiles, that provide information about roads, buildings, waterways, and other terrain features. The front-end and back-end handle feature data differently. The front-end must render the data visually. The back-end must understand how features are to be used as it simulates objects on the terrain.

To use feature data correctly for simulation, the VR-Forces simulation engine must know some things about the feature data. For example, when an areal feature is loaded from a feature source, VR-Forces needs to know how to determine whether that feature is a building footprint, lake boundary, forest boundary, or something else entirely.

The simulation engine determines how features should be used by sending queries to the terrain database. The queries that VR-Forces uses are defined in *./appdata/settings/featureconfig.txt*. Feature data is defined using attributes. Attributes have names and values. If the feature data that you are using uses the attributes that VR-Forces knows about, it can process your terrain correctly. If not, you must edit *featureconfig.txt* so that VR-Forces can understand your data. The configuration file is commented. This chapter provides additional documentation about querying feature data.

65.2. Specifying Different Feature Names for the Front-End and Back-End

As described in "Adding Feature Layers to an Earth File," on page 59-9, feature layers are added to an earth file using the `model` element. The front-end uses the `name` attribute of the `model` element to determine what kind of feature layer is being added. In many cases, the back-end can use the same name to determine how to handle the feature data. However, there are some cases where the front-end needs to handle feature data differently from the back-end and needs to use a name that would cause the back-end to process the data incorrectly.

The back-end determines the name of the feature layer based on the following criteria:

- If features come from the server:
  - If the `vrfsim:layer` element is included in the model, the back-end uses the name specified by `vrfsim:layer`.
  - If `vrfsim:layer` is not included, the back-end uses the value in the `name` attribute.
- If the features come from a local file (that is, they use the "ogr" driver):
  - If the `vrfsim:layer` element is included in the model, the back-end uses the name specified by `vrfsim:layer`.
  - If `vrfsim:layer` is not included, the ogr library picks the name.

The supported feature names are specified in *./appdata/settings/featureconfig.txt*.

The following earth file snippet specifies the name “buildings”, which both the front-end and back-end would use:

```
<model name="buildings" driver="some_driver">
  <features> ... </features>
</model>
```

The following earth file snippet specifies the name “special_buildings” and the `vrfsim:layer` “buildings”:

```
<model name="special_buildings" driver="some_driver">
  <features> ... </features>
  <vrfsim:layer>buildings</vrfsim:layer>
</model>
```

In this case, the front-end loads feature data based on the name “special_buildings”. The back-end loads feature data using the name “buildings”.

The following earth file snippet specifies the ogr driver. The driver would choose the feature names.

```
<model name="special_buildings" driver="ogr">
  <features> ... </features>
</model>
```

For more information about the `vrfsim:` syntax, please see “Differential Processing of Elements and Attributes,” on page 59-18.

65.3. Named Queries

Named queries are used throughout the VR-Forces code and configuration to determine what features are. For example, the following named query is evaluated on a feature to determine whether it should be treated as a building:

```
MAK_BUILDING:  (mak_layer="buildings") OR (mak_layer="BuildingsP") OR
  (DTDESCRIPTION = "Building") OR (DTFIDCODE = 181)
```

In this query definition, `mak_layer`, `DTDESCRIPTION`, and `DTFIDCODE` are attribute names. If a feature has one of these attributes with the specified value, VR-Forces knows to treat the feature as a building.

Please see *./appdata/settings/featureconfig.txt* for information about query syntax.



Queries that begin with “MAK_” are referred to either in a simulation model set configuration, or in the VR-Forces code. Do not change the names of these queries.

65.4. Feature Configuration Attributes

When VR-Forces loads a feature in the simulation engine, it adds attributes that indicate where the feature came from. VR-Forces may add one or more of the following features:

- MAK_LAYER. This attribute represents the source of the data – either the name of the file for features loaded from a local file, or the name of the feature layer in an earth file. In the example in "Named Queries," on page 65-3, MAK_LAYER would match a file named *BuildingsP.shp*, for example, or an earth file feature layer named "buildings".
- MAK_EARTH_FILE. If the feature is loaded from an earth file, the file it was loaded from.
- MAK_SOURCE_FILE. The file from which you loaded the terrain. This could be an MTF file, an earth file, or another format that VR-Forces can load directly.

65.5. Attribute Transformations

Attribute transformations allow you to configure attributes on features matching a specific query. These are used in VR-Forces to:

- Create common names for attribute concepts, such as "width", which may be named differently depending on the source.
- Create new attributes whose value is derived from other attributes, which can be modified through configuration instead of in the code.
- Add entirely new attributes to features that match the given query.

65.5.1. Transformations as Aliases

Different sources of feature data often use different names for the same information. For example, a height attribute could be named "height", "hgt", "h", or some other name. You can use attribute transformations to create an "alias" for the different names so that a single attribute name can be used in code and configuration.

For example, the following code defines the attribute `MAK_WIDTH`:

```
Transform ALL: double MAK_WIDTH = [ width | w | wdt | gdb_width | dtwidth
  | wid2d ]
```

This adds, to each feature matching the named query `ALL`, a new double attribute named `MAK_WIDTH`. The value of `MAK_WIDTH` will be the value of the first attribute found in the bracketed list.

65.5.2. Transformations as Derived Attributes

You can use transformations to create derived attributes – that is, attributes whose value depends on the value of another attribute. For example, the following line defines MAK_NUM_LANES, which depends on the width of a road:

```
Transform MAK_ROAD: int MAK_NUM_LANES = MAK_WIDTH / 4
```

This adds a new int attribute named MAK_NUM_LANES to each feature matching the named query MAK_ROAD. The value of this attribute is based on the value of MAK_WIDTH.

65.5.3. Transformations as New Attributes

You can use transformations to add entirely new attributes – for example, to differentiate one way roads from two way roads. In this example, a query for one way road is created, and a new attribute MAK_ONEWAY is added to each feature that matches the query:

```
MAK_ONEWAYROAD: MAK_ROAD AND (ONEWAY = "yes" OR ONEWAY = 2)
Transform MAK_ONEWAYROAD: int MAK_ONEWAY = 2
```

By doing this, we can query the attribute MAK_ONEWAYROAD on any feature, and its existence will mean that the feature represents a one way road.

65.5.4. Writing Multiple Transforms to Account for Possible Failures

You can write multiple transforms for a particular feature if you anticipate that the preferred transform might fail. A transform might fail if it refers to a feature attribute that is not present in the feature data being processed. If you include multiple transforms in the file, they are processed in order until one is successful.

For example consider the following transforms:

```
Transform MAK_S57_DEPTH_AREAS: double MAK_MIN_DEPTH = DRVAL1
Transform MAK_S57_DEPTH_AREAS: double MAK_MIN_DEPTH = 0
```

This transform configures S-57 depth areas. The first transform uses the value of DRVAL1. If the data does not have the DRVAL1 attribute, it sets the depth to 0.

65.6. Source-Specific Configuration

Since different sources of feature data can have different, even contradictory, definitions of the same data, you can create additional configuration files that exist alongside specific data sources.

65.6.1. Loading Source-Specific Configuration Files

Whenever a feature source is loaded, VR-Forces looks for a configuration file that is specific to that source. The filename that it looks for depends on the source of the data. See *featureconfig.txt* for the details of source-specific configuration file loading by source type.

65.6.2. Redefining Queries

Source-specific configuration files can redefine queries in the main configuration file, so the main configuration file can be kept as simple as possible. In the following example, the terrain loaded points to two different *.shp* files, each with a different concept of a main road.

Test.mtf points to two different Shape files:

- *./data/Terrain/Test/Features/RoadL.shp*. Any feature with the attribute "HIGHWAY" set to 1 is a main road.
- *./data/Terrain/Test/Features/MainRoads.shp*. All features are main roads.

Here is the definition of a main road in the main configuration file:

```
MAINROAD: MAK_ROAD AND MAK_NUM_LANES >= 4
```

To specify a different definition of `MAINROAD` for each source, two new configuration files can be created:

- *RoadL.mfc* contains the following definition:
  ```
  MAINROAD: HIGHWAY = 1
  ```
- *MainRoads.mfc* contains the following definition:
  ```
  MAINROAD: ALL
  ```

At runtime, these queries are combined, using the `mak_layer` attribute, to create a single definition of the `MAINROAD` query:

```
OR
  AND
    MAK_ROAD
    MAK_NUM_LANES >= 4
  AND
    MAK_LAYER = "RoadL"
    HIGHWAY = 1
  MAK_LAYER = "MainRoads"
```



In this example it is possible to create this query entirely within *featureconfig.txt*. Here the source-specific configuration is used for readability.

65.7. Extracting Feature Data from GDB to Shapefiles

When you load feature data from a shapefile, it is used more efficiently than if you load it from a GDB file. (For details about loading feature data into a feature layer, please see "Adding a Feature Layer," on page 57-11.) If you have GDB terrains that contain feature data, you can extract that data to shapefiles with the vectorNetworkToShapeFiles utility. The syntax for the utility is as follows:

```
vectorNetworkToShapeFiles.exe terrain_file
   [output_directory]
```

where:

- `terrain_file` is the GDB file to read.
- *`output_directory`* is the directory in which to create the shapefiles. If *`output_directory`* is not specified then output files are named based on the name of the input file.

65.8. Converting S-57 Feature Data to Shapefiles

S-57 feature data does not have a spatial index, so it loads slowly. By contrast, shapefiles have a spatial index, so they load quickly. Therefore, we recommend that you convert S-57 data to shapefiles. VR-Forces includes a command-line tool, ogr2ogr, that you can use to convert your S-57 data. (OGR is a sublibrary of GDAL, which VR-Forces uses to load feature data. Complete documentation is available at http://www.gdal.org/ogr/drv_s57.html and http://www.gdal.org/ogr2ogr.html.)

S-57 files have many layers of data. A shapefile only contains one layer of data. So, to convert your S-57 data, you must create a shapefile for each layer of data that you want to convert. Since VR-Forces does not use all of the data in an S-57 file, this is a simpler process than it might sound.

Before running ogr2ogr, set the following environment variables:

```
S57_CSV = <full_path_to>/appData/gdal_data
```

(The environment variable should specify a full path. If you use a relative path, it is relative to the path from which you run ogr2ogr, not from the VR-Forces root. If you run ogr2ogr from *./appData/gdal_data*, you do not have to set the variable.)

```
OGR_S57_OPTIONS = "RETURN_PRIMITIVES=ON,RETURN_LINK-
   AGES=ON,LNAM_REFS=ON"
```

This setting works for the layers that VR-Forces needs. If you want to convert other layers, you will have to research the proper environment variable settings.

The command syntax is:

```
ogr2ogr -skipfailures [-nlt type] output_file input_file
  layer
```

where:

- `-nlt` *`type`* is the geometry type of the output layer. *`type`* can be NONE, GEOMETRY, POINT, LINESTRING, POLYGON, GEOMETRYCOLLECTION, MULTIPOINT, MULTIPOLYGON or MULTILINESTRING. Add "25D" to the end to make these 3D points.
- *`input_file`* is the input data source (the S-57 file).
- *`output_file`* is the name of the output file (the layer name with *.shp* at the end). All output files should be placed in the same directory. This simplifies configuring the earth file that references them.
- *`layer`* is the name of the layer from the source to convert.

The following example commands convert the layers that VR-Forces needs:

```
ogr2ogr -skipfailures SOUNDG.shp S57FILE SOUNDG
ogr2ogr -skipfailures -nlt POLYGON LNDARE.shp S57FILE
  LNDARE
ogr2ogr -skipfailures DEPARE.shp S57FILE DEPARE
ogr2ogr -skipfailures M_COVR.shp S57FILE M_COVR
```

There will be some output errors for each of these because they have some attributes that cannot be represented in shapefiles (like lists of numbers and strings).

65.9. Configuring Aggregate-Level Movement Restrictions

In aggregate-level scenarios, terrain features can affect movement of simulation objects. This is configured by the relationship of entries in *featureconfig.txt* and a simulation object's movement system.

In *featureconfig.txt*, the terrain mobility configuration section specifies terrain restriction levels, as follows:

```
-- Terrain Mobility Configuration
--
MAK_UNRESTRICTED_TERRAIN:  MAK_ROAD
MAK_RESTRICTED_L1_TERRAIN: NONE
--MAK_RESTRICTED_L2_TERRAIN: mak_layer = "forest"
MAK_RESTRICTED_L2_TERRAIN: NONE
--MAK_IMPASSABLE_TERRAIN:    MAK_OBSTACLE
MAK_IMPASSABLE_TERRAIN:    NONE

MAK_INFANTRY_UNRESTRICTED_TERRAIN:  MAK_UNRESTRICTED_TERRAIN
MAK_INFANTRY_RESTRICTED_L1_TERRAIN: MAK_RESTRICTED_L1_TERRAIN
MAK_INFANTRY_RESTRICTED_L2_TERRAIN: MAK_RESTRICTED_L2_TERRAIN
MAK_INFANTRY_IMPASSABLE_TERRAIN:    MAK_IMPASSABLE_TERRAIN
```

The restriction levels are referenced in the terrain-mobility block in a system definition file. The following code is from *ground-aggregated-movement.sysdef*:

```
(terrain-mobility
  (entry
    (query-name "MAK_UNRESTRICTED_TERRAIN")
    (speed-factor 1)
    (priority 1)
  )
  (entry
    (query-name "MAK_RESTRICTED_L1_TERRAIN")
    (speed-factor 0.75)
    (priority 2)
  )
  (entry
    (query-name "MAK_RESTRICTED_L2_TERRAIN")
    (speed-factor 0.4)
    (priority 3)
  )
  (entry
    (query-name "MAK_IMPASSABLE_TERRAIN")
    (speed factor 0)
    (priority 4)
  )
)
```

65.10. Dynamic Features

VR-Forces uses feature data for path planning, obstacle avoidance, and other interactions between simulation objects and the terrain. In addition to the features that you can import into a terrain using shapefiles, VR-Forces can treat simulation objects and objects such as cultural features, as features and insert them into a feature layer.

For example, in aggregate-level scenarios, users and simulation objects can create combat engineering objects (CEOs), such as minefields, fortified lines, and bridges, that can impede or facilitate unit movement. VR-Forces implements this capability by dynamically treating combat engineering objects as if they were terrain features.

VR-Forces maps entities to features in *./appData/settings/featureconfig.txt*. The following is an example of some mappings:

```
-- Dynamic Features
--

MAK_ROUTE:            {"1:17:0:0:2:-1:-1:-1"}
MAK_AREA:             {"1:18:0:0:1:-1:-1:-1"}
MAK_DYNAMIC_OBSTACLE: {"1:18:0:0:2:-1:-1:-1"}

MAK_VRF_Roads:                     {"1:17:1:0:62:11:0:0"}
MAK_VRF_Bridges:                   {"1:17:1:0:62:12:0:0"}

MAK_VRF_Dynamic_Paths: (MAK_VRF_Roads OR MAK_VRF_Bridges) AND
   Percent_Complete >= *Minimum_Completion_Percentage_For_Effect

MAK_VRF_Engineering_Object_Points: {"1:16:1:-1:62:-1:-1:-1"}
MAK_VRF_Booby_Trap:                {"1:16:1:0:62:1:0:0"}
MAK_VRF_Unexploded_Ordnance:       {"1:16:1:0:62:2:0:0"}
MAK_VRF_Engineering_Object_Lines:  {"1:17:1:-1:62:-1:-1:-1"}
MAK_VRF_Abatis:                    {"1:17:1:0:62:1:0:0"}
MAK_VRF_Anti_Tank_Ditch:           {"1:17:1:0:62:2:0:0"}
MAK_VRF_Barbed_Wire:               {"1:17:1:0:62:3:0:0"}
```



66. Terrain Tutorials

This chapter explains how to create a local terrain server. It also shows how to extract props from a terrain.

Creating an Earth File to Load Terrain Data 66-2
- Add a Map Element 66-2
- Add Elevation Data to the Earth File 66-2
- Add Imagery to the Earth File 66-3
- Add Feature Data to the Earth File 66-3
- Add Options to the Earth File 66-5
- Save the Earth File 66-6
- Load the Earth File and Save is as an MTF File 66-6

Extracting Props from a Terrain 66-8
- Create a New Terrain and Add a Terrain Patch 66-8
- Extract the Props 66-9
- Change the Model Definition for a Prop 66-11

66.1. Creating an Earth File to Load Terrain Data

Earth files are XML files using osgEarth syntax. All the information that VR-Forces needs to load elevation, imagery, and feature data is provided. The data can be stored locally, on a network, or the internet. Data is loaded locally or streamed over the network and paged in as needed.

VR-Forces comes with a terrain *Little Pond (online).earth* that merges local Little Pond data with streaming terrain from VR-TheWorld Server. For this tutorial, we will ignore the rest of the world and focus on Little Pond.

One approach to creating an earth file is to start with one of the earth files provided with VR-Forces, keep the basic structure, and add, edit, or replace elements as needed. For this tutorial, we will start from an empty file and build all the major elements.

66.1.1. Add a Map Element

The root element in an earth file is the `map` element. It encloses all other elements. Open a text editor and add the following code:

```
<map name="Little Pond" type="geocentric" version="2">

</map>
```

66.1.2. Add Elevation Data to the Earth File

We want this terrain to load the elevation data in *./data/Terrain/Little-Pond/Elevation/w072n042.dt1*. Elevation data is configured using an `elevation` element. The location of data in an earth file is provided in a `url` element. We need to use the gdal driver to load elevation data. Add the following text after the opening of the `map` element:

```
<elevation name="Little Pond Elevation" driver="gdal">
  <url>../../data/Terrain/LittlePond/Elevation/w072n042.dt1</url>
  <tile_size>32</tile_size>
</elevation>
```

The file path is relative to the directory that contains the earth file.

66.1.3. Add Imagery to the Earth File

The Little Pond terrain has two image files: a large low-resolution image and a smaller higher resolution image. Image data is configured using `image` elements. If we were to simply add two `image` elements to the earth file, only the last one listed would be loaded. We would only see one of the images.

To use more than one image, we place an `image` element for each image file inside an `image` element that uses a composite driver. List the images from bottommost to topmost. If you list them in the opposite order from the example code, *LittlePond.tif* would be underneath *W72N42.tif* and you would not be able to see it.

Add the following text after the `elevation` block:

```
<image driver="composite">
  <image name="Little Pond Imagery" driver="gdal">
    <url>../../data/Terrain/LittlePond/Imagery/W72N42.tif</url>
  </image>

  <image name="Little Pond Imagery" driver="gdal">
    <url>../../data/Terrain/LittlePond/Imagery/LittlePond.tif</url>
  </image>
</image>
```

We use the gdal driver to load the images.

66.1.4. Add Feature Data to the Earth File

In earth files, you add feature data, such as buildings and vegetation, using a `model` element. You can add as many `model` elements as you need. A `model` element specifies the location of a shapefile. Then it adds `style` elements for each feature type to add from the shapefile. In this tutorial, we will add one kind of house and all of the vegetation in the shapefile.

Add a Building to the Earth File

Add the following code to your earth file after the `image` block. As in the other elements, the `url` element specifies the location of the shapefile. To load buildings, you need to use the feature_geom driver for the `model` element and the ogr driver for the `features` element.

```
<model name="Little Pond Buildings" driver="feature_geom">
  <features name="Buildings" driver="ogr">
    <url>../../data/Terrain/LittlePond/Features/BuildingsP.shp</url>
  </features>
  <layout>
    <tile_size_factor>3</tile_size_factor>
  </layout>
  <styles>
    <style type="text/css">
    HousesColonialGarrison {
    model:
"../../data/Structures/HousesColonialGarrisonClapboardGreen.medf";
    model-heading:       0-[rotation];
    model-scale:        1;
    model-placement: vertex;
```

```
      altitude-clamping:   terrain;
      altitude-resolution: 0.00008583068847656250;
      altitude-offset:     0;
      }
      </style>

      <selector class="HousesColonialGarrison">
        <query>
          <expr>TYPE = 'HousesColonialGarrison'</expr>
        </query>
      </selector>
    </styles>
    <clustering>false</clustering>
    <lighting>true</lighting>
  </model>
```

The `styles` element lets you specify a `style` for each building you want to display. This example shows just one building. Take a look at *Little Pond (online).earth* to see the other buildings that can be configured from this shapefile.

Each style has a name, in this case HousesColonialGarrison. The `model` attribute specifies the model used to display the object. The `selector` element matches the style to an entry in the TYPE column in the database file that accompanies the shapefile (*BuildingsP.dbf*).

To better understand this relationship, open *BuildingsP.dbf* in a spreadsheet application (Figure 66-1). The TYPE column lists the different types of buildings in the shapefile. The `selector` element looks in the TYPE column.

	A	B	C	D	E
1	LENGTH	WIDTH	HEIGHT	ROTATION	TYPE
203	8.21000000000	11.02000000000	9.00000000000	345	HousesColonialGarrison
204	8.21000000000	11.02000000000	9.00000000000	35	HousesColonialGarrison
205	8.21000000000	11.02000000000	9.00000000000	320	HousesColonialGarrison
206	8.21000000000	11.02000000000	9.00000000000	230	HousesColonialGarrison
207	8.21000000000	11.02000000000	9.00000000000	230	HousesColonialGarrison
208	8.21000000000	11.02000000000	9.00000000000	250	HousesColonialGarrison
209	8.21000000000	11.02000000000	9.00000000000	155	HousesColonialGarrison
210	8.21000000000	11.02000000000	9.00000000000	325	HousesColonialGarrison
211	8.21000000000	11.02000000000	9.00000000000	250	HousesColonialGarrison
212	8.21000000000	11.02000000000	9.00000000000	140	HousesColonialGarrison
213	8.21000000000	11.02000000000	9.00000000000	335	HousesColonialGarrison
214	8.21000000000	11.02000000000	9.00000000000	65	HousesColonialGarrison
215	8.21000000000	11.02000000000	9.00000000000	210	HousesColonialGarrison
216	8.21000000000	11.02000000000	9.00000000000	125	HousesColonialGarrison
217	8.21000000000	11.02000000000	9.00000000000	335	HousesColonialGarrison
218	8.21000000000	11.02000000000	9.00000000000	30	HousesColonialGarrison
219	8.21000000000	11.02000000000	9.00000000000	335	HousesColonialGarrison

Figure 66-1. *BuildingsP.shp*

Add Vegetation to the Earth File

The vegetation in VR-Forces uses SpeedTree models. To load SpeedTrees from an earth file, you need to use the feature_custom driver in the `model` element. You use the ogr driver in the `features` element.

Similar to the buildings, the `model` block for vegetation specifies a shapefile and styles. However, unlike with the buildings, we do not specify individual models to load. The marker attribute indicates that VR-Forces should take every entry in the VEG column of *VegetationP.shp*, append ST to it, and look for a model of that name. For example, the column has several entries named TreesBig-LeafMaple. VR-Forces adds ST to form TreesBigLeafMapleST, finds the model definition TreesBigLeafMapleST, and loads the model specified in the model definition.

To add vegetation to the terrain, add the following code to the earth file after the `model` block for the buildings:

```
<model name="Little Pond Vegetation" driver="feature_custom">
  <features name="Vegetation" driver="ogr">
    <url>../../data/Terrain/LittlePond/Features/VegetationP.shp</url>
  </features>
  <layout>
    <tile_size_factor>3</tile_size_factor>
  </layout>
  <styles>
    <style type="text/css">
      Vegetation {
      marker:              [veg] + "ST" ;
      marker-scale:        1;
      marker-placement: vertex;
      altitude-clamping:   terrain;
      altitude-resolution: 0.00008583068847656250;
      altitude-offset:     0;
      }
    </style>
  </styles>
  <clustering>true</clustering>
  <lighting>true</lighting>
</model>
```

66.1.5. Add Options to the Earth File

Finally, add some common options to the file. In particular, this section specifies a location for the cache. Add this block just before the closing of the `map` element:

```
<options>
  <lighting>true</lighting>
  <elevation_interpolation>triangulate</elevation_interpolation>
  <terrain>
    <min_tile_range_factor>3</min_tile_range_factor>
    <skirt_ratio>0.1</skirt_ratio>
  </terrain>
  <cache type="filesystem">
    <path>../../appData/cache</path>
  </cache>
</options>
```

66.1.6. Save the Earth File

The convention for saving earth files is to save them in the *./userData/servers* directory. Save the file as *myLittlePond.earth*.

66.1.7. Load the Earth File and Save is as an MTF File

Your earth file for Little Pond is complete. After you load it, you should save it as an MTF file.

To load the earth file:

1. Start VR-Forces.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Terrain Contents page.
4. Click Add Terrain Server. The Add Terrain Server dialog box opens.
5. In the Servers list, select *myLittlePond.earth*.
6. Optionally, select the Use Indirect Rendering check box. If the check box is selected, then the terrain server will use the high performance indirect rendering system for model layers that are configured to use indirect rendering. (For more information, please see "Adding Elevation Data (Terrain Patches) to a Terrain," on page 57-4.)
7. Click Connect. The terrain loads. It is visible as a small patch on the earth's sphere (Figure 66-2).

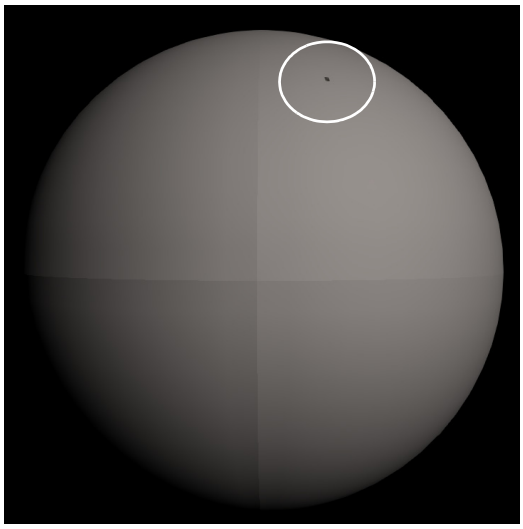


Figure 66-2. Little Pond terrain

8. Zoom in on the terrain (Figure 66-3).

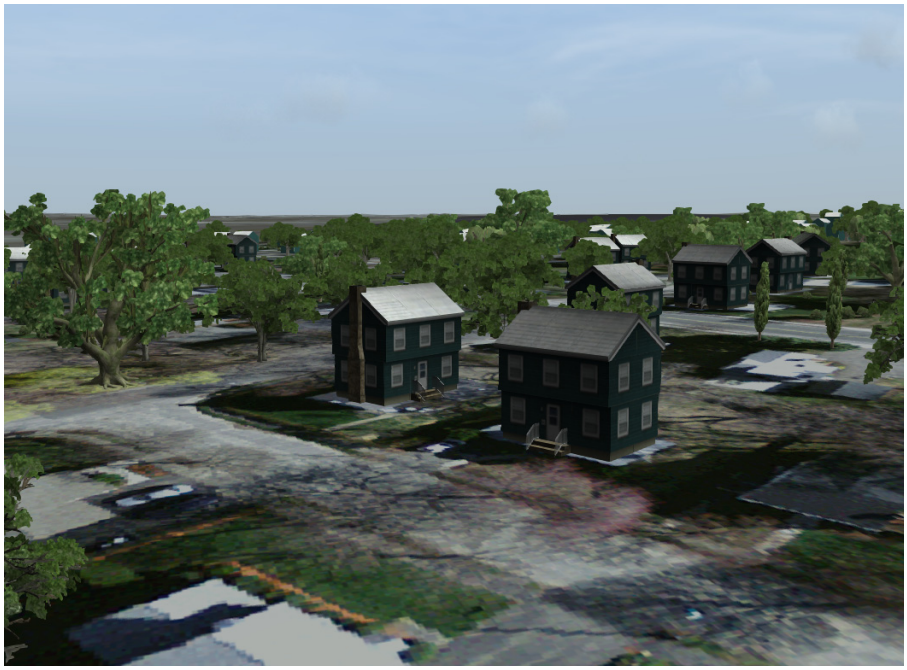


Figure 66-3. Little Pond terrain

9. Choose **File** → **Terrain** → **Save Terrain**. The Save Terrain dialog box opens.
10. Give the terrain a name. Do not use Little Pond unless you want to overwrite the Little Pond terrain provided with VR-Forces.

66.2. Extracting Props from a Terrain

In this tutorial, we show how to create a prop by extracting information from a terrain.

The general steps for extracting props from a terrain are:

1. Create a new terrain with a terrain patch ("Create a New Terrain and Add a Terrain Patch," on page 66-8) or open a terrain that has external references.
2. Select the external references that you want to extract as props and extract them ("Extract the Props," on page 66-9).
3. Optionally, change the model definitions ("Change the Model Definition for a Prop," on page 66-11).

66.2.1. Create a New Terrain and Add a Terrain Patch

To create the new terrain and add a terrain database:

1. Choose **File** → **New Terrain**. If any terrain data is loaded, it is removed.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Terrain Contents page.
4. Click Add Terrain Patch. The Add Terrain Patch dialog box opens.
5. Navigate to the *./data/Terrain/Makland/Geometry/OpenFlight* directory.
6. Select *Makland.flt*.
7. In the Add Terrain Patch dialog box, click OK. The terrain is loaded.

66.2.2. Extract the Props

To extract the props:

1. Move the observer so that you have a view of the plateau in Makland similar to Figure 66-4.



Figure 66-4. Makland

2. Click on the trees or buildings in the scene. Notice that you cannot select them. They are part of the terrain; they do not exist as props.
3. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
4. Select the Extract Props page (Figure 66-5). On the Extract References tab, the External References window lists all of the objects in the terrain that are incorporated into the terrain by reference. These are the objects that you can extract as props. You can extract them individually. For this tutorial, we will extract all of them.

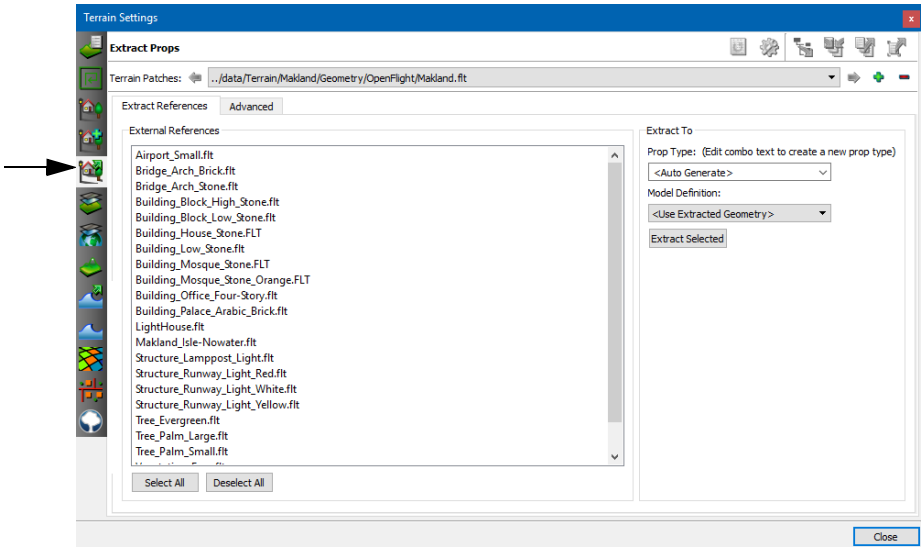


Figure 66-5. Extract props page

5. Click Select All. All of the references are selected.
6. In the Extract To group box, click Extract Selected. The objects are extracted as props. The list of external references is cleared.
7. Click the trees and buildings on the plateau. You can now select them because they are props, not part of the terrain (Figure 66-6). They are displayed using the geometry in the OpenFlight files from which they were referenced. You can change the models that the props use if you want to. For example, you might want to change the trees to SpeedTrees.



Figure 66-6. Selected props

66.2.3. Change the Model Definition for a Prop

To change the model definition for a prop:

1. Select one of the trees.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Edit Existing Props page. The selected prop is highlighted in the list of props (Figure 66-7).

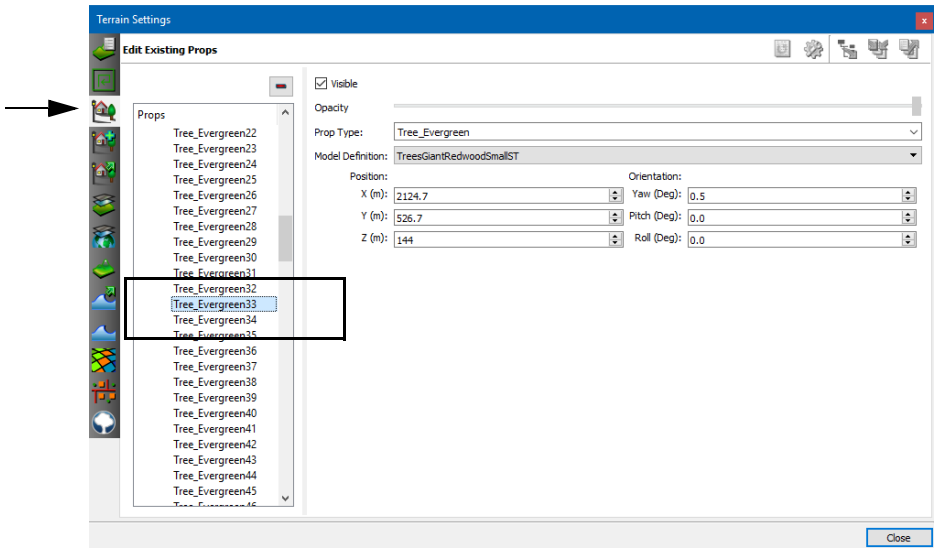


Figure 66-7. Edit Existing Props page

4. In the Model Definition box, select a model definition from the list, such as TreesGiantRedwoodST. The tree changes to the new model definition (Figure 66-8).



Figure 66-8. New model definition for tree

5. To restore the tree to its original model, on the Edit Existing Props page, change the model definition to Use Extracted Geometry.



XII. Creating and Editing Simulation Models

This section explains how to use the Simulation Object Editor to create and edit simulation model sets and the simulation models used by simulation objects. It also covers configuration files that you must edit by hand.



67. The Simulation Object Editor and SMSs

The Simulation Object Editor is a stand-alone tool that lets you configure simulation objects and tactical graphics, and create new simulation model sets.

Introduction to the Simulation Object Editor 67-3
Starting the Simulation Object Editor 67-4
Simulation Model Sets 67-6
- Including Simulation Model Sets in Other Simulation Model Sets 67-7
- Enabling and Disabling Read Only Status for SMSs 67-9
- SMS Priority for Included and Multiple SMSs 67-10
- Lua Scripts in Included SMSs 67-11
- Application Level Files in Included SMSs 67-11
- Search Paths for Referenced Files in an SMS 67-12
- Promoting a Model to the Loaded SMS 67-12
- Promoting a System to the Loaded SMS 67-13
- Hiding Included Objects 67-13

Visual Model Definitions 67-14
- User-Created Visual Definition Files 67-14

Upgrading Old Simulation Model Sets 67-15
- The Old SMS Includes a Standard SMS 67-16
- The Old SMS was Copied from a Standard SMS 67-16
- The Old SMS was Completely New or Copied from base.sms 67-18
- Copying Visual Data 67-18
- Running the SMS Upgrade Tool 67-19
- Migrating an SMS to a New Release by Hand 67-20
- Migrating Pre 4.7 Weapon and Damage Systems 67-21

Opening a Simulation Model Set 67-22
Creating New Simulation Model Sets 67-23
- Creating a New SMS by Including an SMS 67-23
- Creating a New SMS by Copying an SMS 67-24

Importing .entity Files 67-25

Applying New and Updated Variables 67-25
Importing and Exporting Object Information 67-26
Exporting Object Information 67-26
Importing Object Information 67-26
Editing the Tags List 67-27
Adding a New Tag 67-27
Removing a Tag 67-29
Renaming a Tag 67-29
Undoing Tag Changes 67-29
Using Auto Tags 67-30
Assigning Tags Based on Visual Model Characteristics 67-30
Exporting Tags 67-30
Importing Tags 67-31
Managing Tags Using the Object Information File 67-31
Managing the Forces List 67-31
Adding a Force 67-31
Editing a Force 67-33
Removing a Force 67-33
Copying the Object List to the Clipboard 67-33

67.1. Introduction to the Simulation Object Editor

The Simulation Object Editor lets you create and edit simulation model sets and the objects in them.

Many VR-Forces users want to edit some or all of the following simulation object characteristics:

- Weapon systems.
- Basic parameters.
- Sensors.
- 2D icons and 3D models.
- Formations.

Users also want to add new simulation objects or tactical graphics types. The Simulation Object Editor allows you to edit most of the aspects of a simulation object or tactical graphic that a typical VR-Forces user will want to change. You can easily add and remove sensors, weapon systems, and other major component systems, change the 2D and 3D models used to represent them, and edit their properties.

In addition to editing simulation objects, the Simulation Object Editor lets you view information about systems and lets you configure some types of systems. For more information about configuring systems in the Simulation Object Editor, please see the chapters in this manual devoted to configuring different types of systems.



The Simulation Object Editor primarily lets you edit properties that affect simulation. Simulation objects also have visualization models, which are configured in the front-end in the Visual Model Editors and Simulation Object Type Mappings dialog boxes. However, the Simulation Object Editor also lets you configure the 3D models and 2D icons used to display simulation objects. For more information, please see the chapters in Section XIV, *Configuring Visual Models in the Visual Models Editor*.

The Simulation Object Editor does not let you add or remove individual components or edit some of the more detailed entity parameters. For these advanced editing tasks, you must edit specific configuration files, which are described throughout this manual.

67.2. Starting the Simulation Object Editor

➤ To start the Simulation Object Editor in Windows, choose **MAK Technologies** → **Simulation Object Editor**. The Simulation Object Editor opens (Figure 67-1).

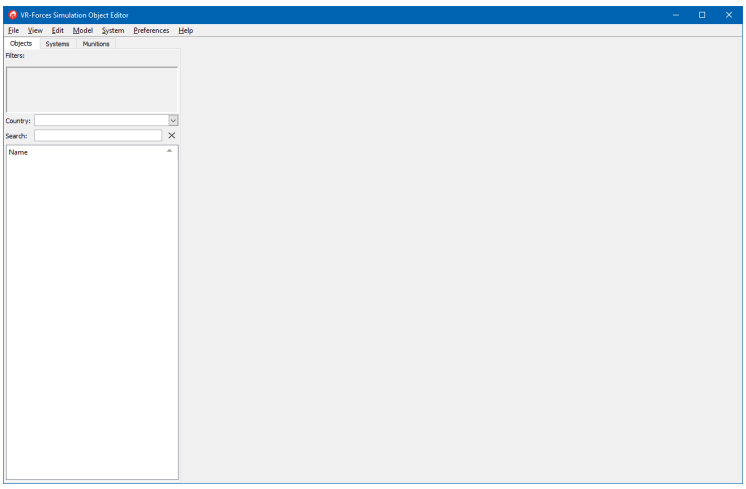


Figure 67-1. Simulation Object Editor

To start the Simulation Object Editor from the command-line in Windows or Linux:

1. Change directory to *./bin64*.
2. Run:

```
simulationObjectEditor options
```

Table 67-1 describes the command-line arguments for the Simulation Object Editor.

Table 67-1: Simulation Object Editor command-line arguments

Argument	Description
`(--` | `--ignore_rest)`	Ignores the rest of the labeled arguments following this flag.
`--appDataDir` *directory*	Specifies the location of application data.
`--dataDir` *directory*	Specifies the location of the data directory (terrains and so on).
`--doNotCheckPluginVersions`	Do not check versions in the plug-ins to make sure they can load against this version of the application.
`--doNotLoadVrfPlugins`	Do not load the plug-ins contained in XML files in the *./plug-ins* directory.
`(-G` | `--locale)` *language*	The name of the language to load into the GUI.
`(-h` | `--help)`	Displays usage information and exits.
`--launcherLocation` *location*	Specifies the location of the VR-Forces launcher.

Table 67-1: Simulation Object Editor command-line arguments

Argument	Description
`--layoutSettingsFile` *`filename`*	Specifies the GUI layout settings file to use. *`filename`* specifies the base filename. It is prefixed with *default_* and the extension *.uisx* is added. Default: SOEUILayoutSettings.
`--loadPlugin` *`filename`*	Load the specified Simulation Object Editor plugin. Can be a specific plug-in or plug-in XML file.
`(-n | --notifyLevel)` *`notification_level`*	Sets the level for messages written to the console or the Simulation Object Editor log file (*simulationObjectEditor.log*), where *`notification_level`* is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2.
`--settingsDirectory` *`string`*	Sets the directory to load all ui and settings files from.
`--showConsole`	Show the console.
`--smsFile` *`string`*	Loads the specified SMS file.
`(-v | --version)`	Displays version information and exits.
`--visualTerrain` *`string`*	Terrain used when displaying simulation objects.

67.3. Simulation Model Sets

A simulation model set (SMS) defines all of the simulation objects, tactical graphics, and other object models that you can create in a VR-Forces simulation. The simulation model set file tells VR-Forces where to find all of the configuration files it needs to create, simulate, and display simulation objects and other objects. Every scenario must reference one or more simulation model sets.

If you use the Simulation Object Editor to edit simulation object models, the only configuration file that you need to be aware of is the simulation model set file that you are using. All of the other files are handled by the Simulation Object Editor.

VR-Forces provides the following SMSs:

- *base.sms*. Defines platforms and graphics used by other SMSs.
- *EntityLevel.sms*. Defines entities, units, and other objects used for entity-level scenarios. Includes *base.sms*.
- *AggregateLevel.sms*. Defines units and other objects used for aggregate-level scenarios. Includes *base.sms*.

For details about entity-level modeling and aggregate-level modeling, please see "Entity-Level Modeling and Aggregate-Level Modeling," on page 13-9.

When you use the Simulation Object Editor, you are editing a simulation model set. We strongly recommend that you keep the default SMSs as reference model sets and not edit them. If you want to edit entity models, create a new SMS that includes one of the provided SMSs. Promote the objects you want to edit to the new SMS and use it as the default SMS for your scenarios. For details about creating a new SMS and changing the default SMS, please see "Creating New Simulation Model Sets," on page 67-23.

67.3.1. Including Simulation Model Sets in Other Simulation Model Sets

VR-Forces simulation model sets can include other SMSs. This allows you to create core SMSs that can be included in multiple other SMSs that may have incompatible characteristics or which override some characteristics of the core SMS, but which still need core objects.

VR-Forces provides three SMSs, *base.sms*, *EntityLevel.sms*, and *AggregateLevel.sms*. *EntityLevel.sms* and *AggregateLevel.sms* both include *base.sms*, which has platform files and graphics that they both need.

It is important to understand the difference between using an SMS that includes other SMSs and creating a scenario that uses multiple SMSs. Since VR-Forces users usually create new SMSs to configure new types of simulation objects, or add new systems to existing objects, we will consider the differences in that context.

If you want to create a custom stand-alone SMS for a new or enhanced object, that SMS must include all of the files required to support that object, such as an OPE file, entity file, system files, damage files, and so on. When trying to build such an SMS, it can be easy to overlook required files. This process also requires you to duplicate files that are contained in the supplied SMSs even if you do not need to change them, because the SMS must contain all the files it needs. Then, if you do not load the SMSs in the correct order, your customization can be overridden by one of the other SMSs you are using in the scenario. In summary, it is a cumbersome process that can create needless duplication of files and be difficult to debug if problems arise.

Using included SMSs minimizes the problems inherent in using multiple stand-alone SMSs. When an SMS includes another SMS, it has a higher priority than the included SMS. When an SMS is loaded, included SMSs are loaded first, and each subsequent SMS has a higher priority than the previously loaded SMS. Any parameters or files in the higher priority SMSs that are also in the lower priority SMSs override those in the lower priority SMSs. The higher priority SMSs do not need to contain any files that they do not change from the included SMSs. Therefore the total footprint of the SMS is smaller. For more information about SMS priority, please see "SMS Priority for Included and Multiple SMSs," on page 67-10.

To include an SMS in another SMS:

1. Choose **MAK Technologies** → **Simulation Object Editor**. The Simulation Object Editor opens (Figure 67-1).
2. Choose **File** → **Open Simulation Model Set**. The Open Model Set dialog box opens.
3. Select the parent SMS.
4. Click Open. The SMS is loaded.
5. Choose **File** → **Properties**. The SMS Properties dialog box opens.

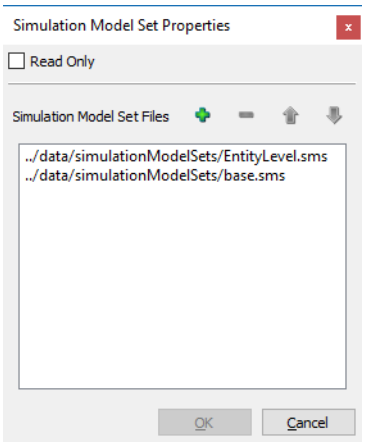


Figure 67-2. SMS Properties dialog box

6. Click the Add button (). A line is added to the window with the text <empty file>.
7. Click the Browse button at the end of the line. The Choose File dialog box opens.
8. Select the SMS that you want to include.
9. Click Open. The SMS is added to the list in the window.
10. Click OK. The SMS is saved and reloaded.

67.3.2. Enabling and Disabling Read Only Status for SMSs

By default, the SMSs provided with VR-Forces are read-only. You can enable and disable editing.



Read only status is to encourage you to create new SMSs for your customizations to the SMSs provided by VR-Forces. Leaving the default SMSs unedited will simplify upgrading to future versions of VR-Forces.

To enable editing of the objects in an SMS, do one of the following:

- Click the Enable Editing button (Figure 67-3). You are prompted to create a new SMS that includes the current one, to enable editing for this and future sessions, or to enable editing for this session only. If the current SMS is one of the three basic SMSs provided by VR-Forces, we recommend that you create a new SMS.

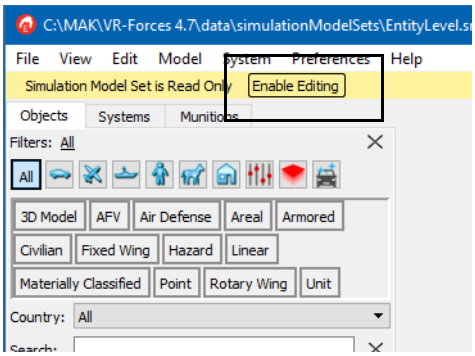


Figure 67-3. Enable Editing

- Choose **File** → **Properties** and clear the Read Only check box in the SMS Properties dialog box.

To enable read only status for an SMS:

1. Choose **File** → **Properties**. The SMS Properties dialog box opens.
2. Select the Read Only check box.

67.3.3. SMS Priority for Included and Multiple SMSs

When you load multiple SMSs either through inclusion or by specifying multiple SMSs for a scenario, conflicts are resolved through the priorities of the SMSs.

For included SMSs, priority is based on the inclusion relationships. The highest level SMS is loaded last and has the highest priority. Simulation objects in higher priority SMSs override those in lower priority SMSs. For example, suppose you create a new SMS called mySMS that includes *EntityLevel.sms*. *EntityLevel.sms* already includes *base.sms*. When VR-Forces loads mySMS it loads *base.sms* first. Then it loads *EntityLevel.sms*, and finally mySMS. Objects in mySMS override objects in *EntityLevel.sms* that have the same object type, which in turn overrides *base.sms*.

When you load multiple SMSs, priority is based on the order in which they are loaded. Assuming a simple case of two SMSs that do not include other SMSs, if the second SMS loaded specifies objects that are also in the first SMS loaded, the specification in the second SMS takes precedence.

If you load multiple SMSs that include other SMSs, priority is based on the order loaded, but duplicate SMSs are not loaded twice. Assume the SMS organization shown in Figure 67-4.

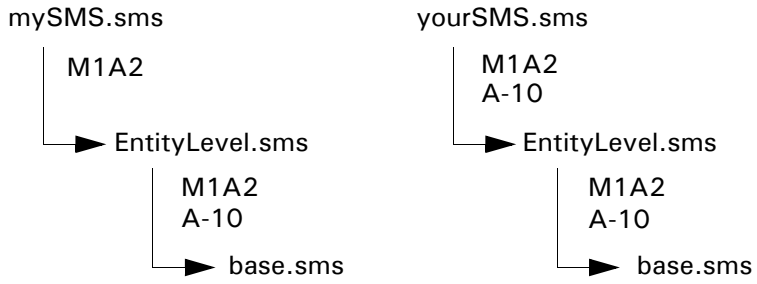


Figure 67-4. Multiple SMSs

If *mySMS.sms* is the first SMS loaded, VR-Forces loads *base.sms*, then *EntityLevel.sms*, then *mySMS.sms*. Priority is determined as described previously for an SMS with included SMSs. The M1A2 entity configured in *mySMS.sms* overrides the one in *EntityLevel.sms*.

Then VR-Forces loads *yourSMS.sms*. It starts with *base.sms*. Since *base.sms* is already loaded, VR-Forces ignores it. Then it moves to *EntityLevel.sms*. Since *EntityLevel.sms* is already loaded, it ignores it. Then it loads *yourSMS.sms*. The M1A2 in *yourSMS.sms* overrides the configuration in *mySMS.sms*. The A-10 in *yourSMS.sms* overrides the A-10 in *EntityLevel.sms*.

The overall priority for the SMSs in this example, from lowest to highest, is *base.sms*, *EntityLevel.sms*, *mySMS.sms*, *yourSMS.sms*.

The priority of multiple SMSs is determined by the order in which they are listed in the Simulation-Model-Set-Files parameter in the scenario file. In the following example, *EntityLevel.sms* has the highest priority. When you create a scenario and list multiple SMSs, the priority in the SMS list is top to bottom.

```
(Simulation-Model-Set-Files "..\data\simulationModelSets\EntityLevel.sms"
  "..\data\simulationModelSets\developer_toolkit_examples\addTask.sms")
```

67.3.4. Lua Scripts in Included SMSs

System scripts are stored in simulation model sets. When you load an SMS that includes another SMS, the scripts in the highest priority SMS supersede those in the lower priority SMSs.

The highest priority scripts get used by all simulation objects, regardless of which SMS they are in. For example, suppose *EntityLevel.sms* has a script called Patrol Area and *mySMS.sms* (the higher priority SMS) also has a script called Patrol Area. If the entity M1A2, which is defined in *EntityLevel.sms* executes the Patrol Area task, it uses the version in *mySMS.sms*.

Lua scripts in included SMSs have the following additional relationships:

- Any scenario scripted task promoted to a system scripted task is promoted into the highest-priority SMS.
- If you edit a scripted task that is in an included SMS, it is saved into the highest-priority SMS. The version of the script in the included SMS does not change. Therefore if you want a change to a script to affect all uses of that script, edit it in its own SMS.

67.3.5. Application Level Files in Included SMSs

Application-level SMS files, such as *commModelParams.mtl* or *gui/initialCreateMenuConfiguration.xml*, are included the same way entity-specific files are and you do not need to have copies of them in a parent SMS as long as one of the child SMSs has them. However every SMS must have *vrfSim.opd*.

67.3.6. Search Paths for Referenced Files in an SMS

The simulation objects in a scenario reference many files to determine their behavior, such as system definition files, scripts, platform files, and so on. If all the simulation objects in a scenario are configured in one SMS, then all of the referenced files are within the file hierarchy of that SMS. However, if you use an SMS that includes other SMSs, then simulation objects may have to search within multiple SMSs to reference the correct files.

An SMS can load files from any SMS it includes, and any currently loaded SMS it is included in.

File referencing, as with other aspects of included SMSs, is based on the priority of the SMS in which an object is configured. Given the SMSs in Figure 67-4, the search paths for each SMS, from lowest to highest priority would be:

- *base.sms*. Can search in *base.sms*, *EntityLevel.sms*, *mySMS.sms*, and *yourSMS.sms*.
- *EntityLevel.sms*. Can search in *base.sms*, *EntityLevel.sms*, *mySMS.sms*, and *yourSMS.sms*.
- *mySMS.sms*. Can search in *base.sms*, *EntityLevel.sms*, and *mySMS.sms*.
- *yourSMS.sms*. Can search in *base.sms*, *EntityLevel.sms*, and *yourSMS.sms*.

67.3.7. Promoting a Model to the Loaded SMS

When you load an SMS that has included other SMSs, the objects from the included SMSs are listed, but you cannot edit them. You can only edit them by editing the SMS to which they belong. Figure 67-5 shows *AggregateLevel.sms* filtered for simulation effect objects. The objects at the end of the list are not editable because they are included from *base.sms*. Note in the figure that most of the parameters for the selected object are read-only.

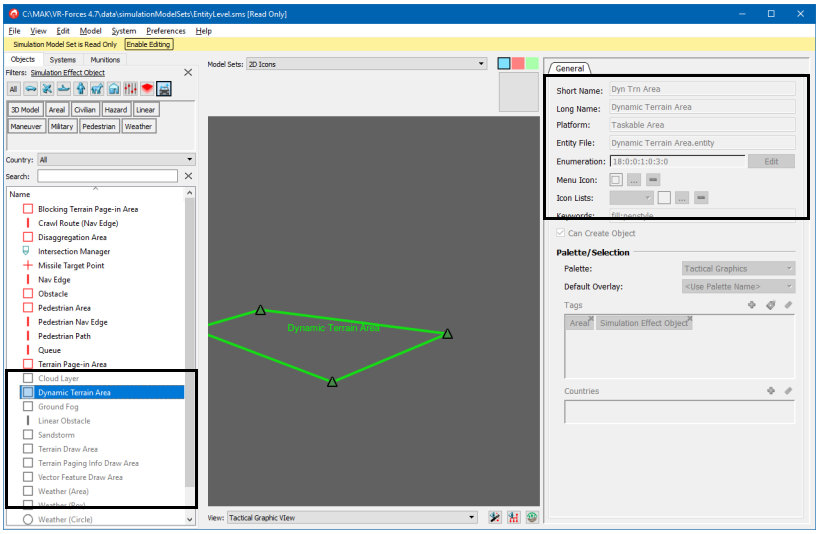


Figure 67-5. Included simulation object

Since one of the purposes of being able to include SMSs in the loaded SMS is to customize simulation objects in the included SMSs, you can promote objects from the included SMS to the loaded SMS. The object then becomes editable and is saved in the loaded SMS. When the SMS is loaded, the object in the loaded SMS overrides the object in the included SMS.

To promote an object to the parent SMS:

1. Load the parent SMS.
2. Choose **Preferences** → **Show Included Simulation Objects**.
3. On the Objects tab or Munitions tab, select the object you want to promote. (Although munitions are on a separate tab, they are also simulation objects.)
4. Choose **Model** → **Promote to Main Simulation Model Set** or click the Promote button (Figure 67-6). The object is added to the parent SMS and is now fully editable. When you save the SMS, a new *.entity* file is added to it.

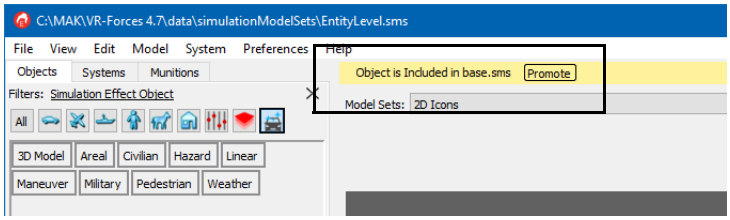


Figure 67-6. Promote button

67.3.8. Promoting a System to the Loaded SMS

Just as with simulation objects ("Promoting a Model to the Loaded SMS," on page 67-12), if you want to edit a system in an included SMS, you must promote it to the loaded SMS.

To promote an object to the parent SMS:

1. Load the parent SMS.
2. Choose **Preferences** → **Show Included Simulation Objects**.
3. Select the system you want to promote.
4. Choose **System** → **Promote to Main Simulation Model Set** or click the Promote button (Figure 67-6). The system is added to the parent SMS and is now fully editable.

67.3.9. Hiding Included Objects

By default, the Simulation Object Editor lists the simulation objects that are part of included SMSs. However, you can hide included objects if you just want to see the objects in the top level SMS.

➤ To hide or display included objects in the objects list, choose **Preferences** → **Show Included Simulation Objects**.

67.4. Visual Model Definitions

VR-Forces stores visual model definitions in different places depending on which SMS is loaded, as follows:

- When the Simulation Object Editor loads *base.sms*, *EntityLevel.sms*, or *AggregateLevel.sms*, it reads the appropriate definition files for the SMS from the directories under *./appData/definitions*. When you save changes to one of these SMSs, the changes are written back to the original definition files.
- If you create a new SMS, model and element definition files get saved with the SMS in the *./data/simulationModelSets/<sms>/gui/visuals* directory for the new SMS. When you load the SMS, it reads model and element definitions from the SMS-specific files.

For information about model and element definition files, please see Chapter 87, *Model Definitions* and Chapter 88, *Element Definitions*.

67.4.1. User-Created Visual Definition Files

Although we recommend that you use the Simulation Object Editor for all changes to visual models, it is possible to add new model definitions and element definitions using the Visual Model Editors dialog box, instead of the Simulation Object Editor. Additionally, some attributes of element definitions cannot be set in the Simulation Object Editor, so if you want to change them, you have to use the Visual Model Editors.

The new and changed definitions can be exported to files other than the default set of definition files. In fact, in many cases, we recommend that you export them to their own files. However, creating new model or element definitions in the Visual Model Editors and saving them to new files has implications for how they are loaded and saved by the Simulation Object Editor.

When the Simulation Object Editor loads one of the three default SMSs, it loads all files in the appropriate directories, including any additional files that you have created. However, when VR-Forces saves changes to the SMS, it writes the changes only to the default definition files for the SMS. It does not write back to the user-created definition files. Since the user-created files are still in the directory, the next time you load the SMS, there may be conflicts between the files and the results may be unexpected.

Given this behavior, the recommended work flow for reconciling new definitions created in the Visual Model Editors dialog box is as follows:

1. In the Visual Model Editors dialog box, save new model or element definitions to new files in the default definition directories.
2. Open the affected SMS in the Simulation Object Editor.
3. Confirm that your new model or element definitions have been loaded by the SMS.
4. Save the SMS. This saves the new definitions into the default set of SMS definition files.

5. Delete the individual definition files that you created in step 1 so that there are no conflicts with the SMS.

If you create a new SMS, when the Simulation Object Editor loads the SMS, it only loads the definition files that it previously created. If you create additional leaf or model definition files with the Visual Model Editors dialog box, they do not get loaded. You must import them into your SMS. Therefore the recommended work flow is as follows:

1. Open a scenario that uses your SMS.
2. In the Visual Model Editors dialog box, load the .hier file for your SMS. It is either *./appData/definitions/Aggregate/ElementDefinitions/coreAggregateElementDefinitions.hier* or *./appData/definitions/Entity/ElementDefinitions/coreEntityElementDefinitions.hier*.
3. Import the model definition or element definition file to which you want to add new definitions. Be sure to import it from the files in the */gui/visuals* directory for your SMS. Be sure that you only edit one element definition file at a time.
4. Add new definitions as desired.
5. Export the new definitions to the definition file that you imported.
6. Open the SMS in the Simulation Object Editor and confirm that the new definitions are available.

67.5. Upgrading Old Simulation Model Sets

The Simulation Object Editor has an upgrade tool that can upgrade SMSs created in VR-Forces 4.4 or later to the current release. The upgrade process varies depending on how you created the SMS that you are upgrading. The possibilities it takes into account are as follows:

- The old SMS was created by including *EntityLevel.sms* or *AggregateLevel.sms* (a standard SMS).
- The old SMS was created by copying *EntityLevel.sms* or *AggregateLevel.sms*.
- The old SMS was a new SMS that did not copy or include *EntityLevel.sms* or *AggregateLevel.sms*. It could have included or be a copy of *base.sms*.

The following sections describe the upgrade process for each of these possibilities. If you need to upgrade an SMS from VR-Forces 4.3.x or earlier, or want to upgrade your SMS manually, please see "Migrating an SMS to a New Release by Hand," on page 67-20.

67.5.1. The Old SMS Includes a Standard SMS

If you created the old SMS by including a standard SMS, the assumption is that you made no changes to the included SMS. You edited your SMS by promoting simulation objects from the standard SMS and editing them, and by adding new simulation objects in your SMS. You may also have made other changes to the SMS, such as adding scripts and adding or editing system definition files.

Since you did not touch the standard SMS, the upgrade tool assumes that you will simply include the standard SMS that is provided in the new version of VR-Forces and that you want to keep all the changes you made when creating the old SMS. The upgrade wizard does the following:

- Copies all files from the old SMS (except visual data as described in the next paragraph).
- If the old SMS is from VR-Forces 4.5 or later and you specify in the upgrade wizard that you want to copy visual data, it is copied to the new SMS. If you specify that you do not want to copy visual data, no visual data files are copied. (For more information about copying visual data, please see "Copying Visual Data," on page 67-18.)
- Compares the files that were promoted into the old SMS to files of the same name in the new included SMS. If the files are different, these files are listed as needing investigation.

67.5.2. The Old SMS was Copied from a Standard SMS

Upgrading an SMS that was created by copying *EntityLevel.sms* or *AggregateLevel.sms* is more complex than upgrading an SMS that was created by including an SMS. The upgrade tool tries to preserve all changes made to the files in the old SMS while incorporating the improvements made to the new version of the standard SMS. However, it errs on the side of keeping changes made to files in the old SMS. This means that if you want your SMS to take advantage of the improvements to the new standard SMS, you may have to compare files by hand. If changes have been made to platform files, you can quickly capture those changes by selecting Apply Platform Changes in the Simulation Object Editor.

The upgrade tool does the following:

- It generates checksums for the files in the old SMS. It compares these checksums to checksums for the files in the standard SMS in the old version of VR-Forces and to the files in the standard SMS in the version of VR-Forces you are upgrading to. It moves files to the new SMS based on these comparisons. Table 67-2 describes the possible results of these comparisons.

Table 67-2: Decision table for comparing SMS files

	Yes	Notes
Is the file the same in old standard SMS, old SMS, and new standard SMS?	Copy file to new SMS.	
Is the file the same in old standard SMS and old SMS, but changed in new standard SMS?	Copy file from new standard SMS to new SMS.	Optionally, run Apply Platform Updates to capture changes in OPE files.
Is the file only in old SMS?	Copy file from old SMS to new SMS.	Optionally, run Apply Platform Updates to capture changes in OPE files.
Is the file in old standard SMS and old SMS, but removed from new standard SMS?	If the file was changed in old SMS, copy it to new SMS. Otherwise, remove it.	Optionally, run Apply Platform Updates to capture changes in OPE files.
Is the file only in new standard SMS?	Add this file to the new SMS.	
Did the file change from the old standard SMS to the old SMS and from the old standard SMS to the new standard SMS?	Copy file from old SMS to new SMS.	Consider merging in changes from the file in the new standard SMS.
Is the file the same in old standard SMS and new standard SMS, but different in old SMS?	Copy file from old SMS to new SMS.	Optionally, run Apply Platform Updates to capture changes in OPE files.

- If the old SMS is from VR-Forces 4.5 or later and you specify in the upgrade wizard that you want to copy visual data, it is copied to the new SMS. If you specify that you do not want to copy visual data, no visual data files are copied. (For more information about copying visual data, please see "Copying Visual Data," on page 67-18.)
- The upgrade tool provides a report of files that have conflicts, files that were added, and files that were removed. You can use this report to determine what files you want to examine for manual upgrades.

67.5.3. The Old SMS was Completely New or Copied from *base.sms*

In rare cases, VR-Forces users may create an SMS by copying or including only *base.sms*, by selectively copying files from one of the standard SMSs, or by creating all the files themselves. In this case, there is no reference for the upgrade process. The upgrade tool does the following:

- Copies all files from the old SMS (except visual data as described in the next paragraph).
- If the old SMS is from VR-Forces 4.5 or later and you specify in the upgrade wizard that you want to copy visual data, it is copied to the new SMS. If you specify that you do not want to copy visual data, no visual data files are copied. (For more information about copying visual data, please see "Copying Visual Data," on page 67-18.)
- Report files that might need to be reviewed, such as system definitions and OPE files.

67.5.4. Copying Visual Data

In VR-Forces 4.4.x, all visual model data was stored in files in *./appData/definitions* and in *./appData/settings/vrfGui*. Visual data was not specific to an SMS. In VR-Forces 4.5 and later, visual data files were restructured and an SMS can have visual data that is specific to the SMS.

For SMSs created in VR-Forces 4.5 or later, the upgrade tool gives you the option of copying visual data from the old version of VR-Forces. If you choose to copy visual data and the old SMS has SMS-specific visual data, the files from the SMS are copied to the new SMS. If the old SMS did not have SMS-specific data, the files from *./appData/definitions* are copied to the new SMS.

If you choose not to copy visual data, the new SMS uses the visual data provided in the new version of VR-Forces.

If you copy visual data to the new SMS, it will take longer to load the SMS when you open a scenario.

67.5.5. Running the SMS Upgrade Tool

To upgrade an SMS to a new version of VR-Forces:

1. Open the Simulation Object Editor.
2. Choose **File** → **Upgrade Old Simulation Model Set**. The Upgrade Simulation Model Set dialog box opens. (The upgrade process varies depending on the type of SMS you are upgrading.)
3. Select the SMS that you are upgrading (the old SMS).
4. Specify a name for the upgraded SMS. The default is the name of the old SMS.
5. Click Next. If the old SMS was created by copying a standard VR-Forces SMS or was a completely original SMS, the Original Simulation Model Set page is displayed. If the old SMS was created by including a standard VR-Forces SMS, the Simulation Model Set Version page is displayed. Skip to step 8.
6. If the old SMS was created by copying *EntityLevel.sms* or *AggregateLevel.sms*, select the standard SMS it was copied from. If the source is completely original or was copied from *base.sms*, select Neither. (SMSs created by including standard VR-Forces SMSs do not see this page.)
7. Click Next. The Simulation Model Set Version page is displayed.
8. Select the version of VR-Forces used to create the old SMS.
9. Click Next. If the old SMS was created with VR-Forces 4.4 or 4.4.2, the upgrade process runs and a report is generated. If the old SMS was created with VR-Forces 4.5, the Visual Data page is displayed.
10. Specify whether or not to copy visual data.
11. Click Next. The upgrade process runs and a report is generated.

67.5.6. Migrating an SMS to a New Release by Hand

If you need to migrate an SMS that was created in VR-Forces 4.3.x or earlier or do not want to use the upgrade tool, you can migrate your old SMS by hand. You can do this by copying your customized files to a new SMS or by recreating your changes in a new SMS.

When you edit an SMS in the Simulation Object Editor, it:

- Creates or edits *.entity* files for the simulation objects that you edit.
- Modifies visual definition files and entity type mappings.

The Simulation Object Editor does not create or change any other files. However, there are many other files in an SMS that could be customized, such as system definitions, the various damage and ammo select files, and system scripts.

If you upgrade to a new version of , you can quickly add any of the new simulation objects you created by copying their *.entity* files into an SMS in the new release. This will make the simulation objects available to the simulation engine, but will not include their visual definitions. For each entity that you add this way, you will have to edit the 2D and 3D models in the Simulation Object Editor.

If you made any changes to other files, you may have to copy those changed files to the new SMS.

If you have written scripts, use the script export process to export them from the previous release of and then import them into the new release.

Since SMSs often have significant changes from one release to the next, the older your legacy SMS is, the harder it will be to upgrade it simply by copying changed files. We recommend that you review the release notes for interim releases to understand the changes that you might need to account for when migrating old files. You can view the release notes for each release on the VR-Forces Product Versions page at https://www.mak.com/support/product-versions/simulate/vr-forces.

There are several major SMS divergence points to be aware of:

- In VR-Forces 4.2, *default.sms* was reorganized as *base.sms* and *EntityLevel.sms*.
- In VR-Forces 4.5, visual model files were restructured.
- In VR-Forces 4.7, weapon system and damage system files were changed significantly so that they could be edited in the Simulation Object Editor.

67.5.7. Migrating Pre 4.7 Weapon and Damage Systems

Due to changes to the system definition files to support editing weapon and damage systems in the SOE, the SMS upgrade tool does not migrate weapon and damage systems from the VR-Forces 4.6.1 and earlier format to VR-Forces 4.7 or later. If you have modified a system using the legacy file formats and want to replicate it in the current release, you have two options, as follows:

- Copy the old system files to the new version of VR-Forces. VR-Forces continues to support the VR-Forces 4.6.1 and earlier format. If you copy the old files to the current SMS, you will not be able to edit them in the SOE.
- Use the SOE to edit an existing system or create a new system to match the system that you customized in the older version of VR-Forces.

67.6. Opening a Simulation Model Set

To edit simulation object models in the Simulation Object Editor, you must open a simulation model set.

To open a simulation model set:

1. Choose **File** → **Open Simulation Model Set**. The Open Model Set dialog box opens.
2. Select the SMS you want to open.
3. Click Open. The SMS is loaded and a list of simulation objects is displayed in the Simulation Object Editor.

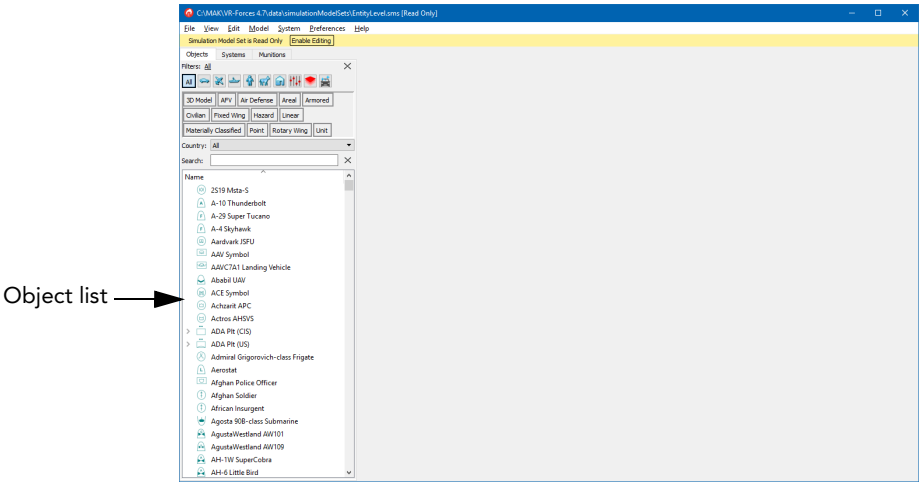


Figure 67-7. Simulation Object Editor with simulation model set loaded

67.7. Creating New Simulation Model Sets

You can create a new SMS by copying an existing SMS or you can create a new SMS that contains no simulation objects, but includes an existing SMS. In the first case, you would edit the simulation objects already in the SMS and add new ones as desired. In the second case, you would add new objects or promote objects from the included SMSs into the parent SMS.

To simplify upgrading to future versions of VR-Forces, the recommended choice is to create a new simulation model set by including an existing SMS.

67.7.1. Creating a New SMS by Including an SMS

Every SMS must include a basic set of files, such as platform files, GUI configuration files, and system definitions. You ensure that your new SMS has those files by including an existing SMS in it. The minimum set of files is provided by *base.sms*.

To create a new simulation model set:

1. Choose **File** → **New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 67-8).

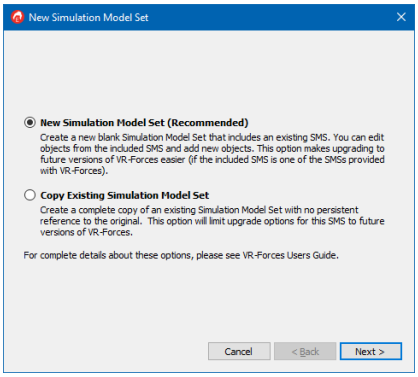


Figure 67-8. New Simulation Model Set dialog box

2. Select New Simulation Model Set.
3. Click Next. The Select Model Set to Include dialog box opens.
4. Select one or more SMSs to include.
5. Click Next. The New Model Set Name dialog box opens.
6. In the File name box, type a name for the new model set.
7. Click Finish. The Simulation Object Editor loads systems information from the included model sets.
8. Edit the SMS as desired.
9. Choose **File** → **Save Simulation Model Set**.

67.7.2. Creating a New SMS by Copying an SMS

You can create a new SMS by copying an existing SMS. When you do this the new SMS includes any SMSs that were included by the original SMS. All of the simulation objects in the new SMS are editable.

To create a new simulation model set by copying an SMS:

1. Choose **File** → **New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 67-8).
2. Select Copy Existing Simulation Model Set.
3. Click Next. The Select Model Set to Copy dialog box opens.
4. Select a model set and click Next. The Include Simulation Model Sets dialog box opens. It lists the SMSs included in the SMS that you are copying, if any.
5. Optionally, edit the list of SMSs to be included. (For details about how to add and remove included SMSs, please see the procedure in "Including Simulation Model Sets in Other Simulation Model Sets," on page 67-7, starting with step 6.)
6. Click Next. The New Model Set Name dialog box opens.
7. In the File name box, type a name for the new model set.
8. Click Finish. The Simulation Object Editor loads information from the source model set.
9. Make any changes you want to the model set.
10. Choose **File** → **Save Simulation Model Set**.

67.8. Importing .entity Files

If you have *.entity* files that you created with a previous version of VR-Forces or by multiple persons on different computers, you may want to import them into an SMS. The easiest way to do this is to copy the files into the *vrfSim* directory in the SMS (*./data/simulationModelSet[<model_set>/vrfSim*). However, you can also import the files from the Simulation Object Editor.

To import *.entity* files into an SMS:

1. Choose **File** → **Add Entity File**. The Add Entity File dialog box opens.
2. Select the file that you want to import.
3. Click Open. The file gets copied to the SMS.

67.9. Applying New and Updated Variables

All simulation object models reference one of the basic platforms, such as ground, fixed-wing, and point object. Most simulation objects reference one or more system definition files. If you update the variable bindings in a platform file or system definition file, you must apply these changes to the simulation objects of that platform type. Although you could do this by hand, as described in "Adding Variable Bindings to Platform and System Files," on page 74-22, that would be a tedious and error-prone activity. The Simulation Object Editor can apply all updates to all files automatically.

Similarly, if new variables are added to the API in descriptor files, these variables must be added to the system definition files and OPE files.

To apply new and updated variable bindings:

1. Choose **File** → **Regenerate SMS Files**. The Regenerate SMS Files dialog box opens.
2. Select the check box for each update you want, as follows:
   - Entity File Updates. All new variable bindings in the OPE files and systems are added as elements in the appropriate *.entity* files.
   - OPE File Updates. Applies changes to platform file descriptors to the OPE files.
   - System File Updates. Updates system files and system template files with any new variables that have been added to the system file descriptors.
3. Click OK.

67.10. Importing and Exporting Object Information

You can export a list of all simulation objects in an SMS to a CSV file. You can open the CSV file in a spreadsheet application. You can edit some of the object information and then import the updated object information back into the SMS. This format lets you quickly edit information without selecting objects one-by-one in the Simulation Object Editor.

For each object in the SMS, the object information file lists the object type enumeration, name, short name and filtering tags. You can edit the name, short name and tags.

Although you can also edit the object type, doing so will break the relationship between the limited information in the spreadsheet and the complete configuration stored in the SMS. It is strongly recommended that you not edit the object type.

67.10.1. Exporting Object Information

To export object information:

1. Choose **Model** → **Export Object Information**. The Export Object Information dialog box opens.
2. Type a name for the file.
3. Click Save.

67.10.2. Importing Object Information

To import object information:

1. Choose **Model** → **Import Object Information**. The Import Object Information dialog box opens.
2. Select the file you want to import.
3. Click Open.

67.11. Editing the Tags List

If you want to organize simulation objects into tags that are not included in the default tags list, you can add new tags. You can also remove and rename tags. Tags for simulation objects are stored in *./data/simulationModel-Sets/<sms>/gui/tags.csv*. Tags for systems are stored in *./data/simulation-ModelSets/<sms>/gui/system-tags.csv*.



The procedures in this section also apply to editing the Country list. Select Country on the Edit menu instead of Tags.

67.11.1. Adding a New Tag

When you add a new tag, you can give it a rank. Ranks determine how tags are displayed in the tag window. They work as follows:

- If a tag is a top-level tag, the rank determines the order in which it is listed relative to other top-level tags.
- If a tag is not a top level tag, the rank determines the preference for including the tag in the tag list. By default, the tag window can only display a maximum of 15 tags. If a top level tag has more than 15 possible sub tags, VR-Forces sorts the tags by rank order starting with 1. Within each group of tags with the same rank, it sorts them by how many simulation objects use that tag. Once it determines the 15 highest ranking tags, it displays them in alphabetical order.



You can configure the maximum number of tags that can be displayed. To do so, change the value for the `myShowTopCategories` element in *./appData/settings/vrfGui/default_PaletteInformationSettings.pisx*.

To add a new tag:

1. Choose **Model** → **Tags**. The Tags Editor opens (Figure 67-9).

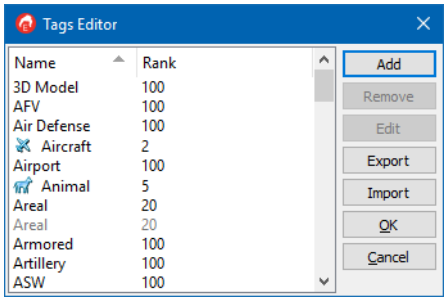


Figure 67-9. Tags Editor

2. Click Add. The Add Tag Information dialog box opens.
3. Type the name of the new tag.
4. Optionally, specify a rank.
5. To put the tag in the top row, select the Top Level Tag check box.
6. To make the tag visible, select the Visible check box.
7. To use an icon instead of the tag name, click the Icon button. The Select Tag Icon dialog box opens.
8. Select an icon file.
9. Click Open.
10. Click OK.



If you add a tag and do not assign a simulation object to it, it does not get saved when you save the simulation model set.

67.11.2. Removing a Tag

If you remove a tag, it gets removed from any entity that is assigned to it. You must save the SMS to make the change permanent.

To remove a tag:

1. Choose **Model** → **Tags**. The Tags Editor opens.
2. Select the tag that you want to remove.
3. Click Remove.
4. Click OK.

67.11.3. Renaming a Tag

When you rename a tag, all simulation objects that are assigned to the old name are reassigned to the new name. The old name remains in the Tags list until you save the simulation model set, but it is not listed in the Tags Editor. If you assign a simulation object to the old tag name before saving the SMS, the old name is retained and added to the list in the Tags Editor.

To rename a tag:

1. Choose **Model** → **Tags**. The Tags Editor opens.
2. Select the tag that you want to rename.
3. Click Edit. The Edit Tag Information dialog box opens.
4. Type the new name.
5. Click OK. The tags list is updated.
6. Click OK.

67.11.4. Undoing Tag Changes

You can undo changes made to the tag list and entity tag assignments.

When you undo tag changes, all changes to the tags list and all changes to entity tag assignments made since the last save of the simulation model set are reversed. Be sure that this is what you want before completing this procedure.

To undo tag changes:

1. Choose **Model** → **Revert Tags**. A confirmation prompt is displayed.
2. Click Yes.

67.11.5. Using Auto Tags

You can automatically assign a set of tags to new models, to specific models, and to all objects. The tags are assigned by object type based on the information in *./appData/settings/simulationObjectEditor/autoTagInformation.csv*. (The Simulation Object Editor reads the file when it starts up, so if you edit the file while the SOE is open, you must restart the SOE to pick up the changes.)

When you create a completely new model (not a new model from an existing model), if the object type has an entry in *simulationObjectEditor.csv*, the associated tags are automatically added to the model. (When you create a new model from an existing model, the new model has the same tag list as the model it was created from.)

➤ To assign auto tags to the entire object list in an SMS, choose **Model** → **Apply Auto Tags**.

To apply auto tags to a specific model:

1. Select the model whose tags you want to update.
2. Click the Apply Auto Tags button () above the Tags window.

67.11.6. Assigning Tags Based on Visual Model Characteristics

You can automatically assign tags to simulation objects that have 3D models based on their visual characteristics, such as articulated parts and display states. This process requires the SOE to examine every visual model and can take a long time.

➤ To assign tags based on visual model characteristics, choose **Model** → **Assign Tags Based on Visuals**.

67.11.7. Exporting Tags

You can export the tags in an SMS. They are saved to a comma separated values (CSV) file. You can then import the tag list into another SMS or another VR-Forces installation.

The tag list lists of tags, priorities, and icons associated with the tags, if any. It does not include any information about simulation objects.

To export the tag list:

1. Choose **Model** → **Tags**. The Tags Editor opens.
2. Click Export. The Export Tag Information dialog box opens.
3. Specify a name for the tag file.
4. Click Save.

67.11.8. Importing Tags

To import a tag list:

1. Choose **Model** → **Tags**. The Tags Editor opens.
2. Click Import. The Import Tag Information dialog box opens.
3. Select the tag file you want to import.
4. Click Open.

67.11.9. Managing Tags Using the Object Information File

If you want to edit the tags for many simulation objects, including adding new tags, editing individual simulation objects one-by-one can be tedious. You can speed up the process by editing them in a spreadsheet. To do this:

1. Export the simulation object list to a CSV file, as described in "Exporting Object Information," on page 67-26.
2. Make the changes you want to tag assignments.
3. Import the edited object information file into the SMS.

67.12. Managing the Forces List

When you create an object in the front-end, you must choose a force. You manage the list of forces available and their coloring in the Simulation Object Editor.

67.12.1. Adding a Force

Adding new forces may be useful if you are simulating participants from multiple countries or if you want to easily model and change force relationships for non-governmental forces such as militias or insurgent groups.

To add a force:

1. Choose **Edit** → **Forces**. The Forces Editor opens (Figure 67-10).

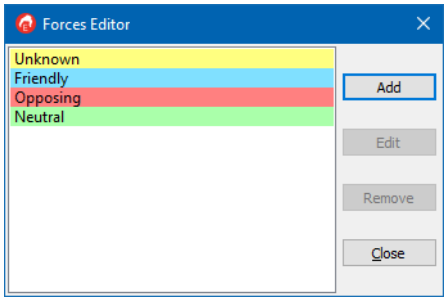


Figure 67-10. Forces Editor

2. Click Add. The Force Hostility Editor opens (Figure 67-11). The new force is assigned a force ID. You cannot edit the force ID.

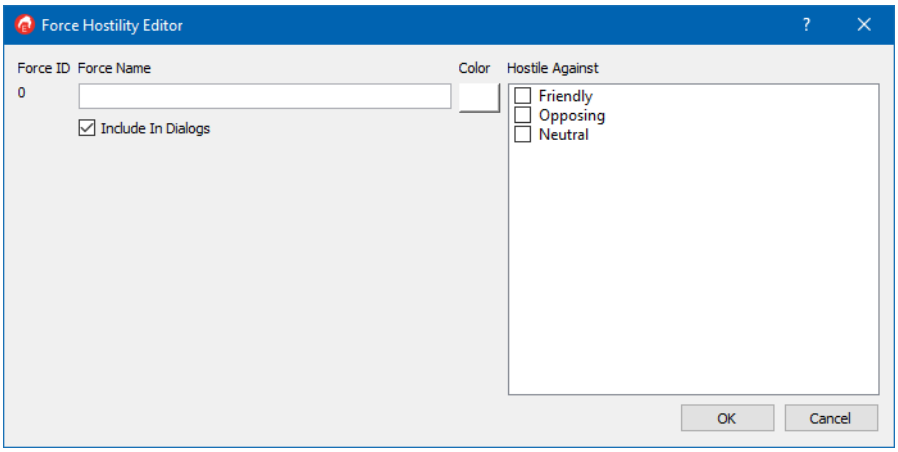


Figure 67-11. Force Hostility Editor

3. In the Force Name box, type a name for the force.
4. Click the Color button. A color picker opens.
5. Select a color for the force.
6. Optionally, in the Hostile Against window, select the forces that the new force will be hostile towards. (You can change hostility at run time in the Hostility Matrix. For details, please see "Managing Force Hostility Relationships," on page 9-13.
7. If you want the force to be listed in dialog boxes that list forces, select the Include in Dialogs check box.
8. Click OK.
9. Click Close.

67.12.2. Editing a Force

You can edit a force's name, color, hostility, and use. Each force has a force number. When you edit a force, you are changing the attributes of that numbered force.

To edit a force:

1. Choose **Edit** → **Forces**. The Forces Editor opens (Figure 67-10).
2. Select the force that you want to edit.
3. Click Edit. The Force Hostility Editor opens (Figure 67-11).
4. To edit the name, change the text in the Force Name box.
5. To edit the color associated with the force, click the Color button and choose a color from the color picker.
6. To edit hostility relationships, in the Hostile Against window, select the forces that the force should be hostile towards.
7. To edit whether or not the force is listed in dialog boxes that list forces, select or clear the Include in Dialogs check box.
8. Click OK.
9. Click Close.

67.12.3. Removing a Force

If you create a scenario that has simulation objects with a particular force and you then remove the force, the next time you open the scenario, VR-Forces still maintains that force assignment for the simulation objects. The force will be listed in the Hostility Matrix. However, entity icons will not be colored and you will not be able to create any simulation objects with the removed force.

To remove a force:

1. Choose **Edit** → **Forces**. The Forces Editor opens (Figure 67-10).
2. Select the force that you want to remove.
3. Click Remove. The force is removed from the list.
4. Click Close.

67.13. Copying the Object List to the Clipboard

You can copy the object list to the clipboard and paste it as text into word processing applications.

➤ To copy the object list to the clipboard, choose **Edit** → **Copy Names in List to Clipboard**.



68. Editing Common Parameters

This chapter explains how to use the Simulation Object Editor to edit simulation object parameters that are common to most simulation object types for both entity-level scenarios and aggregate-level scenarios. For details about editing parameters that are specific to individual entities, please see Chapter 69, *Editing Entities for Entity-Level Scenarios*. For details about editing units, please see Chapter 70, *Editing Units for Entity-Level Scenarios* and Chapter 71, *Editing Aggregate-Level Objects*.

Editing Generic Object Parameters 68-3
- Selecting an Object to Edit 68-3
- Editing an Object's Tags 68-4
- Changing an Object's Object Type Enumeration 68-5
- Changing an Object's Name 68-7
- Specifying Whether an Object Can Be Created 68-8
- Specifying the Object Creation Palette 68-8
- Specifying the Default Overlay 68-8
- Specifying the Platform 68-9
- Editing a Simulation Object's Country List 68-9
- Creating Icon Lists 68-10
- Undoing (Reverting) Changes to an Object 68-12

Editing a Simulation Object's Visual Model 68-13
- Changing the 3D Model or Colorized Model 68-14
- Changing the 2D Icon Using a Font 68-16
- Changing the 2D Icon Using an Image 68-16
- Editing a Spot Report Model 68-18
- Adding a New Visual Model or Image 68-18
- Displaying Details about the Visual Model 68-20
- Changing a Visual Model's Type 68-22
- Specifying Visual Markings 68-23
- Importing Visual Models 68-24
- Pruning Visual Models 68-25

Exporting Visual Models 68-25
Reloading Visual Models in the VR-Forces GUI 68-25
Creating a New Simulation Object 68-26
Creating a New Model from an Existing Model 68-27
Deleting an Object Model 68-27
Creating a Simulation Object Group 68-28
Setting the Simulation Connection when Creating an SOG 68-29
Editing Tactical Graphics 68-30
Editing a Tactical Graphic's Menu Icon 68-30
Setting the Default Values for Tactical Graphics Properties 68-31
Specifying the Properties that Can Be Edited in the Front-End 68-32
Editing the Visual Model for a Tactical Graphic 68-33
Creating a Template 68-36
Editing Systems in the Simulation Object Editor 68-36
Creating a New System 68-37
Editing Probability Tables 68-38
Deleting a System 68-41
Reverting a System 68-42

68.1. Editing Generic Object Parameters

This section explains how to edit the parameters that are common to most objects. The procedures in this section apply to simulation objects, tactical graphics, cultural features, and other objects such as weather and contamination areas.



- By default, the SMSs provided with VR-Forces are read-only. It is strongly recommended that you not edit these SMSs. You should create copies or new SMSs that include them. For details, please see "Creating New Simulation Model Sets," on page 67-23.
- If one of the object parameters described in this chapter does not apply to a particular object type, it is either blank or uneditable in the Simulation Object Editor.

68.1.1. Selecting an Object to Edit

Most of the Simulation Object Editor procedures assume that you have selected an object in the object list to edit.

To select an object to edit:

1. Optionally, in the object list, select the tags for the type of object you want to edit.
2. Select the object in the list.

68.1.2. Editing an Object's Tags

Objects are organized by tags on the Create menu and the object creation palettes. You can change the tags that an object uses. For details about adding new tags to the list, please see "Adding a New Tag," on page 67-27.

To assign a tag to an object:

1. Select the object whose tags you want to edit.
2. Select the General tab (Figure 68-1).

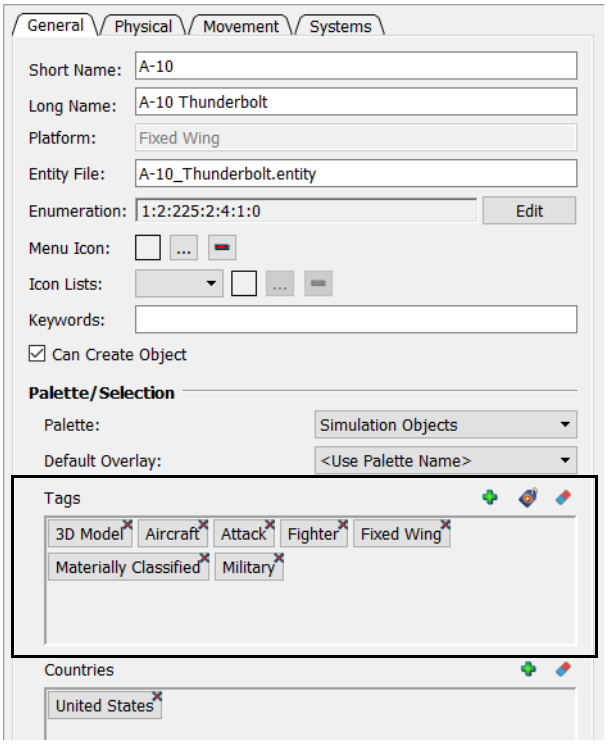


Figure 68-1. General tab

3. Click the Add/Remove button (+) above the Tags window.
4. The Select Tags dialog box opens.
5. Select the check boxes for the tags that apply to the simulation object.

68.1.3. Changing an Object's Object Type Enumeration

The Simulation Object Editor lets you change an object's object type (DIS/RPR enumeration) and matching type. (For details about published and matching object types, please see "Published Object Types and Matching Object Types," on page 74-6.) You might want to or have to do this for any of the following reasons:

- If you create a new model, you must specify an entity enumeration (Object Type). (The Simulation Object Editor automatically constructs a Matching Type whose Domain and Kind field match the Object Type and whose other fields are wild cards.)
- If you create a new model from an existing model, the Simulation Object Editor automatically increments the existing model's enumeration for use with the new model. However, you might want to use a different enumeration than the one the Simulation Object Editor generates. (The Simulation Object Editor automatically creates a Matching Type that is the same as the Object Type.)
- You might want to change an existing object's enumeration.
- You might want to change an object's matching object type.



The Entity Enumeration value displayed in the Simulation Object Editor is the 7-digit DIS or RPR FOM enumeration that VR-Forces uses to build the published object type. The matching object type is not displayed in the Simulation Object Editor, but you can specify it in the Change Enumeration dialog box as described in this procedure.

To change an object's enumeration:

1. Select the object in the object list.
2. Select the General tab (Figure 68-1).
3. Click Edit next to the Enumeration text box. The Change Enumeration dialog box opens (Figure 68-2).

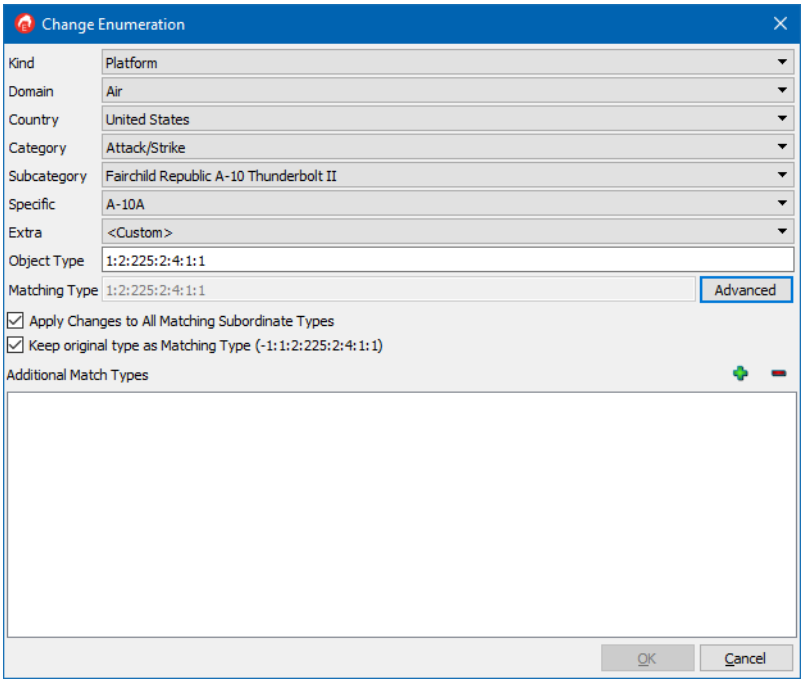


Figure 68-2. Change Enumeration dialog box

4. Edit the enumeration by selecting options from the lists for enumeration elements or type a new enumeration in the Object Type text box.
5. Optionally, specify the Matching Type as follows:
   a. Click Advanced. The dialog box expands (Figure 68-3).

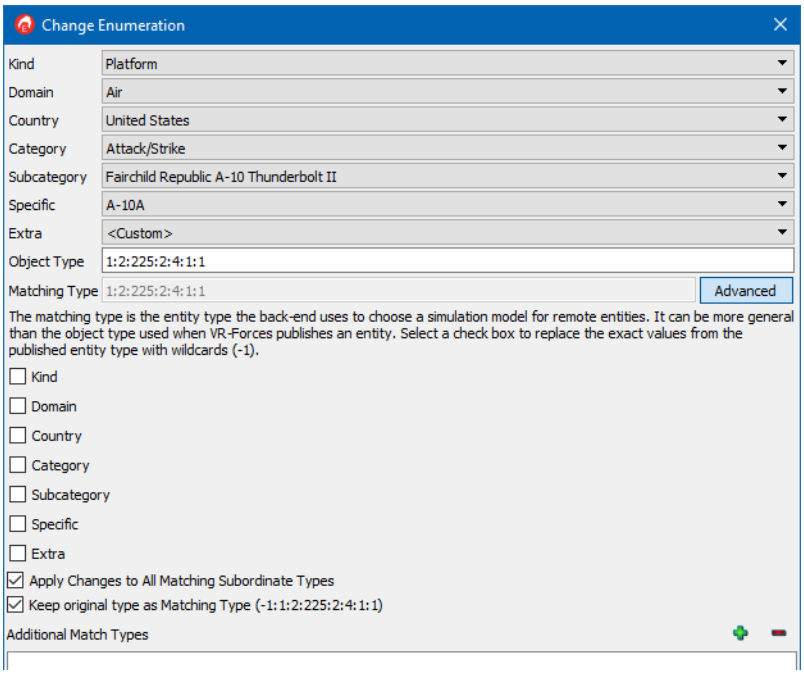


Figure 68-3. Change Enumeration dialog box, Advanced

b. For each field in the matching type enumeration that you want to be a wild card (-1), select the appropriate check box.

6. Optionally, clear the Apply Change to All Matching Subordinate Types check box. Usually you will want an enumeration change to affect all cases where this simulation object will be created. However, if there is some reason that you want an enumeration change to not affect simulation objects created as part of a unit, you can do that by clearing this check box. A more straightforward approach would be to create a new simulation object from the existing simulation object rather than changing the enumeration.
7. Click OK. The enumeration changes.

68.1.4. Changing an Object's Name

The object name is the name listed on the object creation palettes and Create menu. The text placed on the screen next to a 2D icon is the marking text. The marking text is based on the Short Name, which is configured on the General tab.

To change an object's name:

1. Select the object in the object list.
2. Select the General tab (Figure 68-1).
3. Type a new name in the Long Name text box.

68.1.5. Specifying Whether an Object Can Be Created

You can control whether or not a simulation object is listed on an object creation palette. It may not make sense to be able to create some types of simulation objects, such as missiles, which are normally added to a scenario and removed dynamically.

To enable or disable creation of a simulation object from the Simulation Objects Palette:

1. Select the object in the object list.
2. Select the General tab.
3. Select or clear the Can Create Object check box (Figure 68-1).

68.1.6. Specifying the Object Creation Palette

You can specify which of the object creation palettes an object should be listed on.

To specify the object creation palette for an object:

1. Select the object in the objects list.
2. Select the General tab (Figure 68-1).
3. In the Palette list, select the palette you want to put it on.

68.1.7. Specifying the Default Overlay

When an object gets created, it gets placed on an overlay. By default, the object is placed on an overlay that has the same name as the object creation palette from which it is created. You can create additional default overlays and assign them to objects. You can still assign the object to a different overlay when it gets created. For details about overlays and assigning objects to them, please see "Creating and Editing Overlays," on page 39-2 and "Specifying an Object's Properties Before You Create It (Click to Locate)," on page 14-15.

To specify the default overlay:

1. Select the object in the objects list.
2. Select the General tab (Figure 68-1).
3. Select an option from the Default Overlay list.

Adding a Default Overlay

The default option for specifying the default overlay is to use the name of the object creation palette for the object. You can add overlays to use as the default overlay.

To add a default overlay:

1. Choose **Edit** → **Default Overlay Layers**. The Overlay Layers dialog box opens.
2. Click Add. The New Overlay Layer dialog box opens.
3. Type a name for the layer.
4. Click OK. The layer is added to the list in the Overlay Layers window.
5. Click OK. The overlay is now available in the Default Overlay list.

68.1.8. Specifying the Platform

You cannot change the platform for existing simulation objects or new objects created from existing models. You can only specify the platform for new models that are not created from existing models. For details, please see "Creating a New Simulation Object," on page 68-26.

68.1.9. Editing a Simulation Object's Country List

The country code in a simulation object's object enumeration specifies the country in which the object is manufactured. It does not indicate where it is used, and many object types are used in multiple countries. When creating a scenario, it can be convenient if the Simulation Objects Palette only lists the objects used by the country for which you are laying down forces. The Country list in the Simulation Object Editor lets you specify the countries that use a vehicle. You can then filter the Simulation Objects Palette by country.

You can also apply a country to humans and cultural features if they are specific to a country.

The Country list works exactly like the Tags list. For details about how to configure the list of countries, please see "Editing the Tags List," on page 67-27.

To assign a simulation object to a country of use:

1. Select the object in the object list.
2. Select the General tab (Figure 68-1).
3. Click the Add button (+) above the Countries window.
4. The Select Countries dialog box opens.
5. To select one country and close the dialog box, click the country name. To add multiple countries, **CTRL**+click the names of the countries you want to add.

Clearing the Countries List

If you add a new model by copying another model, you may want to quickly clear the Countries list before selecting the countries for the new model. (You can clear the other lists the same way.)

➤ To clear the Countries assigned to an object, on the General tab, click the Clear button ( ) above the Countries window.

68.1.10. Creating Icon Lists

By default, the object lists on the object creation palettes and the Objects List Panel show the 2D icon for the each object. For entities and units, this is the MIL-STD 2525B icon. For tactical graphics it is a simple graphic representation of the object, such as a line for linear graphics or a box for areal graphics. However, if you want to use images that are more specific to simulation object types, you can create custom icon lists that replace the default images. For example, instead of a straight line for a route, you might want to use a segmented line. Or instead of a 2D icon for an entity you might want an image of the entity or a generic platform image.

Figure 68-4 illustrates object palettes with custom images (A-4 Skyhawk and Route).

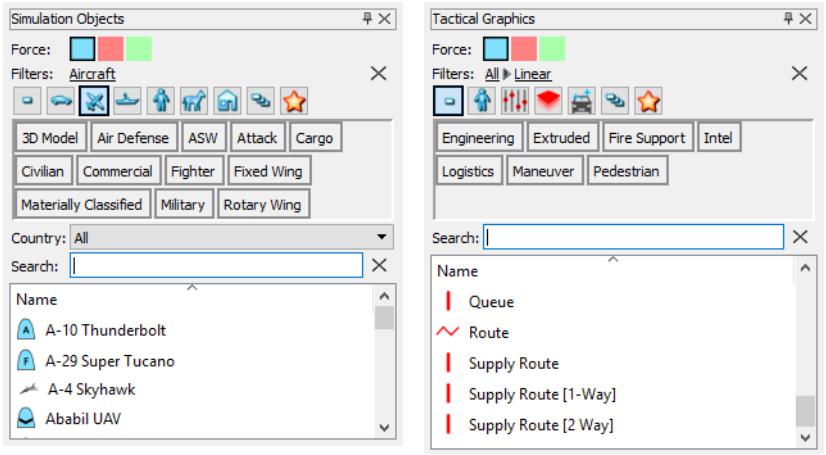


Figure 68-4. Alternative icons

When you create a custom icon list you create a list and then specify an image for each object that you want to change. If you do not specify an image for an object, it uses the default icon. You can create as many icon lists as you want. You specify the icon list to use on the Application Settings dialog box, GUI Settings page.

To create an icon list:

1. Choose **Edit** → **Icon Lists**. The Icon Lists Editor opens.
2. Click Add. The Add List dialog box opens.
3. Type a name for the new list.
4. Click OK.
5. In the Icon Lists Editor, click OK.

To remove an icon list:

1. Choose **Edit** → **Icon Lists**. The Icon Lists Editor opens.
2. Select the icon list that you want to remove.
3. Click Remove.
4. Click OK.

To change the name of an icon list:

1. Choose **Edit** → **Icon Lists**. The Icon Lists Editor opens.
2. Select the icon list whose name you want to change.
3. Click Edit. The Edit List dialog box opens.
4. Type a new name for the list.
5. Click OK.
6. In the Icon Lists Editor, click OK.

Specifying Icons for an Icon List

After you create an icon list, you specify the images for each simulation object that you want to change the icon for.

To specify an image for an icon list:

1. Select a simulation object in the objects list.
2. Select the General tab.
3. In the Icon Lists drop-down list, select the icon list that you want this object to use.
4. Click the Browse button (...). The Select Menu Icon dialog box opens.
5. Select an image. It can be a PNG, SVG, JPG, or BMP file.
6. Click Open. The image is displayed next to the Browse button.
7. Repeat this procedure for each simulation object for which you want to specify an image.
8. Save the Simulation Model Set.

Using an Alternative Icon List

If you create an icon list, you can specify that VR-Forces use it instead of the default icon list.

To specify use of an alternative icon list:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 68-5).

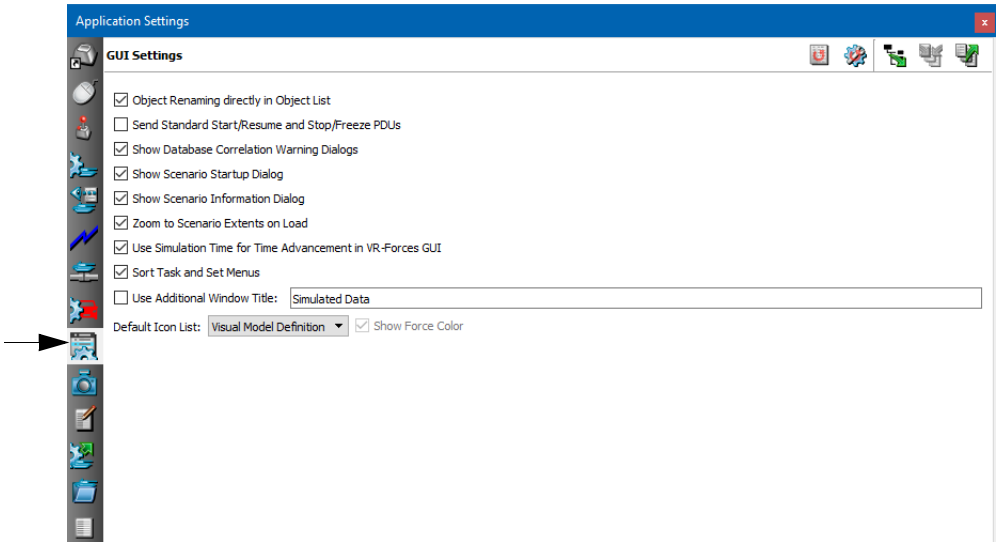


Figure 68-5. GUI Settings page

3. In the Default Icon List list, select the icon list that you want to use.
4. Optionally, select Show Force Color. This places a colored square behind the image to show its force.

68.1.11. Undoing (Reverting) Changes to an Object

If you have made changes to an object and have not yet saved the simulation model set, you can revert to the original simulation object configuration. You can revert an object at any time prior to saving the SMS, regardless of how many other objects you may have changed since you edited the simulation object you want to revert.

To undo changes to an object:

1. Select the object in the object list.
2. Choose **Model** → **Revert Model**.

68.2. Editing a Simulation Object's Visual Model

The Visual Model Editors dialog box allows you to configure all aspects of entity visualization, including the 2D icon and 3D model used. (For details, please see the chapters in XIV, *Configuring Visual Models in the Visual Models Editor*.) However, the Simulation Object Editor lets you quickly and easily change the 3D model or 2D icon used to represent a simulation object. When you change a model in the Simulation Object Editor, the entity's model definition also gets changed.

You edit visual model parameters on the Visual Models Panel (Figure 68-6).



Figure 68-6. Visual Models Panel

The Visual Models Panel is a fully functional window that supports the same mouse-look and keyboard controls as the VR-Forces main window. If you change the observer view of the selected entity's visual model, the window maintains that offset as you select other entities. Press **Spacebar** to reset the view.

The panel has the following buttons:

- Edit Visual Model. Automatically opens the correct editor based on the model set that is displayed.
- Edit Wake Model. Changes the wake settings for surface entities. (Hidden for other platforms.)
- Visual Effects. Lets you display how the model looks when trailing effects, shadows, and wakes are enabled.
- Switch Node States. Lets you display how the model looks when switch node states are set.
- Environment Settings. Lets you display what the model looks like at different times of day and with different wind and ocean states.

To change the visual model for a simulation object:

1. In the object list, select the object whose model you want to change. The model for the currently selected model set is displayed in the Visual Model Panel.

> The visual model window responds to the same keyboard and mouse controls as the front-end.

2. In the Model Set list in the Visual Model group box, select the model set that you want to edit for this simulation object.
3. Change the 3D model, 2D icon, or 2D image as described in the following sections.

68.2.1. Changing the 3D Model or Colorized Model

To change a 3D model or 3D colorized model:

1. In the Visual Model panel, click the Edit button ( ). The Set 3D Models Visuals dialog box opens (Figure 68-7).
2. If the dialog box has options for 3D Models or DI-Guy, select Use 3D Model. The Modify Visuals dialog box lists available models (Figure 68-7). (You can add new models. For details, please see "Adding a New Visual Model or Image," on page 68-18.)

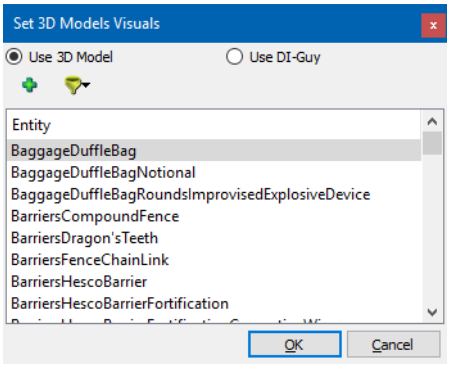


Figure 68-7. Modify Visuals dialog box (3D)

3. Select the model that you want to use. The new model is displayed in the panel.

By default, the Modify Visuals dialog box tries to filter the list to show only the type of entity that you are editing. If you do not see the models you expect to see in the list, particularly if you know that the models are available in the *./data* directory, try changing the filter. For example, surface entities are of type Watercraft, while subsurface entities are of type Entity.

---

4. Click OK to close the Modify Visuals dialog box.

68.2.2. Changing the 2D Icon Using a Font

To change a 2D icon using the MIL-STD font:

1. Select the simulation object whose icon you want to edit.
2. In the Visual Models Panel, in the Model Sets list, select 2D Icons.
3. Click the Edit button (). The Set 2D Icons Visuals dialog box opens (Figure 68-8).

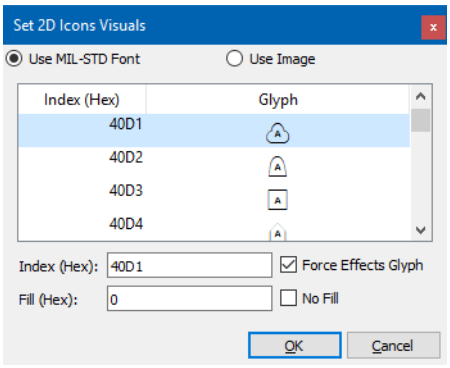


Figure 68-8. Modify Visuals dialog box (2D)

4. Select the Use MIL-STD Font option. The dialog box lists index values and associated glyphs.
5. Select the icon glyph that you want to use. The new icon is displayed in the panel.
6. Optionally, change the parameters.
7. Click OK to close the dialog box.

Regenerating Spot Report Definitions

When you change the 2D icon for a simulation object, the spot report icon for that object does not change. You can change the icon manually, as described in "Editing a Spot Report Model," on page 68-18, or you can automatically update spot report definitions.

To regenerate spot report definitions:

1. Choose **File** → **Regenerate Spot Report Definitions**. A prompt opens.
2. Click Yes.

68.2.3. Changing the 2D Icon Using an Image

To change a 2D icon using an image:

1. Select the simulation object whose icon you want to edit.
2. In the Visual Models group box, in the Model Sets list, select 2D Icons.

3. Click the Edit button (). The Set 2D Icons Visuals dialog box opens (Figure 68-8).
4. Select the Use Image option. The dialog box lists the forces that are configured for this simulation model set (Figure 68-9).

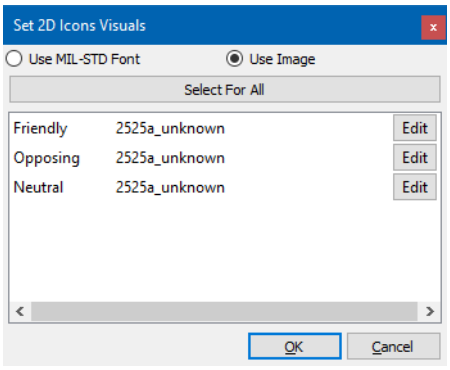


Figure 68-9. Modify Visuals dialog box (2D image)

5. Click the Edit button for the force whose icon you want to change, or to choose the same image for all forces, click Select for All. The Select Model Definition dialog box opens (Figure 68-10). It lists the available images. By default the list is for MIL-STD 2525a icons. (You can add new images. For details, please see "Adding a New Visual Model or Image," on page 68-18.)

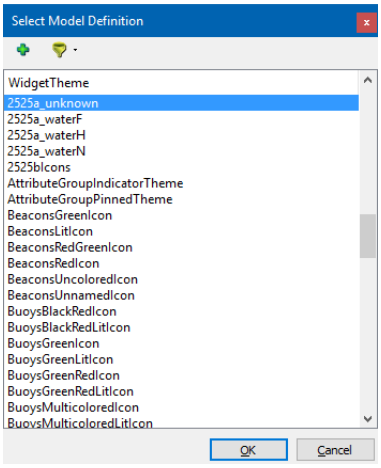


Figure 68-10. Select Model Definition dialog box

6. Select the image that you want to use. The new image is displayed in the panel.
7. Click OK.
8. Click OK to close the Modify Visuals dialog box.

68.2.4. Editing a Spot Report Model

You can edit the model that is used when a spot report is at the full knowledge level. The default configurations assign 2D icons to spot reports. In theory, you could assign some other type of model to a spot report. (The icons used at the detected, classified, and identified levels are not editable.)

To edit a spot report icon:

1. Select the simulation object whose icon you want to edit.
2. In the Visual Models Panel, in the Model Sets list, select one of the spot report options.
3. Click the Edit button (). A dialog box appropriate to the model type opens.
4. Change the model, as described in "Changing the 3D Model or Colorized Model," on page 68-14, "Changing the 2D Icon Using a Font," on page 68-16, or "Changing the 2D Icon Using an Image," on page 68-16.

68.2.5. Adding a New Visual Model or Image

When you change a model or image in the Simulation Object Editor, you are presented with a list of the currently configured model definitions or images. You can add additional models to the list. You can add new model definitions in the context of editing a simulation object or independently of editing a simulation object. If you edit the list in the context of editing a simulation object, you can also add images to the list.

Adding A Model Definition while Editing a Simulation Object

To add a new model or image to the visual models list while editing a simulation object:

1. In the object list, select the object whose model you want to change. The model for the currently selected model set is displayed in the Visual Model group box.
2. In the Visual Model group box, in the Model Sets list, select the model set that you want to edit.
3. For 3D models or 3D colorized models:
   a. Click the Edit button ( ). The Set 3D Models Visuals dialog box opens (Figure 68-7).
   b. Click the Add button ( ). The Model Definition dialog box opens. This is a standard file chooser dialog box.

   For 2D icons, do the following:
   a. Click the Edit button ( ). The Set 3D Icons Visuals dialog box opens (Figure 68-8).
   b. Select the Use Image option. The dialog box lists the forces that are configured for this simulation model set.
   c. Click the Edit button for the force to which you want to add an image. The Select Model Definition dialog box opens (Figure 68-9).
4. Select the file for the model that you want to add to the list. It must be a model or image type that VR-Forces supports. (This file becomes the value for the filename attribute of the model definition.)
5. Click OK. The Model Definition Name dialog box opens.
6. Type a name for the model definition.
7. Click OK.
8. Click OK to close the Modify Visuals dialog box.

Adding Model Definitions Independently of Editing a Simulation Object

You can add model definitions in the Simulation Object Editor independently of any simulation object just as you can in the Visual Model Editors dialog box. The only attribute that you can set is the filename. However, in many cases that is the only attribute you need. If you need to specify other attributes, you must use the Visual Model Editors dialog box.

To add a model definition independently of editing a simulation object:

1. Choose **Edit** → **Add Model Definition**. The Add Model Definition dialog box opens.
2. Click the Add button (✚). The Model Definition dialog box opens. This is a standard file chooser dialog box.
3. Select the file for the model that you want to add to the list. It must be a model or image type that VR-Forces supports. (This file becomes the value for the filename attribute.)
4. Click OK. The Model Definition Name dialog box opens.
5. Type a name for the model definition.
6. Click OK. The new model definition is added to the list.
7. Click OK to close the Add Model Definition dialog box.

68.2.6. Displaying Details about the Visual Model

The Visual Models Panel displays the model used for the selected simulation object. You can change how the model is displayed and show information about it, as follows:

- Bounding volumes. Display the bounding volume.
- Environment settings. Show the model at different times of day, different view magnification, different wind speed, and different water quality.
- Markers and marker labels. Displays labels for the markers for sensor and weapon positions. Also, show markers as 2D only.
- Visual effects. Show effects such as shadows, trailing effects, smoke, and flames.
- Model sets. Specifies which model set to use to display the entity.
- Switch node states. Displays the different switch nodes for the model, if applicable. If a model has articulated parts, you can enable them individually. They move at five times the minimum rotation or translation speed.
- View. Perspective view or view the model from the specified side (left, front, top, and so on.)

Displaying a Visual Model's Bounding Volume

You can display the bounding volume of a 3D model. You can edit the dimensions of the bounding volume on the Physical tab.

➤ To display a model's bounding volume, choose **Preferences** → **Show Bounding Volumes**.

Displaying Switch Node States

If a visual model has switch node states, you can display them. Switch nodes can represent damage states and articulated parts.

To display switch node states:

1. Choose **View** → **Switch Nodes States** or click the Switch Nodes States button (). The Switch Nodes States dialog box opens.
2. To see a damage state, select an option from the Damage list.
3. To see articulated parts, select the check box for the state you want to display.
4. Click Close.

Changing the View of the Model

The default view for 3D models is a perspective view. You can also view a model from the front, back, sides, or top.

➤ To change the view of a model, in the Visual Models Panel, select an option from the View list.

Changing the Model Set

You can select the model set to use to display a visual model. In addition to the three primary model sets, you choose a spot report model set.

➤ To change the model set used to display a visual model, in the Visual Models Panel, select an option from the Model Sets list.

Displaying Markers and Marker Labels

You can display markers that show the location of support points for a model. You can display the markers as 2D or 3D and can also display text labels that identify them. You can edit the location of support points on the Physical tab.

➤ To toggle markers on and off, choose **Preferences** → **Show Markers**.

➤ To toggle marker labels on and off, choose **Preferences** → **Show Marker Labels**

➤ To toggle markers between 2D and 3D, choose **Preferences** → **Show Markers as 2D**

Displaying the Effects of Environment Settings

You can display how changing the following environment settings affects a visual model:

- Time of day
- View magnification
- Wind speed
- Wind direction
- Choppiness
- Ocean quality.

Time of day affects all models. The other settings only affect surface entities.

To display the effects or environment settings for a model:

1. Choose **View** → **Environment Settings** or click the Environment Settings button (). The Environment Settings dialog box opens.
2. Change the settings as desired.
3. Click Close.

Displaying Visual Effects for a Model

You can display the following visual effects for models:

- Dust clouds
- Contrails
- Tracks
- Flames
- Smoke
- Footprints
- Wakes
- Shadows
- Speed.

If an effect is not valid for the selected model, the check box is not available.

To display visual effects for a model:

1. Choose **View** → **Visual Effects** or click the Visual Effects button (). The Visual Effects dialog box opens.
2. Select the check box for each visual effect that you want to display.
3. To change the apparent speed of a moving model, adjust the Speed slider.
4. Click Close.

68.2.7. Changing a Visual Model's Type

The entity element definition hierarchy categorizes element definitions by type. These types associate a variety of parameters with the models in the category. Changing the visual type changes the 3D model for any simulation object that is using a generic model. If a simulation object has a specific model, changing the visual type does not affect it. However, it affects how the model is displayed in the SOE. Surface models are displayed with a background that lets you view wakes and other effects.

To change a visual model's type:

1. In the object list, select the model whose type you want to change.
2. Choose **Model** → **Advanced Visual Model Editing**. The Advanced Visual Model Editing dialog box opens.
3. Select an option in the Visual Type list.

68.2.8. Specifying Visual Markings

VR-Forces uses a process called texture mapping to apply different surface textures to a simulation object's physical geometry. For example, you can use this technology to change the colors of a car, change the logo and design on aircraft, and change markings on a vehicle. The Simulation Object Editor lets you see how different markings look on a vehicle, lets you apply specific markings to a vehicle, and lets you configure a vehicle to apply randomly assigned markings.

You can specify the visual markings that get applied to a simulation object when it is added to a scenario. Markings can be specific or random. A simulation object may have more than one type of visual marking. You can set each of them individually. The options are:

- Specific. Use the specific marking configured for the object.
- Random. Choose randomly from the markings available for the object.
- Weighted Random. Given a list of markings, there is an 80% chance that one of the markings in the top 20% of the list will be applied. There is a 20% chance that one of the other markings will be applied.
- Weighted Random Exponential. The markings towards the top of the list are more likely to be chosen then those later in the list.

For weighted random options the order of the list is based on the order that markings are listed in the relevant metadata file. If you want the markings to have a different priority, you must edit the metadata file.

For example, if colors in the metadata file are listed in the order red, blue, green, yellow, brown and you select the weighted random approach, most cars will be red, because there is an 80% chance of selecting the color in the top 20% of the list. (In a list of five items, the first item in the list is the top 20%.) If you want most cars to be brown, you would need to edit the metadata file to put brown at the top. (For details about metadata files, please see "The Metafile for Markings," on page 90-27.)

For details about how to change markings at run time, please see "Markings," on page 34-27. For details about how to configure texture maps and their associated files, please see "Adding Dynamically Changeable Markings to Models," on page 90-24.

Viewing a Model's Visual Markings

The Simulation Object Editor lets you see how the available visual markings look on a simulation object.

To view a model's visual markings options:

1. Select the simulation object.
2. In the Visual Model panel, click the Visual Markings button (). The Visual Markings dialog box opens (Figure 68-11). It has a list of options for each type of marking that the simulation object supports.

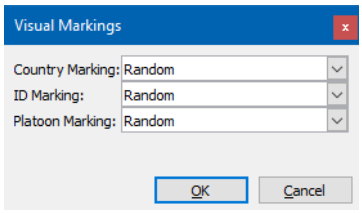


Figure 68-11. Visual Markings dialog box for the Fenneck vehicle

3. Select specific options in the lists for each marking type that you want to view.
4. Click Cancel. (If you click OK, the options you have chosen are applied to the model.)

Setting Visual Markings

In the Simulation Object Editor, you can specify the visual markings that get applied to a simulation object when it is added to a scenario.

To set the visual markings for a simulation object:

1. Select the simulation object.
2. In the Visual Model panel, click the Visual Markings button ( ). The Visual Markings dialog box opens (Figure 68-11).
3. For each type of marking, do the following:
   a. If you want to use a specific marking, select it from the list.
   b. If you want markings assigned randomly, select a random option in the list. For weighted random options, the markings at the top of your list are the most likely to be used.
4. Click OK.

68.2.9. Importing Visual Models

You can import visual models (element definitions, model definitions, and object type mappings) into an SMS. You might want to do this if you have edited an SMS and then decide that you need to revert to a previous version that you have saved in *./factory* or elsewhere. When you import visual models, the imported model definitions replace any matching definitions in the SMS. The import operation imports all leaf files in the selected directory.

To import visual models:

1. Choose **File** → **Import Visual Models**. A confirmation prompt opens.
2. Click Yes. A file chooser dialog box opens.
3. Select the directory from which you want to import visual models.
4. Click Choose.

68.2.10. Pruning Visual Models

Pruning visual models (element definitions, model definitions, and object type mappings) removes duplicate visual models from the set of leaf files that get loaded for an SMS. This improves the load time in the SOE and when you load a scenario.

Pruning only affects the SMS-specific definitions files. It does not touch any additional definition files that may have been saved from the Visual Model Editors dialog box.

To prune visual models:

1. Choose **File** → **Prune Visual Models**. A confirmation prompt opens.
2. Click Yes. A file chooser dialog box opens.
3. Select the directory from which you want to prune visual models.
4. Click Choose.

68.2.11. Exporting Visual Models

When you update an SMS, you can export the visual models (element definitions, model definitions, and object type mappings) to a directory other than the default directory in the SMS. This would typically be done for archival purposes, such as updating the files in *./factory*.

Only the SMS-specific definitions files get exported. If user-created leaf files were previously loaded, the definitions do not get exported back to those leaf files. They get included in the SMS-specific files.

To export visual models:

1. Choose **File** → **Export Visual Models**. A confirmation prompt opens.
2. Click Yes. A file chooser dialog box opens.
3. Select the directory to which you want to export visual models.
4. Click Choose.

68.2.12. Reloading Visual Models in the VR-Forces GUI

If you update the visual models associated with simulation objects and you have a scenario open, you can refresh the models used in the scenario without shutting down VR-Forces.

➤ To refresh visual models, in the VR-Forces GUI choose **Settings** → **Reload Visuals**.

68.3. Creating a New Simulation Object

You can create a new simulation object by using an existing object as a template or by building a completely new object.

> The procedure for creating a new unit is largely the same as for creating a new entity. The main difference is that units do not have 3D models and when you create a unit, you specify its subordinates.

To create a completely new object:

1. Choose **Model** → **New Model**. The Create New Model dialog box opens (Figure 68-12).

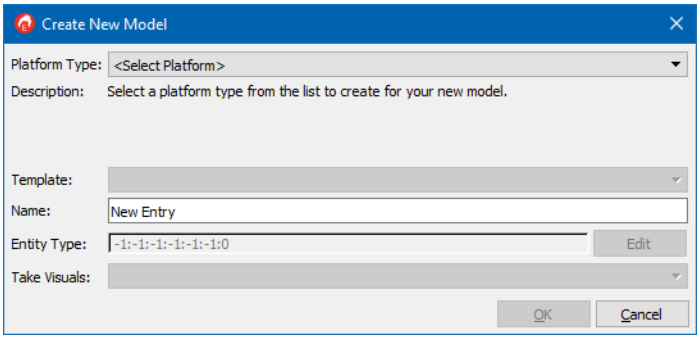


Figure 68-12. Create New Entity dialog box

2. In the Platform Type list, select a platform type.
3. Optionally, select a template. (Templates are specific to a platform type. There might not be a template for some platforms.)
4. In the Name box, type a name for the entity.
5. Optionally, for the Entity Type, click Edit and specify the object type. You can do this after you create the entity if you want to. You can also just accept the object type supplied by VR-Forces.
6. In the Take visuals list, select a specific model or Closest Match. If you select Closest Match, VR-Forces selects the entity model whose object type enumeration is closest to the one you specified.
7. Click OK. The Simulation Object Editor adds a new entry to the object list. The editor shows default values for all the fields.
8. Optionally, edit the enumeration.
9. Optionally, specify that this object is a favorite.
10. Specify 2D and 3D visual models.
11. Specify appropriate values for the object parameters.

12. Type a name to be used in the object's label (marking text) in the Short Name text box.
13. Specify a movement system.
14. Specify a damage model.
15. Optionally, add weapon systems.
16. Optionally, add sensors.
17. Save the simulation model set.

68.3.1. Creating a New Model from an Existing Model

If you create a new simulation object model from an existing model, the new model starts with all of the behavioral characteristics and systems of the original one. It increments the object enumeration. You just have to name it and customize it for the new simulation object.



You can use this procedure to create new units.

To create a new simulation object model from an existing model:

1. In the object list, select the source object.
2. Choose **Model** → **New Model from Existing**. The Create New Model dialog box opens.
3. Type a name for the model.
4. Optionally, edit the enumeration.
5. Click OK. The new simulation object is added to the object list. If you did not edit the enumeration, the original enumeration is incremented. A *.entity* file is created based on the name.
6. Edit the parameters to provide the new simulation object the characteristics you want for it.
7. Save the simulation model set.

68.4. Deleting an Object Model

To delete an object model from a simulation model set:

1. Select the object in the object list. (You can select multiple models using the **`SHIFT`** or **`CTRL`** keys.)
2. Choose **Model** → **Delete Model** or press **`Delete`**.

68.5. Creating a Simulation Object Group

You create simulation object groups (SOGs) in the Simulation Object Editor. You specify the members of the SOG in a version of VR-Forces that gets launched from the Simulation Object Editor. When the Simulation Object Editor launches VR-Forces, it needs to use a simulation connection. For details about setting the simulation connection, please see "Setting the Simulation Connection when Creating an SOG," on page 68-29.

Simulation object groups can include scripted tasks and sets. They become scenario-specific scripts when you add the SOG to a scenario.

Simulation object groups are saved with the extension *.sog*.

To create a simulation object group:

1. Start the Simulation Object Editor.
2. Load the SMS you want to use.
3. Choose **Model** → **New Object Group**. The New Object Group dialog box opens.
4. Type a name for the new SOG.
5. Click OK. The new group is added to the objects list and the Simulation Object Editor displays a simulation object group page.
6. Specify the tag, creation palette option, and other generic information as you would for any simulation object.
7. Optionally change Creation Options and the altitude offset.
8. Click Launch Object Group Editor. You are prompted to save your changes to the SMS.
9. Click Yes. A limited version of VR-Forces opens. It has a green terrain grid.
10. Use the object creation palettes to add objects to the SOG as you would for a scenario. You can change to Stealth observer mode if you want to view the terrain in 3D. It is a flat terrain.



When you add objects to an SOG, they are placed on overlays, just as when you add objects to a scenario. There may be cases where you add tactical graphics to an SOG, but do not want them to be displayed when you place the SOG in a scenario. For example, you might put routes on the deck of an aircraft carrier, but do not want to clutter up the display with them. If you edit the name of the tactical graphics overlay and put an asterisk (*) in front of the name, when the SOG is placed, the overlay will be hidden. For more information, please see "Hiding an Overlay by Default," on page 39-4.

11. Choose **File** → **Save Object Group** to save the SOG and continue editing. Choose **File** → **Save Object Group and Exit** to save the SOG and close VR-Forces.

68.5.1. Setting the Simulation Connection when Creating an SOG

When you create an SOG, the Simulation Object Editor starts VR-Forces so that you can specify the members of the SOG. By default it uses the DIS localhost connection for *EntityLevel.sms* and the HLA 1516 Evolved RPR 2.0 with MAK extensions connection for *AggregateLevel.sms*. You can change the connection if you want to. (Aggregate-level scenarios must use HLA 1516 Evolved RPR 2.0 with MAK extensions.)

To set the simulation connection when specifying the members of a simulation object group:

1. In the Simulation Object Editor, choose **Edit** → **Object Group Connection**. The Choose Connection dialog box opens.
2. Select the connection you want to use.
3. Click OK.

68.6. Editing Tactical Graphics

You can add tactical graphics and edit existing tactical graphics just like you can add and edit simulation objects. You can create a new tactical graphic from an existing one or create a new tactical graphic using one of the basic graphic types - point, route, area, symbol, box, or circle.

When you edit a tactical graphic, you can change the default characteristics of that tactical graphic. When VR-Forces users create one of these graphics, they can edit its specific characteristics as desired.

The basic procedures for adding and editing tactical graphics are the same as for adding and editing simulation objects. This section describes procedures that are specific to tactical graphics.

68.6.1. Editing a Tactical Graphic's Menu Icon

Most objects on the Create menu and the various object creation palettes have icons associated with them. For simulation objects, the icon is the icon used to represent them in the front-end. For tactical graphics, the icon illustrates the type of graphic. You can change the icon associated with a tactical graphic and add icons for newly created tactical graphics.

➤ To specify the palette icon for a tactical graphic, on the General tab, click the Browse button (…) for the Menu Icon line (Figure 68-1) and select the icon that you want to use.

➤ To remove the icon associated with a tactical graphic, click the Delete button (–) on the Menu Icon line.

68.6.2. Setting the Default Values for Tactical Graphics Properties

When you select a tactical graphic, the Visual Model group box displays all of the properties that can be set for all of the different types of tactical graphics. Properties that do not apply to the selected tactical graphic are disabled. You can set default values for all of the properties that are applicable to the selected graphic. Setting a default value does not mean that the property will be editable in the VR-Forces front-end. The properties that are editable in the front-end are determined by keywords. For details, please see "Specifying the Properties that Can Be Edited in the Front-End," on page 68-32.

To set a default value for a tactical graphic property:

1. In the Objects List, select the tactical graphic that you want to edit.
2. In the visuals panel, click the Edit button (). The Edit Tactical Graphics 2D Icons dialog box opens (Figure 68-13).

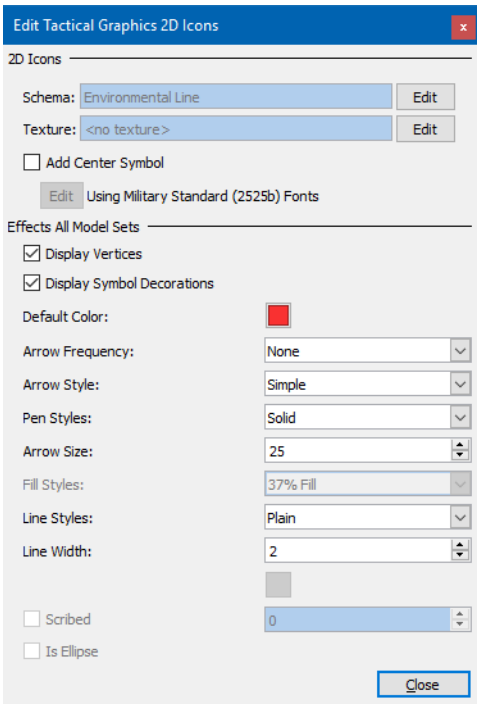


Figure 68-13. Edit Tactical Graphics 2D Icons dialog box

3. The applicable properties are enabled.
4. Set the default values.

68.6.3. Specifying the Properties that Can Be Edited in the Front-End

When you create a tactical graphic in the front-end, you can edit its properties, such as line style, line width, color, and fill. The properties that you can edit in the front-end are determined by keywords that you specify in the Simulation Object Editor.

The keywords are as follows:

- min=<num>. Specifies the minimum number of points required to create the object.
- max=<num>. Specifies the maximum number of points allowed.
- nostyle. Do not allow style editing.
- freehand. The object can be drawn freehand.
- penstyle. The user can change line style for the object.
- fill. The user can change the object's fill.
- arrow. The user can change the object's arrow style.
- arrowsize. The user can change the object's arrow size.

When you edit a tactical graphic in the front-end, VR-Forces builds an Edit Style dialog box with the properties specified by the keywords. If no editing is allowed, it removes the Edit icon from the Edit Object dialog box.

➤ To allow a tactical graphic's properties to be edited, on the General tab, type the appropriate keyword in the Keywords text box. Separate each keyword with a semi-colon (;).

68.6.4. Editing the Visual Model for a Tactical Graphic

The visual model for a tactical graphic is based on a schema or on a texture. Schemas employ vector graphics and are best for basic lines, points, and areas. Textures apply raster images and are used primarily for specialized linear and areal graphics, such as fortified lines and minefields. (If you use a texture for a graphic, you cannot edit its style properties.)

To edit the visual model for a tactical graphic:

1. In the list of tactical graphics, select the one that you want to edit.
2. In the Visual Model panel, in the Model Sets list, select the model set that you want to edit.
3. Click the Edit button (). The Edit Tactical Graphics *model_set* dialog box opens (Figure 68-14), where *model_set* is the model set you selected.

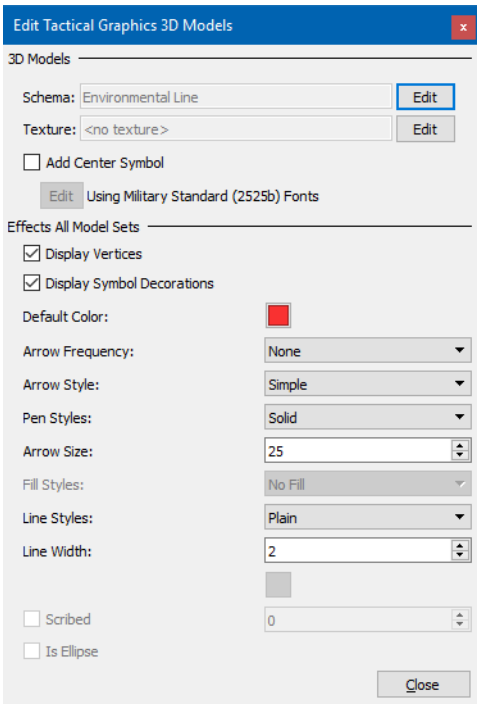


Figure 68-14. Edit Tactical Graphics Models dialog box

4. On the Schema line, click the Edit button. The Select Schema for Model Set *model_set* dialog box opens (Figure 68-15).

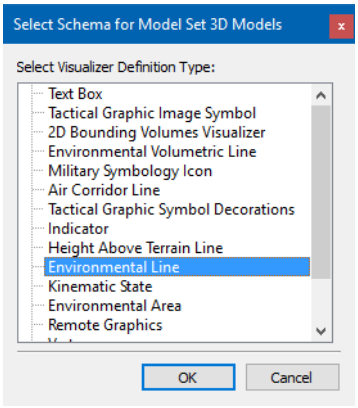


Figure 68-15. Select Schema for Model Set dialog box

5. Select the schema that you want to use.
6. Click OK.

Editing the Texture for a Tactical Graphic

When you use a texture for a tactical graphic you specify a model definition and the height and width of the texture.

To specify a texture for a tactical graphic:

1. In the list of tactical graphics, select the one that you want to edit.
2. In the Visual Model group box, in the Model Sets list, select the model set that you want to edit.
3. Click the Edit button (). The Edit Tactical Graphics *model_set* dialog box opens (Figure 68-14), where *model_set* is the model set you selected.
4. On the Texture line, click Edit. The Select Texture dialog box opens (Figure 68-16).

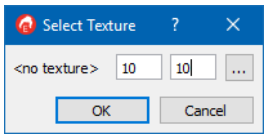


Figure 68-16. Select Texture dialog box

5. Click the Browse button (...). The Select Model Definition dialog box opens (Figure 68-17).

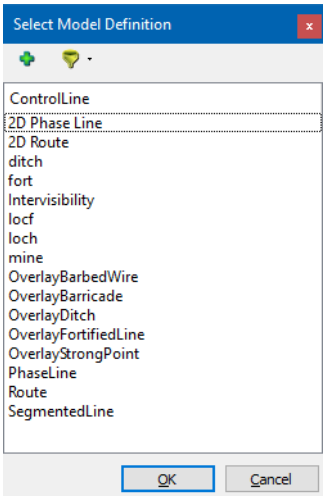


Figure 68-17. Select Model Definition dialog box

6. Select the model definition that you want to use for this texture.
7. Click OK.
8. In the Select Texture dialog box, specify the width and height of the texture, in pixels.
9. If this is an areal graphic, specify whether or not to use the texture for the outline of the area. If you do not use the texture, a regular line is used.

10. Click OK.

68.7. Creating a Template

When you create a new simulation object, you can select a template to base it on. You can create templates from existing simulation objects. In many ways the result is not much different from creating a new model from an existing model. However, when you create a template you can edit it to be more generic than a particular simulation object, which makes it more broadly applicable. Templates are particularly useful for creating aggregate-level units because they are so much more complex than simulation objects in entity-level scenarios.

Templates are listed at the end of the objects list. They are named with the format [Template] *object_name*.

To create a template:

1. In the objects list, select the simulation object that you want to use as a template.
2. Choose **Model** → **Make Template**. A template is created.
3. Optionally, select the template and edit it as you would any simulation object.

68.8. Editing Systems in the Simulation Object Editor

You can create and edit systems in the Simulation Object Editor. The support for different system types varies. Details of the use of the SOE to edit particular systems is described in the chapters devoted to those system types.

When you select a system, the SOE presents information and options on one or more tabs. All systems display the General tab. It includes the following information:

- Name. System name.
- Description. A description of the system.
- Supported entity kinds. A list of the entity types that can use this system.
- Objects that use the system. This list is read-only. It updates when you add or remove a system from a simulation object.
- Parameters. A list of the parameters that you can edit for individual simulation objects. This field is read-only.
- Tags. The tags that you want to apply to the system.

Most systems have a set of parameters that have default values. When you add a system to a simulation object, you can edit the parameters for that simulation object. The Attributes tab lists the system parameters for which you can set default values. Some systems have additional tabs for parameters, largely as a convenience to keep the window size manageable.

Weapon systems have a Targets tab that lets you configure hit probabilities, ammunition, and other targeting details. Damage systems have a Damage tab on which you can configure how entities get damaged by different power types.

Munitions are on a separate tab in the Objects List Panel. They have a Detonation tab for configuring the power types supported by the munition.

The Damage Probabilities dialog box helps you understand the impact of different types of weapons on different types of entities.

68.8.1. Creating a New System

You can create a new system by copying an existing system or by basing it on a template. Templates represent a type of system that has a set of parameters that all systems of that type typically have. The choice between creating a new system by copying an existing one or using a template is a matter of estimating how much editing of the system you will have to do once you create it. For example, the parameter values for a 130mm gun are probably very similar to those of a 125mm gun, in which case you might just want to copy the 125mm gun. But they might be quite different for a 70mm gun, in which case you might want to create the new system from a template.

Creating a New System from an Existing System

If you want to create a new system that is very similar to an existing system, you can copy the existing system to a new system and then edit the parameters.

To create a new system from an existing system:

1. Select the Systems tab.
2. Select the system that you want to copy.
3. Choose **System** → **New System from Existing**. The New System from Existing dialog box opens.
4. In the System Name box, type a name for the new system.
5. In the System Filename box type a filename for the new system or click the Browse button and select a system definition file. If you select a system definition file, the new system will overwrite the existing system.
6. Click OK. The new system is added to the system list. Edit the system as desired.

Creating a New System from a Template

If you want to create a system that has all the basic elements of a certain type of system, but which will not be very similar to an existing system, you can create it from a template.

To create a new system from a template:

1. Choose **System** → **New System**. The New System from Template dialog box opens.
2. In the System Type list, select the type of system you want to create. If the type of system you want to create is not in the list, VR-Forces does not have templates for that system type.

3. In the System Template list, select the template that you want to use for the new system. The dialog box provides a brief description of the selected system type. It lists the systems that use the selected template and they simulation object types that support that system.
4. In the System Name box, type a name for the new system.
5. In the System Filename box, accept the default system definition filename or type the name you want to use.
6. Click OK. The new system is added to the system list.
7. Edit the parameters of the system.

68.8.2. Editing Probability Tables

Some systems, such as weapon systems and damage systems, use probability tables to determine the likelihood that a weapon will strike an entity, an entity will be damaged by a munition, and so on. These tables can be specified as a range list or a coefficient list.

A range list is a step function, a function that increases or decreases abruptly from one constant value to another. A coefficient list is a continuous function. Its values change according to a curve specified by a polynomial. You can specify the probabilities by editing a table. Once you create a table you can edit it directly or graphically.

Figure 68-18 illustrates the following range list:

```
(range-list
  (range 1000.000
    (probability 0.9000)
  )
  (range 2000.000
    (probability 0.8000)
  )
  (range 3000.000
    (probability 0.7000)
  )
  (range 4000.000
    (probability 0.7000)
  )
)
```

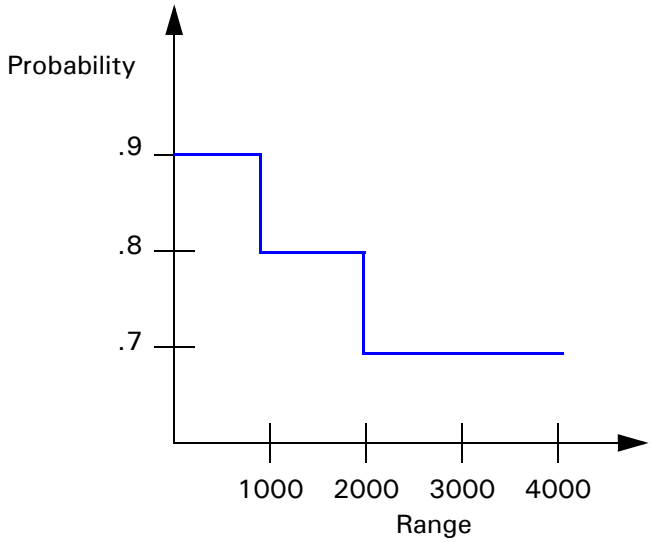


Figure 68-18. Range list (step function)

Figure 68-19 illustrates the curve specified by a coefficient list. The hit probability decreases along the curve rather than at specified steps. (If a coefficient list only has two points, it is graphed as a straight line, not a curve.)

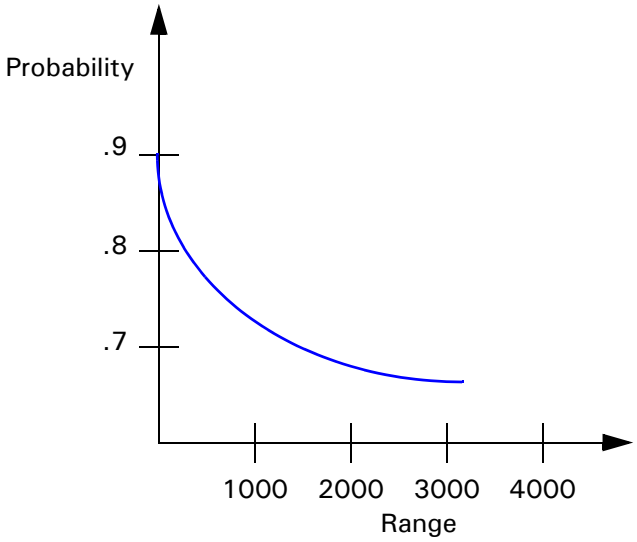


Figure 68-19. Coefficient list polynomial curve

Editing a Range List Probability Table

Range lists are specific to a particular target type, damage power, or other system parameter. This procedure assumes that you have selected the system parameter for which you want to edit a range list.

To edit a range list table:

1. To add a column to the table (that is, add a range value):
   a. Click the Add button (+) above the table. The New Range dialog box opens.
   b. Type a range value.
   c. Click OK. A column is added to the table. It is automatically sorted to fit in the correct location relative to the other columns.
2. To edit a a range value:
   a. Select the column whose range you want to edit. (If you select a row, the column defaults to the first column.)
   b. Click the Edit button (✎) above the table. The Edit dialog box opens. (You can also double-click the range to open the Edit dialog box.)
   c. Type a range value.
   d. Click OK. The range in the column heading is changed. The table is resorted if necessary.
3. To edit the probability for a given range:
   a. Select the probability value.
   b. Type a probability as a whole number, for example, for 30%, type 30.

Editing a Range List by Editing its Graph

When you edit a range list graph, you can change the range values and percentage for existing ranges. You cannot add new ranges to the table.

If a system has more than one parameter type, there is a graph for each type. You can edit each of them individually. (Depending on range settings, the graphs might overlap.)

Each graph has a different color. You can change the colors assigned to each graph.

To edit a range list graph:

1. Move the mouse cursor over the graph. It change to a cross-hair.
2. Hold down the mouse button above the horizontal or vertical portion of the graph that you want to edit.
3. Drag the line to the new value. The value in the corresponding cell in the table changes as you drag the line.

Editing a Coefficient List Probability Table

The columns in a coefficient list probability table have predetermined values: 1, r, r^2, r^3, and so on. You can set the value for each column.

To edit a coefficient list table:

1. To add a column to the table (that is, add a coefficient value), Click the Add button (+) above the table. A new column is added to the table.
2. To edit the probability for a given coefficient:
   a. Select the probability value.
   b. Type a probability.

Changing the Color of a Probability Graph Line

Each system parameter that is being graphed is listed below the horizontal label for the graph. A color swatch next to the parameter name indicates the color used for that line in the graph.

To change the color used for a line in a probability graph:

1. Click the label for the parameter whose color you want to change. A color picker dialog box opens.
2. Select the color you want to use.
3. Click OK.

Changing the Type of List for a Probability Table

You can change the type of probability table you use for a system parameter, as follows:

- To change from a range list to a coefficient list, click the Coefficient List button () above the table. Click Yes in the confirmation dialog box.
- To change from a coefficient list to a range list, click the Range List button () above the table. Click Yes in the confirmation dialog box.

68.8.3. Deleting a System

To delete a system:

1. Select the system in the system list.
2. Choose **System** → **Delete System**. The system name is struck-through.
3. Save the SMS.

68.8.4. Reverting a System

If you edit a system and have not yet saved the SMS, you can revert all changes to the system since the SMS was last saved.

To revert changes to a system:

1. Select the system in the system list.
2. Choose **System** → **Revert System**.
3. Save the SMS.



69. Editing Entities for Entity-Level Scenarios

This chapter explains how to use the Simulation Object Editor to edit individual objects in *EntityLevel.sms*. Many of the concepts and procedures also apply to units. For details about editing units, please see Chapter 70, *Editing Units for Entity-Level Scenarios* and Chapter 71, *Editing Aggregate-Level Objects*.

Editing Simulation Object Parameters 69-3
- Editing a Simulation Object's Bounding Volume (Size) 69-3
- Displaying a Simulation Object's Bounding Volume 69-4
- Specifying Support Points 69-5
- Excluding Entities from Batch Bounding Volume and Support Point Updates 69-6
- Editing Behavioral Parameters 69-6
- Adding Object Geometry to a Simulation Object 69-7
- Configuring Cultural Features To Be Embedded in the Terrain ... 69-9
- Ground Clamping Cultural Features 69-10
- Editing a Lifeform's Visual Model and Animation 69-11

Editing Sensor Signatures 69-14

Editing a Simulation Object's Systems 69-15
- Adding a System 69-16
- Selecting a Damage Model 69-17
- Selecting a Movement Model 69-17
- Editing a System's Properties 69-18
- Removing a System 69-18

Creating and Editing Munitions 69-19
- Creating a Simulated Munition 69-19
- Creating a New Ammunition Type 69-20

Configuring Wakes 69-20

Creating Randomized Simulation Objects 69-22

Configuring Scripted Entity Movement 69-24
Configuring Ballistic Missile Movement 69-25
Configuring Animated Movement 69-27
Editing Movement File Configurations 69-29

69.1. Editing Simulation Object Parameters

This section explains how to edit parameters that are only applicable to simulation objects and cultural features. (Cultural features are objects that you can add to a scenario to simulate human artifacts such as buildings, bridges, and so on. They are modeled as simulation objects because they exist in the physical world.)

69.1.1. Editing a Simulation Object's Bounding Volume (Size)

VR-Forces uses a simulation object's bounding volume to do intersection tests. The bounding volume is specified by the values in the Size/Supports section of the Physical tab. You can display a simulation object's bounding volume in the Visual Model panel. If you change size values, the bounding volume changes accordingly.

You can set the bounding volume in the following ways:

- Edit the value manually.
- Calculate the bounding volume of a particular entity based on the model.
- Calculate the bounding volumes for all entities based on their models.



- The bounding volume is not the same thing as the selection box that is displayed around a simulation object when you select it. The selection box is just part of the front-end display. The bounding volume is used by the simulation engine.
- You can exclude individual entities from the global updates.
- VR-Forces calculates the bounding volume for units based on the locations and bounding volumes of the members of the unit.

To change an entity's bounding volume manually:

1. Select the entity you want to edit in the object list.
2. Select the Physical tab (Figure 69-1).

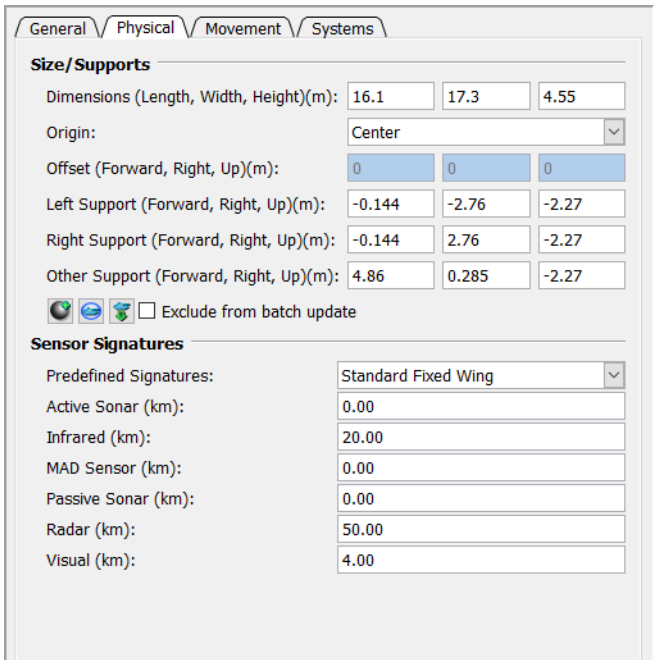


Figure 69-1. Physical tab

3. Change any of the values in the Size/Supports section.

➤ To calculate the bounding volume of an entity from its model, click the Calculate Support Points and Bounding Volume from Displayed Model button ( ) or the Calculate Bounding Volume from Displayed Model button ( ).

➤ To calculate the bounding volume for all entities, choose **Model** → **Compute Bounding Volume and Support Points**.

69.1.2. Displaying a Simulation Object's Bounding Volume



You must run VR-Forces at least once before you can display bounding volumes in the Simulation Object Editor.

To display a simulation object's bounding volume:

1. Select the entity in the object list.
2. In the Visual Model Panel, on the Model Sets list, select 3D Models.
3. Choose **Preferences** → **Show Bounding Volumes**. The bounding volume is displayed (Figure 69-2).

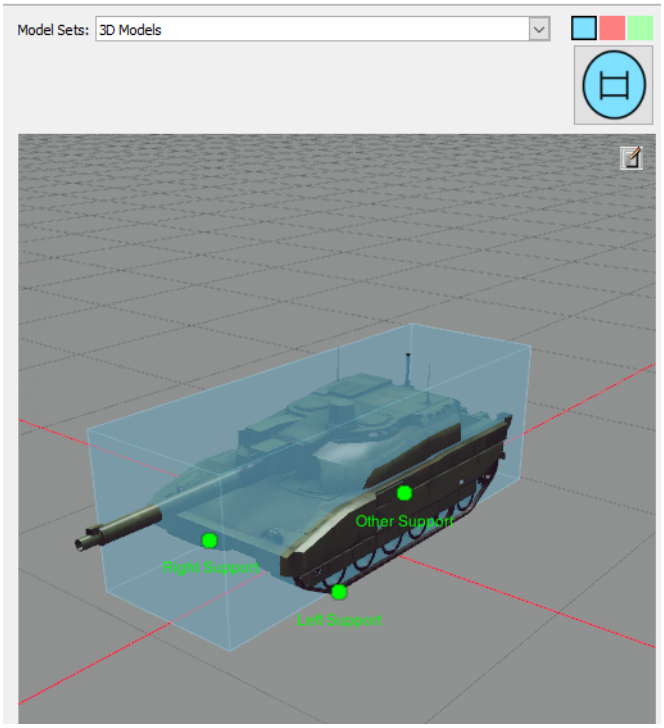


Figure 69-2. Bounding volume

69.1.3. Specifying Support Points

Support points are the locations on a model that contact the ground. You can specify support points in any of the following ways:

- Set them manually.
- Calculate them from the 3D model.
- Calculate the support points for all entities based on their models.

To specify support points manually:

1. Select the entity you want to edit in the object list.
2. Select the Physical tab (Figure 69-1).
3. In the Size/Supports section type the desired values.

➤ To calculate support points based on the model, in the Size section, click the Calculate Support Points and Bounding Volume from Displayed Model button ( ) or the Calculate Support Points from Displayed Model button ( ).

➤ To calculate the support points for all entities, choose **Model** → **Compute Bounding Volume and Support Points**.

69.1.4. Excluding Entities from Batch Bounding Volume and Support Point Updates

If you run the Compute Bounding Volume and Support Points command, it updates all entities. If you have specified the bounding volume or support points for an entity manually, this operation might override those settings. You can exclude individual entities from the global update process.

To exclude an entity from the global update process:

1. Select the entity in the object list.
2. Select the Physical tab (Figure 69-1).
3. In the Size/Supports section, select the Exclude From Batch Update check box.

69.1.5. Editing Behavioral Parameters

The Simulation Object Editor lets you edit some of the parameters that affect simulation object behavior.



The Sensor Signature parameters specify the distance at which other simulation objects can sense the simulation object you are editing. By contrast, the sensor systems on the simulation object control its ability to sense other simulation objects. For more information, please see "Editing a Simulation Object's Systems," on page 69-15.

➤ To edit simulation object parameters, edit any of the parameters in the Physical and Movement tabs. (The tabs reconfigure as you select different types of simulation objects.)

69.1.6. Adding Object Geometry to a Simulation Object

Object geometry is to a simulation object as terrain is to the earth. It allows embarked entities to move about on the entity on which they are embarked and it allows entities to move about on cultural features that are embedded in the terrain or embarked on other simulation objects. For example, adding object geometry to an aircraft carrier lets planes taxi on the deck of the carrier. You can add object geometry to simulation objects in the Simulation Object Editor. Object geometry must be in OpenFlight, MEDF, or GDB format. OpenFlight geometry gets converted to GDB and saved. For complete details about adding object geometry and generating navigation data, please see "Generating Navigation Data for Entities," on page 61-17.



When you edit an entity's embarkation points and slots, the model used is the visual model. This will not necessarily line up with the object geometry for that entity, if it has any.

To add object geometry to a simulation object:

1. Select the simulation object in the Object List.
2. Select the Movement tab (Figure 69-3).

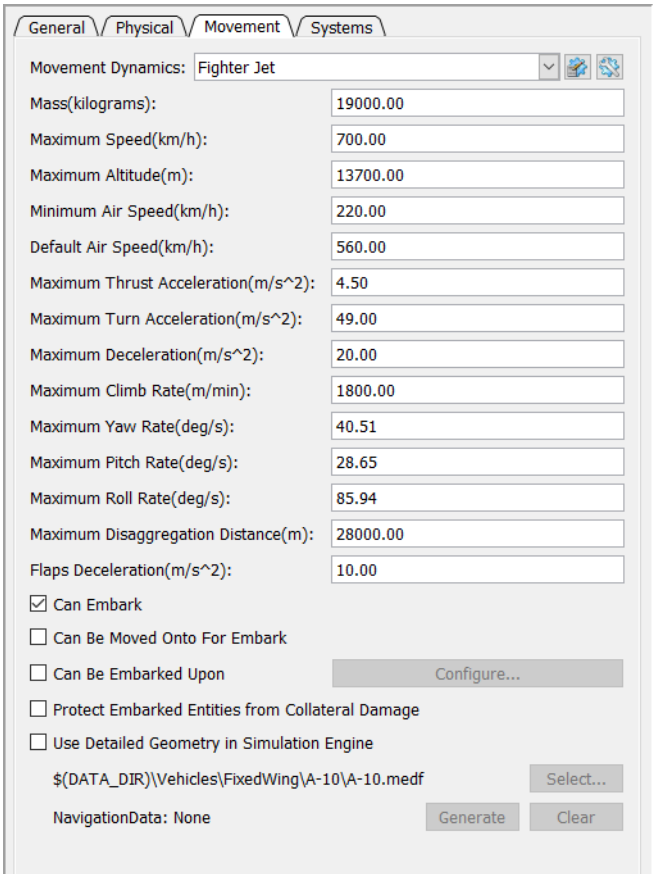


Figure 69-3. Movement tab

3. Select Use Detailed Geometry in Simulation Engine.
4. Click Select. The Select Object Geometry dialog box opens.
5. Click the Browse button. The Object Geometry dialog box opens.
6. Select the file that contains the object geometry for this entity.
7. Click Open.
8. Click OK.

69.1.7. Configuring Cultural Features To Be Embedded in the Terrain

If a cultural feature, such as a house, has object geometry, you can specify that it can be embedded in the terrain. When you place such an object into a navigation area, VR-Forces automatically generates navigation data for that object. This allows entities to walk around and inside the building (if it has interior geometry) as if it were part of the terrain. For more information, please see "Embedding Cultural Features in the Terrain," on page 14-34.



You do not need to generate navigation data in the Simulation Object Editor for cultural features that you want to embed in the terrain. The navigation data is generated when you place the cultural feature in a scenario. Generating navigation data for an object in the Simulation Object Editor is for when you want to embark an entity onto that object.

To allow a cultural feature to be embedded in the terrain:

1. Select the cultural feature in the Objects List.
2. Select the Movement tab (Figure 69-3).
3. Select Use Detailed Geometry in Simulation Engine.
4. Click Select. The Select Object Geometry dialog box opens.
5. Click the Browse button. The Object Geometry dialog box opens.
6. Select the file that contains the object geometry for this entity.
7. Click Open.
8. Click OK.
9. Select Embed Object Geometry Into Terrain.

69.1.8. Ground Clamping Cultural Features

You can specify that cultural features be aligned to the terrain (ground clamped). If they are not aligned to the terrain they might end up sticking out at odd angles.

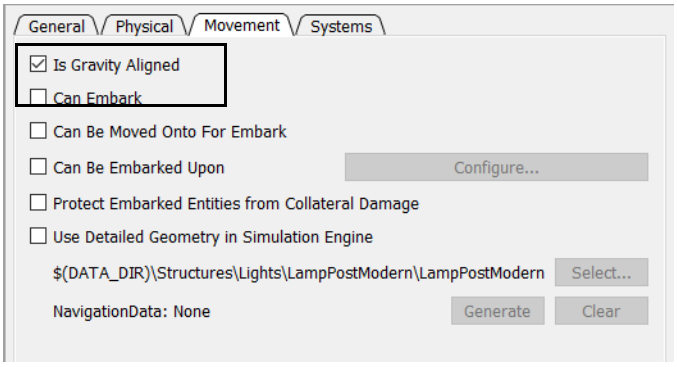


Figure 69-4. Is Gravity Aligned check box

➤ To align cultural features to the terrain, on the Movement tab, select the Is Gravity Aligned check box (Figure 69-4).

69.1.9. Editing a Lifeform's Visual Model and Animation

VR-Forces can specify the character, appearance, hand item, head, and animation values for DI-Guy models used in the 3D view and visualization applications such as VR-Vantage. A model's character determines the kinds of movement it can make. The appearance determines what it looks like. The animation specifies the movements that the character makes. In the VR-Forces front-end, you can specify the animation with the DI-Guy Animation task. You can set the appearance with the DI-Guy Appearance set data request. This feature is valid only for lifeforms (the Human and Animal tags).

To specify the default DI-Guy characteristics for a lifeform:

1. Select the Human or Animal tag.
2. Select a lifeform in the object list.
3. In the Visual Model Panel (Figure 69-5), on the Model Sets list, if 3D Models is not selected, select it.

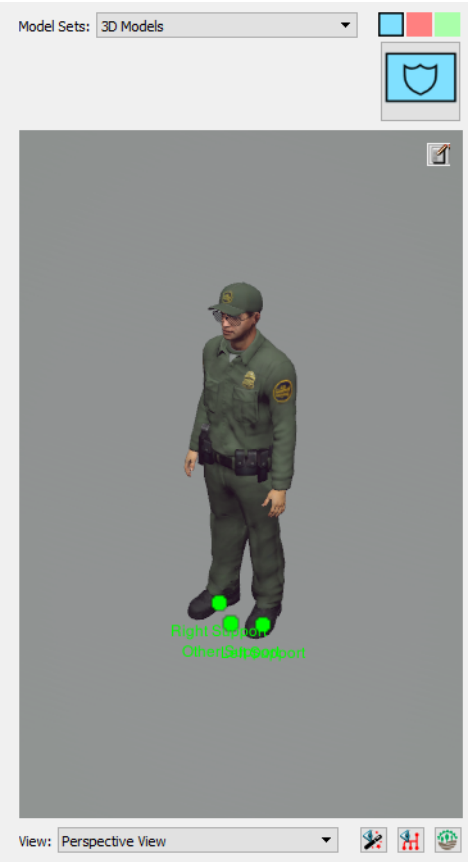


Figure 69-5. Visual Model Panel

4. In the 3D display, click the Edit button ( ). The Set 3D Models Visuals dialog box opens (Figure 69-6). The Use DI-Guy option should be selected. If it is not selected, select it.

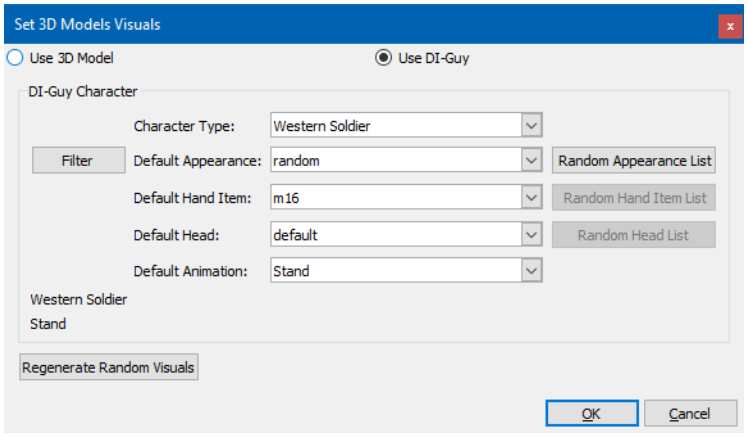


Figure 69-6. DI-Guy Character group box

5. Optionally, filter the options available in the lists, as follows:
   a. Click Filter. The DI-Guy Filters dialog box opens.
   b. Select an option for each characteristic that you want to filter on.
   c. Click OK.
6. Select the DI-Guy character you want to use from the Character Type list.
7. Select the default appearance from the Default Appearance list. One of the default options is to use a random appearance.
8. If you selected random appearance, optionally, specify the list of appearances to choose from. For details, please see "Configuring the Random DI-Guy Characteristics Lists," on page 69-13.
9. Optionally, in the Default Hand Item list, select a hand item. If you choose random, click Random Hand Item List to specify the hand items that can be chosen.
10. Optionally, in the Default Head list, select a head. If you select random, you can specify the list of heads to choose from. If you choose random, click Random Head List to specify the heads that can be chosen.
11. Select the default animation from the Default Animation list. The default animation is the animation the entity uses if it is not tasked and if it has a posture and speed that is appropriate to the animation.

Configuring the Random DI-Guy Characteristics Lists

When you configure the visual model for a DI-Guy, you can specify that a life-form use a random appearance, hand item, head, or some combination of these. When VR-Forces creates a lifeform, it chooses from the random characteristic lists. By default, a random list has all options available for the chosen character. However, you can specify a subset of the available characteristics to use.

To configure the DI-Guy characteristics lists:

1. Select the Human tag (or Animal tag).
2. Select a lifeform in the object list.
3. In the Visual Model Panel (Figure 69-5), on the Model Sets list, if 3D Models is not selected, select it.
4. In the 3D display, click the Edit button (). The Set 3D Models Visuals dialog box opens (Figure 69-6). The Use DI-Guy option should be selected. If it is not selected, select it.
5. Click the random list button for the characteristic you want to configure. A dialog box opens (Figure 69-7). The options available for the character type are listed.
6. Optionally, filter the list by selecting options in the filter lists.

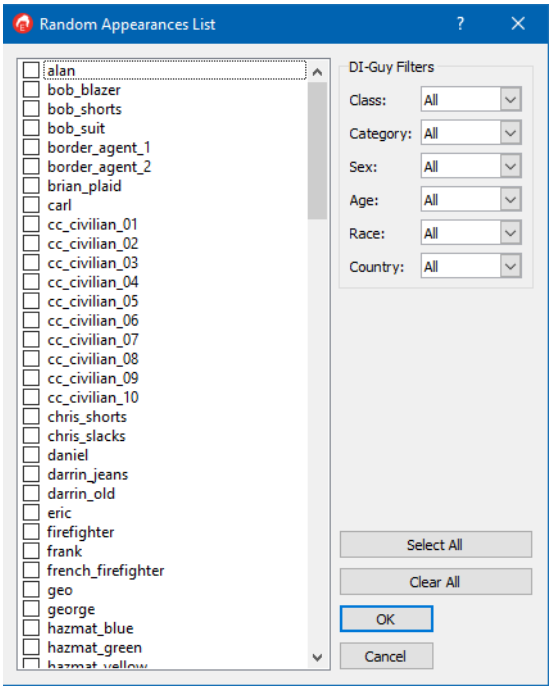


Figure 69-7. Random Appearances List dialog box

7. Select the options that you want to include in the random list.
8. Clear any options that you do not want to include in the random list.

9. Click OK.
10. Click OK to close the Modify Visuals dialog box.

69.2. Editing Sensor Signatures

Sensor signatures specify the distance at which other entity sensors can locate a simulation object. VR-Forces includes a set of predefined sensor signatures for each of the sensor domains supported. You can edit the predefined sensor signatures in *./data/simulationModelSets/EntityLevel/extra/sensor-signatures.csv*.



The VR-Forces simulation engine does not read *sensor-signatures.csv*. The file is only used to set predefined sensor signatures in the Simulation Object Editor. If you make change to the predefined sensor signatures, you must update objects in the Simulation Object Editor to put the changes into effect.

To specify sensor signatures:

1. Select the simulation object.
2. Select the Physical tab (Figure 69-1).
3. In the Sensor Signatures section, specify a value in each domain in which the simulation object can be sensed or select a set of predefined signatures in the Predefined Signatures list.

69.3. Editing a Simulation Object's Systems

Simulation objects have complex capabilities, such as the ability to move, to take damage, to sense other simulation objects, and to fire munitions. These capabilities are organized as systems. You can quickly change how a simulation object functions by adding, removing, and changing its systems. The Simulation Object Editor lists the systems available in each category and only lists systems that are appropriate for the entity type (Figure 69-8). For example, a lifeform can have an M-16 or an RPG, but not a cruise missile.

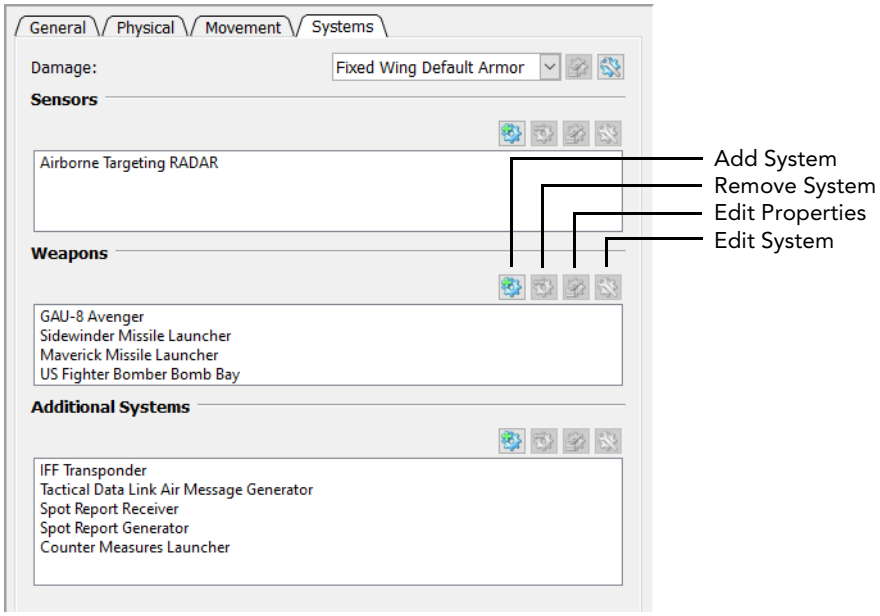


Figure 69-8. Systems tab

Some systems are interdependent. For example, if you add a weapon system to a simulation object, it probably cannot use it unless it also has a sensor system so that it can detect simulation objects. If you change or add systems, it is up to you to know if they have dependencies.

The components that make up a system have parameters and you can edit the parameters of system components by editing the properties of the system.

The Simulation Object Editor lets you edit the following types of systems:

- Damage. You can specify the type of damage model that a simulation object uses. When you select a damage model, the Simulation Object Editor configures the damage files that are appropriate for the object type.
- Sensors. You can specify the sensors that a simulation object uses to locate other simulation objects in the simulation. (By contrast, sensor signatures, which you can also set, specify the distance at which other entity sensors can locate this simulation object. For a list of the sensor signatures used for VR-Forces by category, please see Table 75-1.) When you add a weapon, the Simulation Object Editor configures the components and connections necessary for the sensor to work.
- Weapons. The Simulation Object Editor lists the weapon systems that are appropriate for the object type. When you add a weapon, the Simulation Object Editor configures the components, targets, ammunition, and connections necessary for the weapon to work.
- Movement. You can specify the type of movement system that a simulation object uses, for example, tracked, wheeled, pedestrian, and so on. When you select a movement system, the Simulation Object Editor configures all of the actuators and controllers that the simulation object needs to implement the system.
- Other systems. This category includes spot reports and IFF transponders.

69.3.1. Adding a System

There is no specific limit to the number of sensors, weapons, or other systems that you can configure for a simulation object.

To add a system:

1. Select a simulation object in the list.
2. Select the Systems tab (Figure 69-8).
3. Click the Add System button () for the system type you want to add. The Add System dialog box opens (Figure 69-9). It lists the systems of this type that are available for the simulation object you are editing. If you select a system in the window, the dialog box displays a brief description.

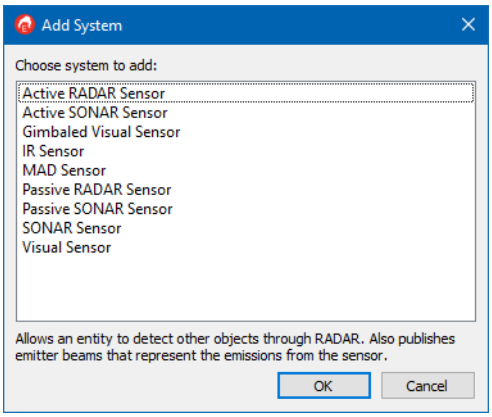


Figure 69-9. Add System dialog box

4. Select the system you want to add.
5. Click OK. The system is added to the list in the Simulation Object Editor.

69.3.2. Selecting a Damage Model

You can only have one damage model specified for a simulation object.

➤ To specify a damage model, on the Systems tab (Figure 69-8), select an option from the Damage list.

69.3.3. Selecting a Movement Model

You can only have one movement model specified for a simulation object.

➤ To specify a movement model, on the Movement tab (Figure 69-3), select an option from the Movement Dynamics list.

69.3.4. Editing a System's Properties

Some systems have properties that you can edit.

To edit a system's properties:

1. Select the simulation object whose system you want to edit.
2. Select the Systems tab (Figure 69-8).
3. For sensors, weapons, or other systems, select the system whose properties you want to edit. (You do not have to select the damage or movement model, because there can only be one.) If the system has properties, the Properties button is enabled.
4. Click the Edit Properties button (). A Properties dialog box opens. The 3D model in the Visual Model Panel displays a red dot on the simulation object where the system is located. In some cases there are multiple locations.
5. Edit the properties as desired.
6. Click OK.

69.3.5. Removing a System

You can remove sensor, weapon, and additional systems.

> If you remove a system that another system depends on, the other system will not work. It is your responsibility to know these dependencies.

To remove a system:

1. On the Systems tab, select the system you want to remove.
2. Click the Remove System button ().

69.4. Creating and Editing Munitions

Munitions fall into two broad categories – simulated munitions and non-simulated munitions. As the name implies, simulated munitions are simulated as individual objects. This category includes missiles, bombs, mines, and hand grenades. Since they are simulated, they have visual and 2D models. They have all of the various tabs for configuring their parameters that other simulation objects have, plus a Detonation tab for configuring their damage power.

Non-simulated munitions are not simulated as individual objects. This category is for ammunition that is fired by some of the direct fire weapons, such as rifles, machine guns, and some ballistic rounds. These munitions have a General tab for basic identifying information and a Detonation tab for configuring their damage power.

69.4.1. Creating a Simulated Munition

Simulated munitions have visual models and are configured just like other simulation objects.

To create a simulated munition:

1. Choose **Model** → **New Munition**. The New Munition dialog box opens.
2. Select the New Munition option.
3. Click OK. The Create New Model dialog box opens.
4. In the Platform Type list, select the platform type for the munition. The Description field provides information about the type of munition represented by this platform.
5. In the Template field, select the template you want to use. It is possible that there are no templates for the platform type.
6. In the Name box, type a name for the munition.
7. Optionally, edit the Entity Type.
8. In the Take Visuals list, select the model you want to use for this munition.
9. Click OK. A new entry is added to the munitions list.
10. If applicable, on the General tab specify the object types that this munition can target.
11. Edit the Detonation power, as described in ("Configuring Detonation Power in the SOE," on page 77-6).
12. Edit other parameters for this munition as desired.

69.4.2. Creating a New Ammunition Type

Ammunition munitions are not simulated as individual objects.

To create a new ammunition type:

1. Choose **Model** → **New Munition**. The New Munition dialog box opens.
2. Select the New Ammunition option.
3. Click OK. The Add New Ammunition dialog box opens.
4. Specify the object type for the new ammunition by selecting options from the lists of typing the object type directly.
5. Click OK.
6. Click OK. A new entry is added to the munitions list.
7. On the General tab specify the object types that this munition can target.
8. If this munition has a tracer type, specify it on the General tab.
9. Edit the Detonation power, as described in ("Configuring Detonation Power in the SOE," on page 77-6).

69.5. Configuring Wakes

You can configure the wake parameters for surface entities in the Simulation Object Editor. The model window shows the ocean state and you can vary the wind speed, entity speed, and choppiness to see how the wake looks under varying ocean conditions (Figure 69-10).

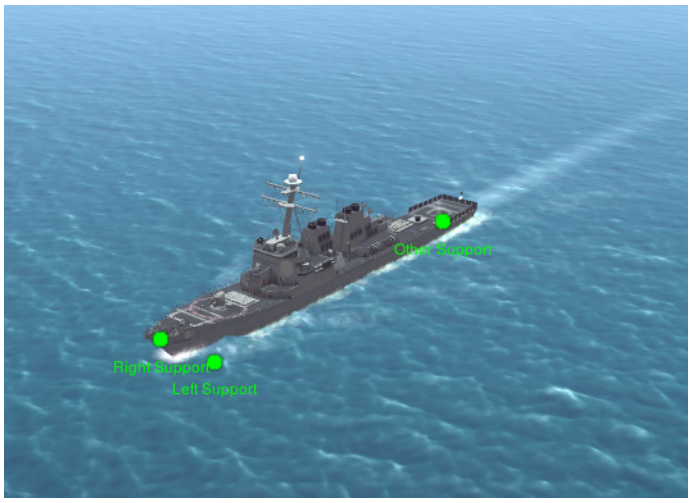


Figure 69-10. Wake display

To configure wakes:

1. Select a surface entity in the object list.
2. Click the Edit Wake button ( ) on the Visual Models Panel (Figure 68-6). The Edit Wake Model dialog box opens (Figure 69-11).

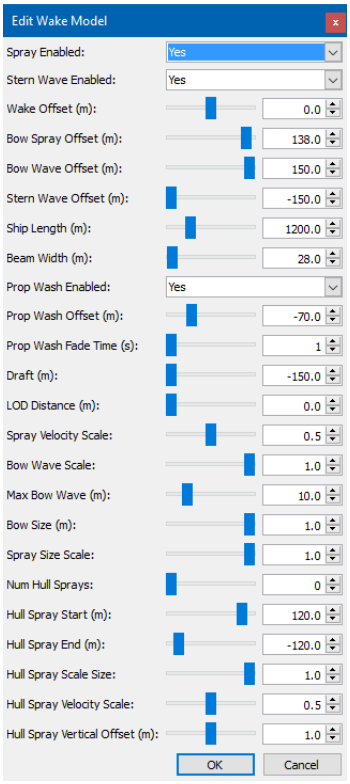


Figure 69-11. Edit Wake Model dialog box

3. Change wake parameter values as desired. For details about wake parameters, please see "Configuring Wakes," on page 90-18.
4. To see the effect of parameter changes under different sea conditions:
   a. Click the Environment Settings button ( ) on the Visual Models Panel. The Environment Settings dialog box opens.
   b. Adjust the wind speed, wind direction, choppiness, and ocean quality values.
   c. Click Close.
5. Click OK.

69.6. Creating Randomized Simulation Objects

A randomized simulation object is an object that you can use to quickly add a generic simulation object type, such as a civilian vehicle or commercial aircraft, using randomly chosen specific models of these object types. For example, if you want to add several cars to a scenario, you could individually select a Nissan Maxima, a Toyota Corolla, a Honda Accord, and so on, and add them to the scenario. Or you could select a randomized simulation object that references a list of civilian vehicles and click multiple times on the terrain, With each click a randomly chosen vehicle would be added to the scenario.

You create randomized simulation objects in the Simulation Object Editor. You can create them in entity-level and aggregate-level scenarios.

To create a randomized simulation object:

1. Start the Simulation Object Editor.
2. Load the SMS you want to use.
3. Choose **Model** → **New Randomized Simulation Object**. The Randomized Simulation Object dialog box opens.
4. Type a name for the simulation object. The new object is added to the object list.
5. Select the new object in the object list (Figure 69-12).

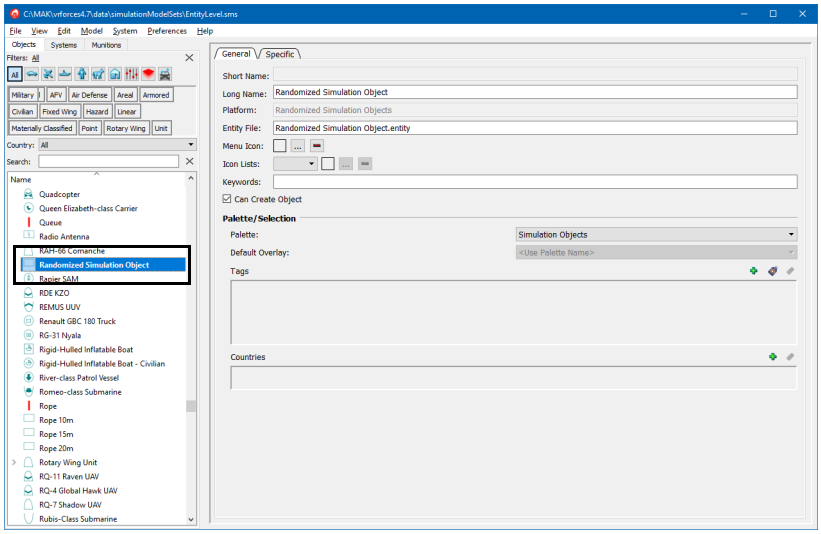


Figure 69-12. Randomized simulation object

6. Optionally, edit the common simulation object attributes, such as tag, palette, default overlay, and so on.
7. Click the Add button (+) (Figure 69-12). The Add to Randomized Simulation Object dialog box opens.

8. Select the simulation objects that you want to be part of this randomized simulation object. To select an object, double-click it or select it and click Add. As you select objects, they are added to the list (Figure 69-13). If you want the probability of a particular simulation object being created to be higher than others, add multiple instances of the same simulation object to the list.

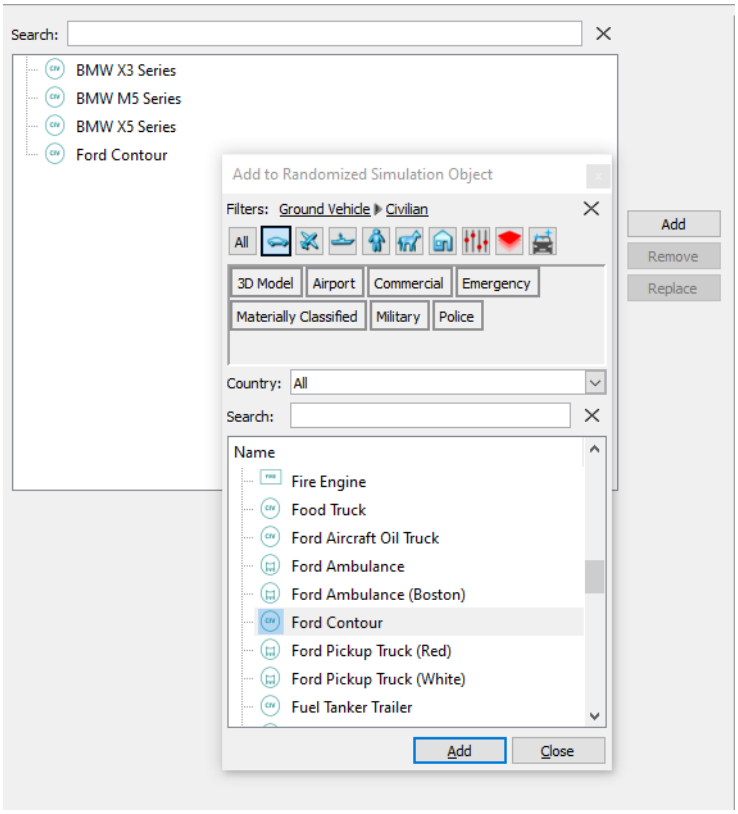


Figure 69-13. Randomized object list

9. Click Close.
10. Save the SMS.

69.7. Configuring Scripted Entity Movement

In most cases, simulation objects move based on the intelligence built into their components – sensors, controllers, and actuators. However, there are two instances in which entity movement follows a predetermined sequence, or script. These cases are ballistic missile movement and animated movement tasks.

When you launch a ballistic missile (by creating a missile target in VR-Forces), it does not follow a route or do any path planning; it flies based on the parameters in the scripts it is configured to use. Similarly, if you give an animated movement task to an entity, it moves in whatever way the script prescribes.

Ballistic missile movement is configured using comma-delimited value (CSV) files. Animated movement tasks can be configured with CSV files or 3DS ASCII Export (ASE) files. We refer to these files generically as movement files.

A movement file specifies a time sequence and a set of parameters that define the entity's movement for each time entry. Ballistic missile movement files define lateral-distance, altitude, and range. Animated movement movement files define lateral distance, altitude, range, heading, pitch, and roll.



Ballistic missile movement files get configured in a dialog box that is shared with animated movement movement files ("Configuring Animated Movement," on page 69-27). Animated movement movement files have more parameters than ballistic missile movement files. If a ballistic missile is configured with a movement file that has these additional parameters, they are ignored.

69.7.1. Configuring Ballistic Missile Movement

When you create a Missile Target event in VR-Forces, you specify a missile type. Missile types are configured in the Simulation Object Editor.

To configure ballistic missiles:

1. Choose **Edit** → **Ballistic Missiles**. The Ballistic Missiles dialog box opens (Figure 69-14).

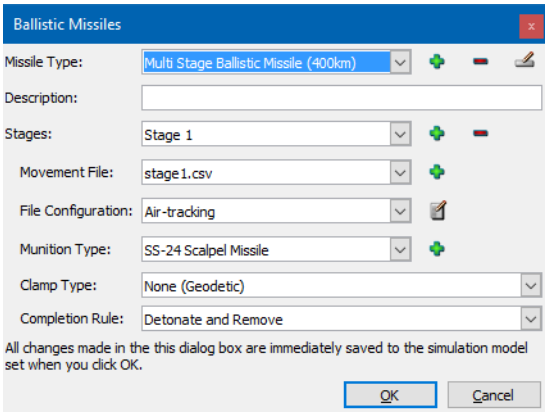


Figure 69-14. Ballistic Missiles dialog box

2. In the Missile Type list, select the missile type that you want to edit. You can add, delete, and rename missile types using the Add (+), Delete (−), and Rename (✎) buttons.
3. Optionally, in the Description text box, type a description. This description gets used as a tooltip for the Missile Type in the Create Missile Target and Edit Missile Target dialog boxes.

4. In the Stages list, select the stage that you want to configure. Missiles can be single stage or multi-stage. For single stage missiles, configure Stage 1. For multi-stage missiles, use the Add and Delete buttons to add or delete stages.

   For each stage, specify the following:

   – Movement File. The CSV file to use. The list lists the files in *./data/simulationModelSets/<model_set>/scriptedObjectMovement*. If you want to use a CSV file that is not listed:

     a. Click the Add button (+) next to the Movement File list. A file chooser dialog box opens.

     b. Select the CSV file that you want to use. The file is copied to *./data/simulationModelSets/<model_set>/scriptedObjectMovement*.

   – File Configuration. Select a file configuration from the list. File configurations specify the type of value in each column in the movement file. For details, please see "Editing Movement File Configurations," on page 69-29.

   – Munition Type. The missile to use.

   – Clamp Type. Determines how to calculate altitude for the animation. Table 69-1 describes the clamp type options.

   – Completion Rule. The action to take when the movement in the CSV file is complete.

Table 69-1: Clamp type options

Option	Description
None (Geodetic)	Altitude in the animation is used as altitude above the altitude of the animation reference location. A constant altitude in the animation results in a path that follows the curvature of the earth, when a round earth terrain is being used.
None (Topographic)	Altitude in the animation is used as altitude above a plane defined by the animation reference orientation. A constant altitude in the animation results in a straight path, even as the earth curves, when a round earth terrain is being used.
High Fidelity (Including Orientation)	Altitude in the animation is ignored, and the object is ground clamped to the terrain using three support points, which also sets the orientation of the object.
Low Fidelity (Terrain Clamp Only)	Altitude in the animation is ignored, and the object is ground clamped to the terrain using only its origin point. Orientation is set solely from the animation.
Altitude Above Terrain (High Fidelity)	Altitude in the animation is used as the altitude above the terrain height.

Table 69-1: Clamp type options

Option	Description
Ground Clamp if Altitude Offset is 0 (High Fidelity)	Acts like High Fidelity for frames where altitude is 0, and acts like None (Geodetic) for frames where altitude is non-zero.
Radar Face Cartesian	Similar to None (Topographic), except the Y coordinate is inverted. The animation would typically be started with an orientation reference corresponding to the angle of a radar face. Useful for cases where animation data is captured radar data.

5. Click OK.

69.7.2. Configuring Animated Movement

The animated movement feature lets you script a sequence of entity movements. You can do this using a CSV file or a 3DS ASCII Export (ASE) file. VR-Forces includes some sample animated movement files.

Animated movement is assigned to an entity as a task from the Movement submenu. You configure the animated movement options in the Simulation Object Editor.

To configure animated movement:

1. Choose **Edit** → **Animated Movement**. The Animated Movement dialog box opens (Figure 69-15).

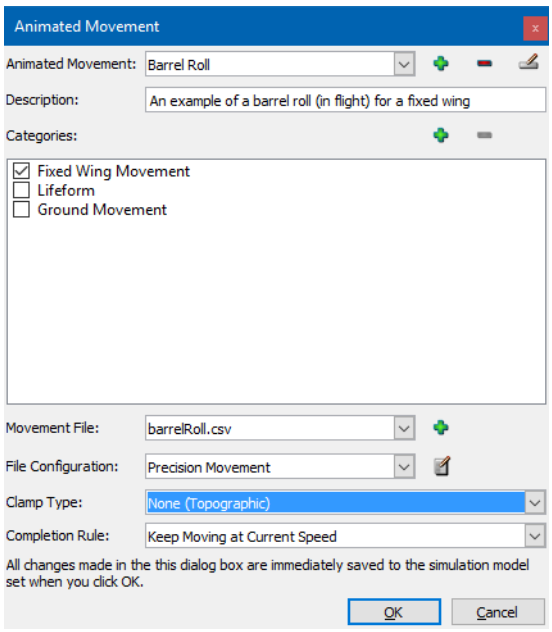


Figure 69-15. Animated Movement dialog box

2. To edit an existing animated movement, select it in the Animated Movement list. To create a new animated movement:
   a. Click the Add button (+) next to the Animated Movement list. The New Configuration dialog box opens.
   b. Type a name for the new animated movement.
   c. Click OK.
3. Optionally, type a description for the animated movement. This description is displayed in the Animated Movement dialog box when an Animated Movement task is assigned.
4. In the Categories window, select the categories that this animated movement applies to. Use the Add and Delete buttons to add and delete categories.
5. In the Movement File list, select the movement file to use. The list lists the files in *./data/simulationModelSets/<model_set>/scriptedObjectMovement*. If you want to use a CSV file or ASE file that is not listed:
   a. Click the Add button (+) next to the Movement File list. A file chooser dialog box opens.
   b. Select the file that you want to use. The file is copied to *./data/simulationModelSets/<model_set>/scriptedObjectMovement*.
6. If you chose an ASE file, the dialog box displays a field called Animation. Select an animation from the list.

   If you chose a CSV file, it displays a field called File Configuration. Select a file configuration from the list. File configurations specify the type of value in each column in the movement file. For details, please see "Editing Movement File Configurations," on page 69-29.
7. In the Clamp Type list, select an option for calculating the altitude. For details, please see Table 69-1.
8. In the Completion Rule list, select the action you want the entity to take after it completes the animated movement task.

69.7.3. Editing Movement File Configurations

VR-Forces assumes that a ballistic missile CSV file contains four columns of data that describe time, lateral distance, altitude, and range. It assumes that animated movement files have seven columns that describe time, lateral distance, altitude, range, heading, pitch, and roll. VR-Forces includes file configurations for the sample CSV files provided. They are organized in the order that the columns are listed in this paragraph. If a CSV file you are using is organized differently, you must create a file configuration that matches your file or edit one of the provided configurations.

To create or edit a movement file configuration:

1. In the Ballistic Missiles dialog box or the Animated Movement dialog box, click the Edit button (✎) next to the File Configuration list. The Edit Movement Data File Configurations dialog box opens (Figure 69-16).

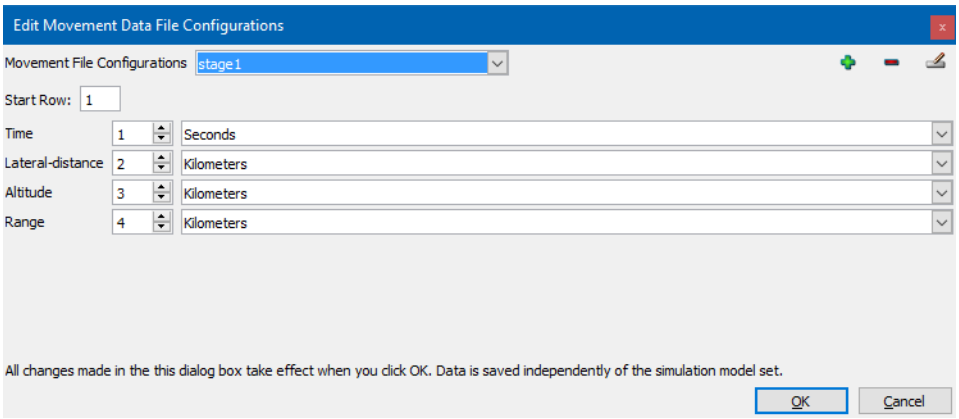


Figure 69-16. Edit Movement Data File Configurations dialog box

---

- Ballistic missile file configurations have four rows. Animated movement file configurations have seven rows. If you specify a seven-row configuration for a ballistic missile, it ignores the values that it does not use.
- If you are editing an animated movement that uses an ASE file, there is no File Configuration option. This only applies to CSV files.

---

2. In the Movement File Configurations list, select the configuration you want to edit. If you are creating a new configuration, click the Add button (✚). The New Configuration dialog box opens.
   a. Type a name for the configuration.
   b. Click OK.

3. Optionally, change the Start Row value. This is the row in the CSV file from which VR-Forces should start reading values for entity movement.
4. Specify the column in the CSV file that has the Time value.
5. Optionally, select the time unit from the list.
6. Repeat steps 4 and 5 for each movement parameter.
7. Click OK.



70. Editing Units for Entity-Level Scenarios

This chapter explains how to edit units for entity-level scenarios (*EntityLevel.sms*) in the Simulation Object Editor.

Introduction to Editing Units 70-2
Creating a Unit for Entity-Level Scenarios 70-2
Specifying an Aggregate As Option 70-3
Editing Unit Composition 70-4
- Adding Subordinates 70-4
- Removing Subordinates 70-5
- Replacing Subordinates 70-6
- Changing the Order of Subordinates 70-6
- Editing a Subordinate Unit 70-7

Editing Formations 70-8
- Adding Formations 70-9
- Removing Formations 70-10
- Renaming Formations 70-10
- Specifying the Default Formation 70-10
- Expanding and Collapsing the Display of Formations 70-10
- Displaying Bounding Boxes 70-11
- Copying Formations 70-12
- Laying Out a Formation Automatically 70-13
- Changing a Formation's Layout by Hand 70-13

70.1. Introduction to Editing Units

VR-Forces can simulate units in two different ways:

- In entity-level scenarios, VR-Forces models the individual entities of a unit. Although you can aggregate them into one simulated model for movement and certain tasks, when they engage in combat, they disaggregate into their leaf-level entities and interact at the individual entity level. Units for entity-level scenarios are configured in *EntityLevel.sms* or SMSs based on it.
- Units in aggregate-level scenarios do not model individual entities. Combat operations are data-driven and are not based on individual entity interactions. Units for aggregate-level scenarios are configured in *AggregateLevel.sms* or SMSs based on it.

Units in entity-level scenarios and aggregate-level scenarios have many of the same parameters as individual entities. Therefore, for the most part, configuring them in the Simulation Object Editor involves the same procedures as those described in Chapter 67, *The Simulation Object Editor and SMSs* and Chapter 69, *Editing Entities for Entity-Level Scenarios*.

The two major differences in configuring the different types of units is that units in entity-level scenarios have subordinates and formations and do not have sensors. Units in aggregate-level scenarios do not move in formation. They have sensors and they have a large amount of information about their combat capabilities, personnel, equipment, and supplies.

70.2. Creating a Unit for Entity-Level Scenarios

To create a unit for entity-level scenarios:

1. Follow the procedures for creating a new entity in "Creating a New Simulation Object," on page 68-26.
2. Add subordinates, as described in "Adding Subordinates," on page 70-4.
3. Optionally, configure formations.

70.3. Specifying an Aggregate As Option

When you create a unit using the Aggregate As command, the Aggregate As dialog box has a list of unit types that you can select for the unit that is to be created. VR-Forces is configured with just a short list of generic unit types. You can specify that any unit configured in the Simulation Object Editor be included in the list of Aggregate As options.



This procedure is valid for entity-level and aggregate-level SMSs.

To specify that a unit can be an option in the Aggregate As list:

1. Select the unit in the object list.
2. Select the General tab (Figure 70-1).

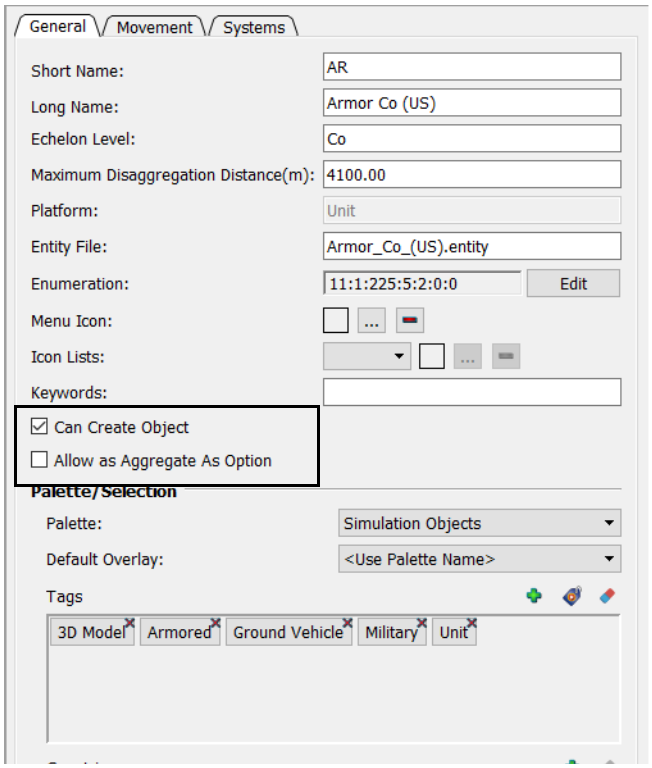


Figure 70-1. General tab, entity-level unit

3. Select the Allow as Aggregate As Option check box.
4. Save the SMS.

70.4. Editing Unit Composition

Each unit provided with VR-Forces has a predefined set of subordinates.

> The Convoy unit provided with VR-Forces is an exception to the discussion of units in this section. It does not have an Auto Aggregation Controller, which allows a unit to aggregate and disaggregate. Although you can assign this controller, it is not recommended, because the resulting behavior would be undefined.

Figure 70-2 illustrates the Simulation Object Editor with a unit selected. The Unit Composition group box is highlighted.

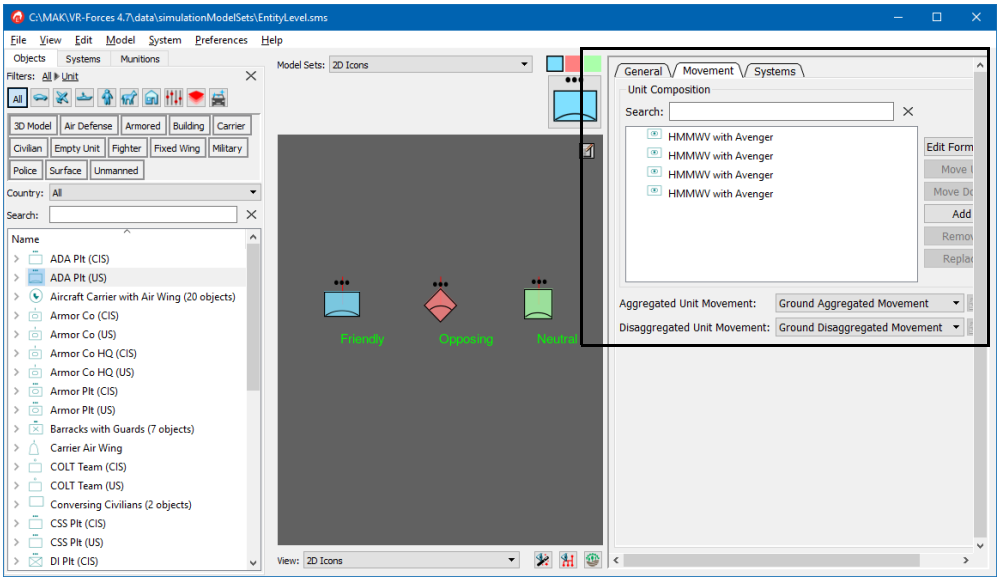


Figure 70-2. Unit Composition

70.4.1. Adding Subordinates

When you create a new unit without copying an existing unit, you must specify its subordinates. You can also edit subordinates for existing units.

To add a subordinate to a unit:

1. In the Objects List, select the unit.
2. Select the Movement tab.
3. In the Unit Composition group box (Figure 70-2), click Add. The Add to Hierarchy dialog box opens (Figure 70-3).

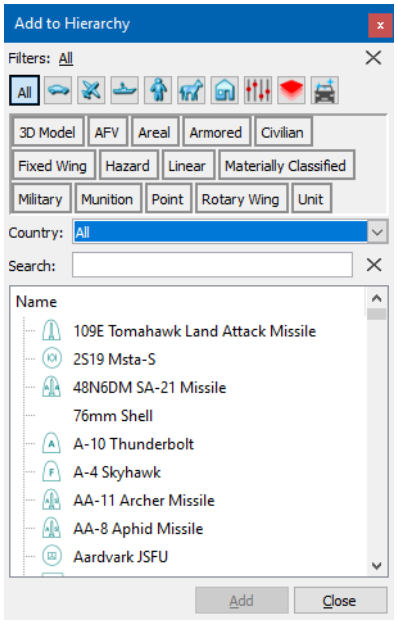


Figure 70-3. Add to Hierarchy dialog box

4. In the list of simulation objects, select the simulation object that you want to add.
5. Click Add. The simulation object is added to the list in the Unit Composition window. It is also added to each formation using a default layout algorithm.
6. Continue to add as many subordinates as desired.
7. Click Close.

70.4.2. Removing Subordinates

To remove a subordinate from a unit:

1. In the Unit Composition window (Figure 70-2), select the subordinate that you want to remove.
2. In the Unit Composition group box, click Remove. The subordinate is removed from the unit.

70.4.3. Replacing Subordinates

Replacing a subordinate is a convenient way to remove a subordinate and insert a different subordinate in the same location in one step.

To replace a subordinate:

1. In the Unit Composition window (Figure 70-2), select the subordinate that you want to replace.
2. In the Unit Composition group box, click Replace. The Add to Hierarchy dialog box opens (similar to Figure 70-3).
3. In the list of simulation objects, select the object that you want to add as a replacement.
4. Click OK. The new object replaces the selected simulation object in the Unit Composition window. It is also added to each formation in the location of the replaced simulation object.

70.4.4. Changing the Order of Subordinates

The order of subordinates in the Unit Composition window determines which entity is the leader, the order in which units are placed into formation, and the succession when units are reorganized.

To change a subordinate's position in the unit composition:

1. In the Unit Composition window (Figure 70-2), select the subordinate that you want to move.
2. In the Unit Composition group box, click Move Up or Move Down.

70.4.5. Editing a Subordinate Unit

Units can contain entities and other units as subordinates. When you are editing a unit, you might decide that you want to edit one of its subordinates. You can jump to the subordinate quickly without selecting it in the main list of simulation objects.

If you edit a subordinate unit, the changes affect all uses of that subordinate, not just the one in this particular unit.

To edit a subordinate:

1. In the Unit Composition window or in the Formation Editor, right-click the subordinate that you want to edit. A menu is displayed.
2. In the menu, choose **Edit**. The Simulation Object Editor redisplays to show the selected subordinate. (If the Formation Editor is open, you must close it before you can edit the subordinate.)

70.5. Editing Formations

Each unit in *EntityLevel.sms* has the following defined formations:

- Column
- Line
- Vee
- Wedge.



The Convoy unit provided with VR-Forces is an exception to the discussion of units in this section. It does not have any formations because there is no preconceived membership in a convoy.

You can add, remove, and rename formations. You can also:

- Change the layout of subordinates in a formation.
- Create new formations.
- Specify the default formation.

The Formation Editor allows you to change the display of formations as follows:

- Expand and collapse the display of subordinate units.
- Zoom in and out on the display.
- Display subordinate bounding boxes.

You can also edit formations in formation files. For details, please see "Configuring Formations," on page 78-2.

70.5.1. Adding Formations

To add a formation:

1. In the Unit Composition group box (Figure 70-2), click Edit Formation. The Formation Editor opens (Figure 70-4).

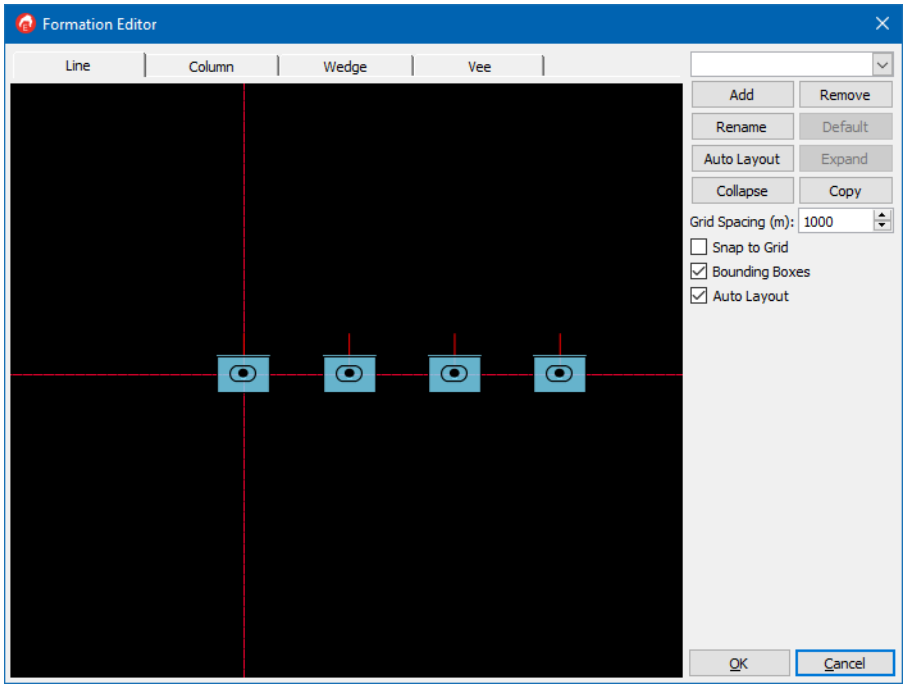


Figure 70-4. Formation Editor

2. In the Formation Editor, click Add. The Add Formation dialog box opens (Figure 70-5).

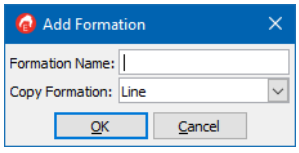


Figure 70-5. Add Formation dialog box

3. In the Formation Name box, type a name for the formation.
4. In the Copy Formation list, select an existing formation to copy the new formation from.
5. Click OK.
6. Edit the layout of the formation as described in "Changing a Formation's Layout by Hand," on page 70-13.

70.5.2. Removing Formations

To remove a formation:

1. In the Unit Composition group box (Figure 70-2), click Edit Formation. The Formation Editor opens (Figure 70-4).
2. In the Formation Editor, select the tab for the formation that you want to remove.
3. Click Remove. The tab is removed.

70.5.3. Renaming Formations

To rename a formation:

1. In the Unit Composition group box (Figure 70-2), click Edit Formation. The Formation Editor opens (Figure 70-4).
2. In the Formation Editor, select the tab for the formation that you want to rename.
3. Click Rename. The New Name dialog box opens.
4. Type a name for the formation.
5. Click OK.

70.5.4. Specifying the Default Formation

The default formation is the formation assigned to a unit when you create it in VR-Forces. The default formation is the leftmost tab in the Formation Editor.

To specify the default formation:

1. In the Unit Composition group box (Figure 70-2), click Edit Formation. The Formation Editor opens (Figure 70-4).
2. In the Formation Editor, select the tab of the formation that you want to specify as the default formation.
3. Click Default. The selected tab gets moved to the leftmost position.

70.5.5. Expanding and Collapsing the Display of Formations

You can expand and collapse the display of formations in the Formation Editor window. This lets you see how simulation objects are dispersed over the terrain. It can help you quickly identify any unintended variations in subordinate formations.

➤ To expand or collapse the display of formations in the Formation Editor, click the Expand or Collapse buttons.

70.5.6. Displaying Bounding Boxes

You can display bounding boxes for the subordinate units in a unit. This may be useful when you collapse the views of individual units, but you still want to have a sense for the area that they occupy. Figure 70-6 illustrates a unit with the display of bounding boxes off and on.

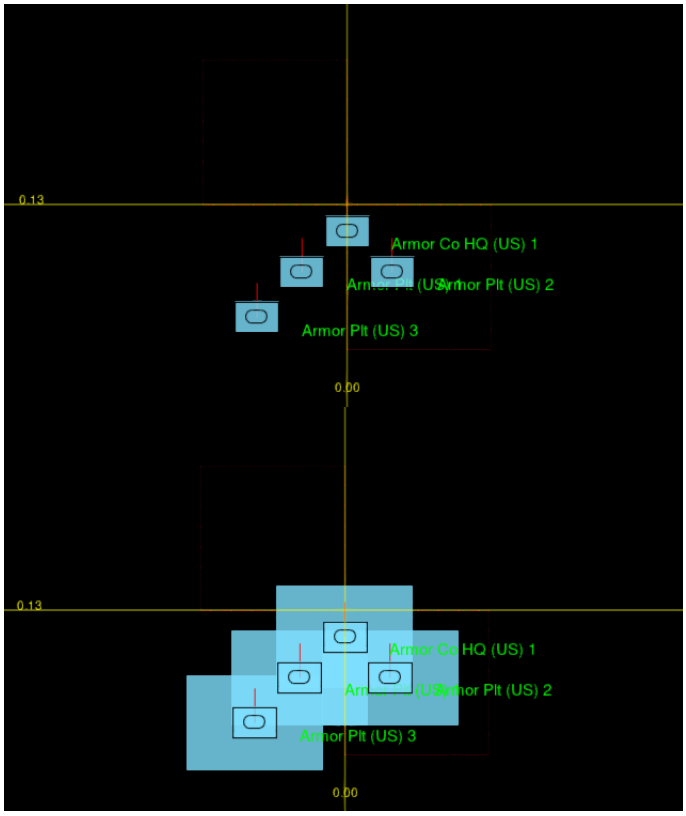


Figure 70-6. Unit bounding boxes in Formation Editor

➤ To display or hide bounding boxes, in the Formation Editor (Figure 70-4), select or clear the Bounding Boxes check box.

70.5.7. Copying Formations

You can copy formations from one unit to another. When you copy formations, the subordinates in the unit are laid out in each formation according to their order in the Unit Composition window.

To copy a formation:

1. In the object list, select the unit that you want to copy formations to.
2. In the Unit Composition group box, click Edit Formation. The Formation Editor opens (Figure 70-4).
3. In the Formation Editor, click Copy. The Select Formations dialog box opens (Figure 70-7).

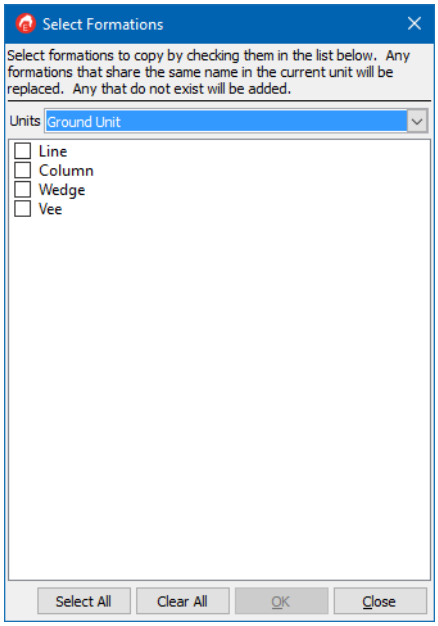


Figure 70-7. Select Formations dialog box

4. In the Aggregates list, select the unit whose formations you want to copy.
5. In the window, select the formations that you want to copy.
6. Click OK. The formations are copied from the selected unit to the currently displayed unit.

70.5.8. Laying Out a Formation Automatically

You can automatically distribute the simulation objects in a unit into one of the built-in formations. You can specify the distance between each entity in the formation.

To lay out a formation automatically:

1. In the Unit Composition group box, click Edit Formation. The Formation Editor opens (Figure 70-4).
2. In the Formation Editor, select the tab for the formation that you want to lay out.
3. Click Auto Layout. The Auto Layout dialog box opens.
4. In the list, select a formation type.
5. Specify the distance, in meters, between each entity in the formation.
6. Click OK. The simulation objects in the unit are reorganized to the new layout.

70.5.9. Changing a Formation's Layout by Hand

You can lay out a formation by hand. The gridline in the Formation Editor indicates the center of the formation (Figure 70-8). This is where the unit icon is placed when the unit is collapsed or aggregated.

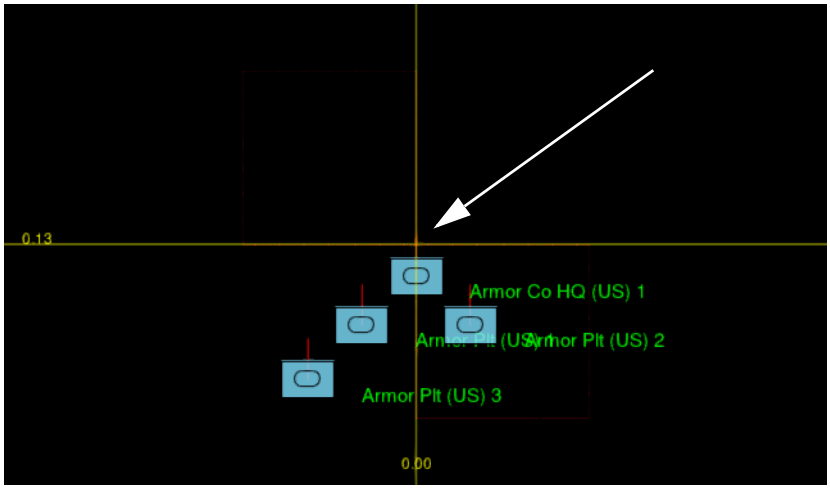


Figure 70-8. Formation Editor crosshair

To change a formation's layout:

1. In the Unit Composition group box, click Edit Formation. The Formation Editor opens (Figure 70-4).
2. In the Formation Editor, select the tab for the formation that you want to lay out.
3. Drag each subordinate to the location where you want it to be.

Snapping Subordinates to the Grid

When you lay out a formation you can specify that simulation objects snap to the grid. That is, they can only be placed at the intersection of vertical and horizontal grid lines. When snap-to-grid is enabled, if you want to place simulation objects more closely together than the grid allows, you can change the scale factor.

➤ To enable snap-to-grid for entity placement, in the Formation Editor, select Snap To Grid.

Changing the Grid Spacing

Grid spacing determines the distance, in meters, between grid lines. The default is 1000 meters.

➤ To change the grid spacing, enter a value in the Grid Spacing box.



71. Editing Aggregate-Level Objects

This chapter explains how to edit simulation objects and combat engineering objects for aggregate-level modeling.

Creating Objects for Aggregate-Level Modeling 71-2
Configuring Simulation Objects for Aggregate-Level Modeling 71-2
Aggregating Units into Higher Echelon Units 71-3
Configuring Ammunition, Weapons, and Equipment 71-4
Defining Ammunition, Weapons, and Equipment Items 71-7
Assemblies 71-10
Creating Assemblies 71-11
Editing Assemblies 71-12
Adding Assemblies to a Unit 71-13
Removing Assemblies 71-13
Creating Roll Up Rules 71-14
Rolling Up Assemblies 71-15
Adding Activities 71-16
Object Parameters for Aggregate-Level Modeling 71-17
General Parameters 71-18
Physical Parameters 71-18
Movement Parameters 71-19
Sensor Parameters 71-21
Attack Parameters 71-22
Defense Parameters 71-23
Combat Parameters 71-24
Electronic Warfare (EW) Parameters 71-24
Personnel and Equipment Parameters 71-25
Supplies Parameters 71-25
Configuring Combat Engineering Objects 71-26
Editing Reader/Writer Key/Value Pairs 71-27

71.1. Creating Objects for Aggregate-Level Modeling

You create new aggregate-level simulation objects and combat engineering objects the same way you create entity-level objects, by creating a new model or creating an object from an existing model.

Simulation objects in an aggregate-level scenario have a more complex set of parameters than simulation objects in an entity-level scenario, so ensuring consistency among similar types of simulation objects could become difficult if after creating new models you had to configure all of the parameters from scratch or if after creating a model from an existing model you had to back out all of the assemblies and parameters that you wanted to change. Therefore, VR-Forces includes templates for common entity types. If you create a new model from a template (by selecting a template and then choosing **Model → New Model From Existing**), you know that you will have a simulation object with similar characteristics to other simulation objects of its type, that is ready to customize with assemblies or direct specification of equipment, weapons, and personnel.

71.2. Configuring Simulation Objects for Aggregate-Level Modeling

Simulation objects for aggregate-level modeling (those used in the aggregate warfare model) are configured in the Simulation Object Editor (*AggregateLevel.sms*). Some aspects of their configuration, such as name, enumeration, and tags are the same as for simulation objects in an entity-level scenario. Other parameters, the focus of this chapter, are used in the aggregate warfare model (Figure 71-1).

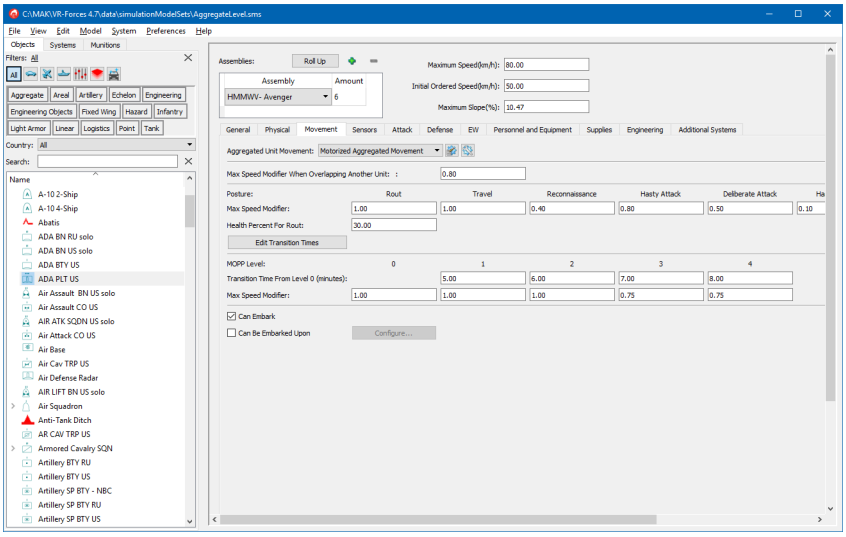


Figure 71-1. Unit in *AggregateLevel.sms*

The parameters for simulation objects in *AggregateLevel.sms* are organized into the following groups:

- General. Basic identification information for the unit.
- Physical. Overall health of the simulation object, its footprint, and sector size for each posture.
- Movement. The movement system to use and how speed is modified by different postures and MOPP levels. (Does not apply to air units.)
- Sensors. Sensor systems and signatures (similar to simulation objects in an entity-level scenario) and how signatures are modified by different postures and MOPP levels.
- Attack. Specifies combat power, ammunition use, weapon systems, and modifiers to combat power. (Does not apply to air units.)
- Defense. Specifies vulnerability to different types of attack and modifiers to vulnerability. (Does not apply to air units.)
- Combat. For air units only. Specifies weapon systems, combat systems, loadout, and other attributes.
- EW. Specifies electronic warfare systems, EW defense, and dependence.
- Personnel and Equipment. Lists personnel and equipment. This information is not used by the back-end in simulations. It is reported as part of the simulation object's state information.
- Supplies. Specifies the supplies a simulation object needs to move, support personnel, and carry out combat engineering operations.
- Engineering. Lets you specify engineering systems. (Does not apply to air units.)
- Additional Systems. Similar to simulation objects in an entity-level scenario, this tab lets you add other systems such as spot report systems.

The individual parameters in each group are described in "Object Parameters for Aggregate-Level Modeling," on page 71-17.

To configure a simulation object, you specify values for as many of the parameters as are appropriate for the object type. However you cannot configure ammunition, weapons, and equipment (aircraft, ships, and vehicles), without first defining them.

71.2.1. Aggregating Units into Higher Echelon Units

You can aggregate simulation objects into higher echelon units. And just as with simulation objects in entity-level scenarios, you specify that a simulation object can be listed in the Aggregate As list by selecting the Allow as Aggregate As Option check box. However, the aggregate-level simulation objects that are configured to use the aggregate warfare model do not have the configuration options for specifying subordinates. Therefore, *AggregateLevel.sms* has a platform, Aggregate Container, for use when aggregating simulation objects. The SMS has several simulation objects configured with that platform for use when aggregating simulation objects. If you want to create additional hierarchical units for aggregation, it is recommended that you use the Aggregate Container platform.

71.3. Configuring Ammunition, Weapons, and Equipment

To engage in combat, simulation objects need appropriate amounts of ammunition, weapons, and equipment. When you create a simulation object in the Simulation Object Editor, you specify the amounts that it should have by selecting them from lists of the available types of these resources. The items in these lists are also configured in the Simulation Object Editor. You can configure the ammunition, weapons, and equipment for simulation objects directly for each entity, or you can simplify the process by creating assemblies.

For example, a U.S. armored cavalry troop (AR CAV TRP US) has 9 M1A2s, 13 M3A3s, and 2 M1064 vehicles. An M1A2 has a 120mm gun, a 12.7mm machine gun, and 2 7.62 mm machine guns. An M3A3 has 2 TOW missiles, a 25mm gun, and a 7.62mm machine gun. An M1064 has a 120mm mortar and 12.7mm machine gun. Therefore, the entity as a whole has 26 TOW missiles, 13 25mm guns, 9 120mm guns, 11 12.7mm machine guns, 31 7.62mm machine guns, and 2 120mm mortars (Figure 71-2).

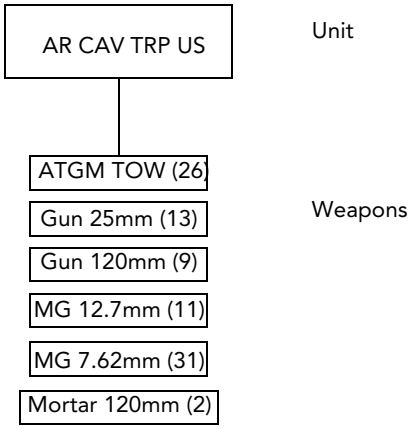


Figure 71-2. Anatomy of a unit without using assemblies

To configure this unit directly, you would go to the Personnel and Equipment tab in the Simulation Object Editor and select all these weapons in the Weapons list and enter the amounts for each. You would also have to add up all the ammunition required by these weapons and enter them on the Attack tab.

This configuration method works fine for simple units and individual entities. However, if you had to calculate weapons and ammunition at this level of granularity for all units, it could become a tedious and error-prone activity. However, there is an alternative to directly configuring all ammunition, weapons, and equipment individually. You can create assemblies. Assemblies are resources such as complete weapons, vehicles, or even small units, such as platoons. They are a convenient way to package ammunition, weapons, and equipment to more easily add them to units.

Each assembly configures the ammunition and weapons it needs for one instance of the resource, such as an M1A2. When you create a unit, you just give it the assemblies it needs. Their capabilities are "rolled up" to define the total amount of resources for the entity.

Figure 71-3 illustrates the anatomy of an AR CAV TRP US entity that uses assemblies to represent M1A2, M3A3, and M1064 vehicles. Each assembly specifies its weapons and ammunition. When you configure the AR CAV TRP US, you just specify that it has 9 M1A2, 13 M3A3, and 2 M1064 assemblies. You roll up the assemblies and the Simulation Object Editor calculates the number of each type of weapon needed (Figure 71-4).

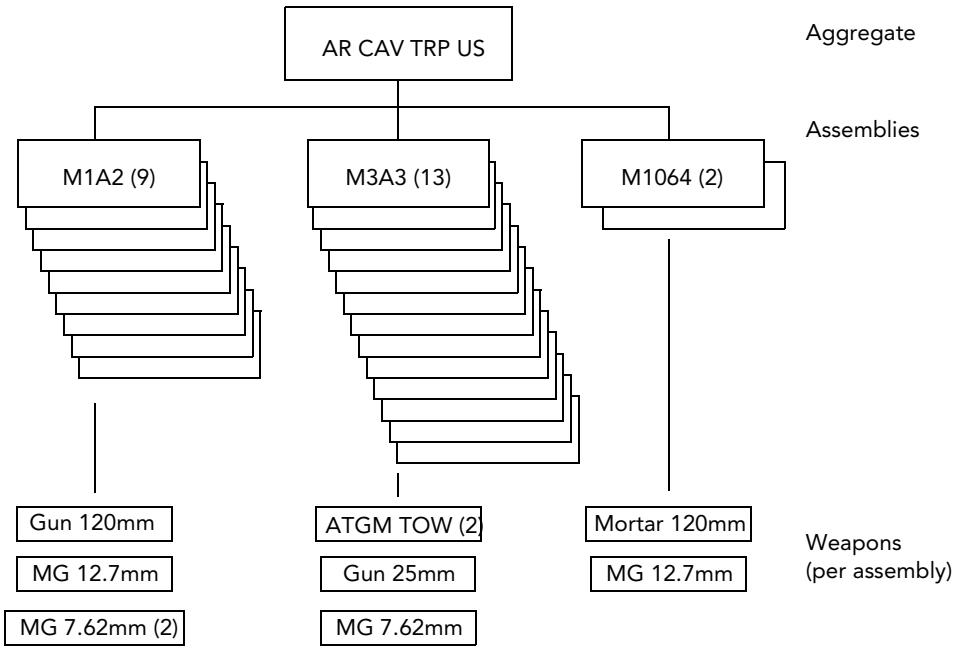


Figure 71-3. Anatomy of a unit using assemblies

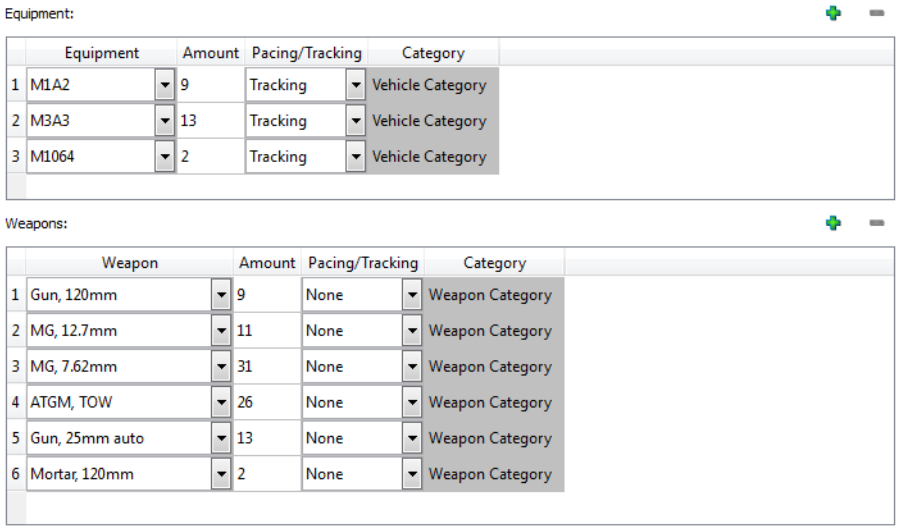


Figure 71-4. AR CAV TRP US weapons rolled up on Personnel and Equipment tab

Using assemblies, the process for configuring units is to:

1. Define the weapons, equipment, and ammunition that units can use.
2. Define assemblies.
3. Create a unit.
4. Specify the unit's assemblies.
5. Roll up the assemblies.
6. Configure all other parameters.

For more information about assemblies, please see "Assemblies," on page 71-10.

71.3.1. Defining Ammunition, Weapons, and Equipment Items

Each ammunition, weapon, and equipment item that you want to be available for use by simulation objects is defined as a key/value pair. The key can be any text string you want to use. There is no relationship to any underlying VR-Forces code. The value is a Display Name – the string used in lists when creating assemblies or adding resources directly to a simulation object.

For each ammunition key/value pair you also must specify a type, where the types are:

- Anti-air
- Anti-personnel
- Anti-ship
- Anti-tank
- Biological
- Chemical
- High explosive
- Nuclear
- For weapon system.

Use the anti-air, anti-personnel, anti-tank, and anti-ship categories for ammunition that will be used by assemblies or added directly to a simulation object. Use For Weapon System for any type of ammunition that will be used by a weapon system that must be added to a simulation object, such as a missile system, a guided bomb, or indirect fire system.

For each weapon and equipment key/value pair, you also must specify a category, where the categories are aircraft, ships, vehicles, and weapons.

The ammunition, weapons, and equipment defined in *AggregateLevel.sms* are representative of the weapons and equipment in common use by the United States and Russia. However, you can add any sort of resource you want. For example, you could create entries for knives, sling shots, and shovels. These could be added to assemblies or directly to a simulation object. The only real requirement for making them useful in a scenario is to assign an appropriate level of combat strength when you add them to a simulation object.

To define an ammunition, equipment, or weapon item:

1. Choose **Edit** → **Ammunition and Equipment**. The Edit Ammunition and Equipment dialog box opens (Figure 71-5).

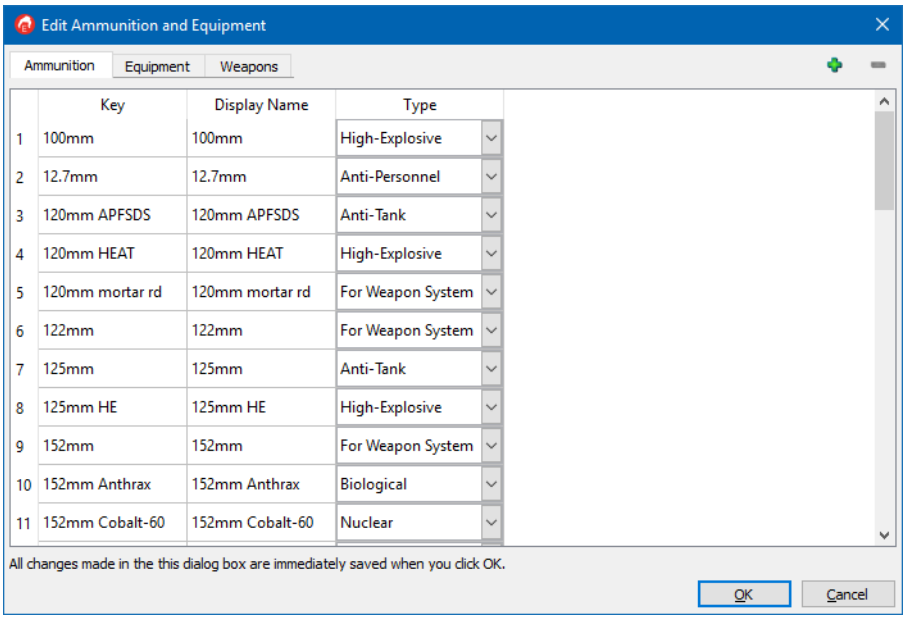


Figure 71-5. Edit Ammunition and Equipment dialog box

2. Select the tab for the type of resource you want to add.
3. Click the Add button (+). A row is added to the list. The Key cell is editable.
4. Type the key value.
5. Click Tab or double-click the Display Name cell to make it editable.
6. Type the text you want to be visible in the lists where this item will be included.

7. For ammunition items, select an option in the Type list.
   For equipment items, select the appropriate category.
   For weapon items, select Weapon Category.
8. Click OK.

71.4. Assemblies

Assemblies are a convenient way to provide a simulation object with a set of resources without having to individually configure each resource.

A unit does not model individual entities. It represents some number of vehicles and personnel. For example, a mechanized infantry company has personnel, armored personnel carriers, and so on, which carry a variety of weapons and ammunition. Rather than explicitly configuring a unit to have, for example, 20 APCs, 20,000 rounds of ammunition, 10 howitzers, and 200 infantry, you can configure the ammunition and weapons for one APC as an assembly. Then you would assign a unit 20 APC assemblies. You would not have to tally all the resources associated with an APC.

This becomes even more beneficial if more than one unit uses an assembly. For example, the Scout Hvy PLT US unit uses M3A3 assemblies. The Tank CO US uses M1A2 assemblies. The AR CAV TRP US uses M1A2s, M3A3s, and M1064 assemblies. While it might be a manageable process to configure M1A2s just for one unit and M3A3s just for one unit, to use them in multiple units and have to calculate the sum of all of their weapons and ammunition each time would become tedious and error-prone. Using assemblies makes it easy to assign multiple units of each type to a unit.

71.4.1. Creating Assemblies

When you create an assembly, you specify the equipment, weapons, ammunition, variables, and systems that get configured for a simulation object when you add the assembly to it.

To create an assembly:

1. Choose **Edit** → **Assemblies**. The Edit Assemblies dialog box opens (Figure 71-6).

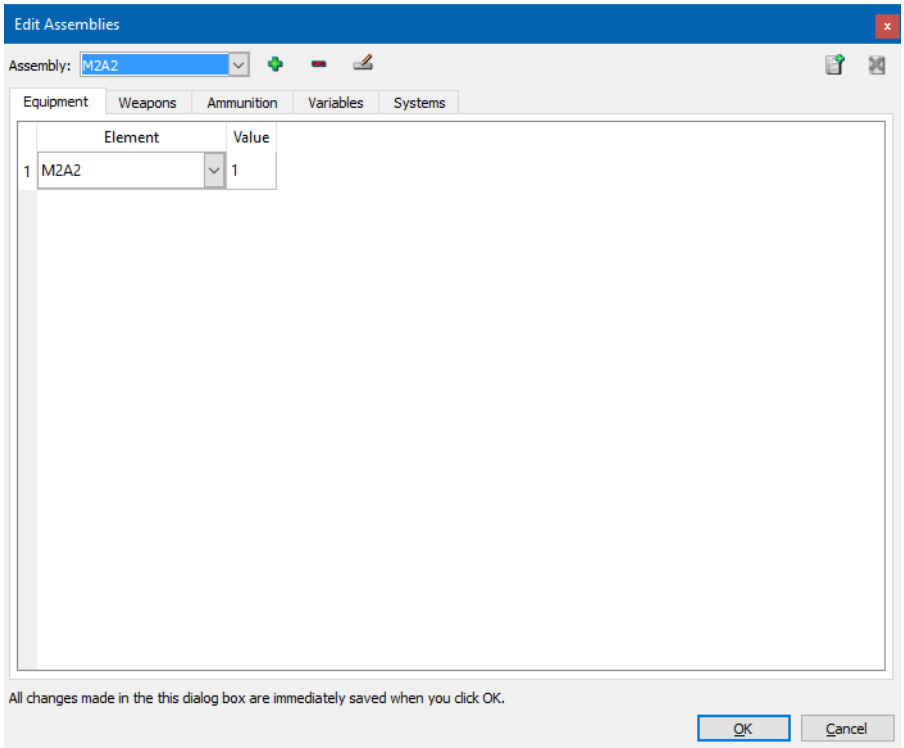


Figure 71-6. Edit Assemblies dialog box

2. Click the Add button (). The New Assembly dialog box opens.
3. Type a name for the assembly.
4. Optionally, select an assembly to base the new one on.
5. Click OK. The new assembly is added to the Assembly list.
6. Edit the assembly.

71.4.2. Editing Assemblies

An assembly specifies equipment, weapons, ammunition, variables, and systems. Variables represent entity resources that do not fit into weapons, ammunition and equipment. These include tangible resources, such as fuel, water, and personnel, and intangible variables, such as health and combat power. If you do not specify particular variables for an assembly, you can specify them directly for a simulation object.

When you create an assembly, you can edit the properties of its systems. You can also edit system properties when you edit a simulation object. Many system properties do not change when you roll up a unit, but some do. For example, indirect fire weapons have a maximum sheaf parameter. The default maximum sheaf value for the 155mm howitzer is 80 meters. If an artillery unit has six howitzers and you roll it up, the maximum sheaf value for the howitzers in that entity becomes 480.

! If you specify variables for an assembly, when you roll up assemblies, the variables in the assembly overwrite the values that have been set directly. Therefore, it is recommended that you use them consistently in all your assemblies and that you not edit these values directly in a unit if you know that you will use assemblies that might overwrite them.

To edit an assembly:

1. Choose **Edit** → **Assemblies**. The Edit Assemblies dialog box opens (Figure 71-6).
2. In the Assembly list, select the assembly that you want to edit.
3. For the Equipment, Weapons, Ammunition, and Variables tabs, do the following:
   a. Click the Add Attribute button (). A new row is added to the list of resources on the tab. In the Element column, it lists the first resource in the list of available resources.
   b. In the Element list, select the resource you want to add to this assembly.
   c. Double-click the Value cell to make it editable.
   d. Type the amount of this resource that you want to be in this assembly.
4. For the Systems tab, add a system as described in step 3. There is no Value column to complete, but you can edit the system's properties. To edit properties:
   a. Click the Edit Properties button (). A Properties dialog box opens.
   b. Set the property values as desired.
5. Click OK.

71.4.3. Adding Assemblies to a Unit

You can configure individual vehicles and weapons for a unit, or you can add assemblies, which add all the relevant parameters and combine them for the number of instances of each assembly that you add.

To add an assembly to a unit:

1. Select the entity in the object list.
2. Click the Add button (+) above the Assemblies list (Figure 71-7). A new row is added to the list.

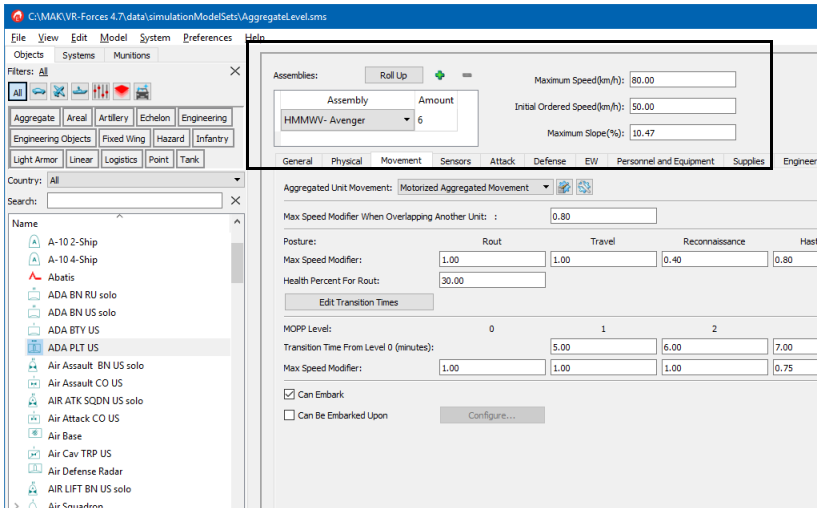


Figure 71-7. Assemblies table

3. In the Assembly column, select an assembly from the list.
4. Double-click the Amount value to make it editable.
5. Enter how many of this assembly that you want to add.

71.4.4. Removing Assemblies

To remove an assembly:

1. In the Assemblies list, select the index number or any cell in the assembly you want to remove.
2. Click the Delete button (−).

71.5. Creating Roll Up Rules

VR-Forces rolls up assemblies based on a set of rules. Roll up rules are not specific to simulation objects; they are global for an SMS. A roll up rule specifies the assemblies to roll up and the formula to use to roll them up, as follows:

- Sum. Add up the values for all the assemblies.
- Average. Use the average value for identical variables in the assemblies.
- Maximum. Use the highest value specified for identical variables in the assemblies.
- Minimum. Use the lowest value specified for identical variables in the assemblies.

Ammunition, weapons, equipment, and systems are all rolled up by summation. There are no user-editable roll up rules for these resources.

To create roll up rules:

1. Choose **Edit** → **Roll Up Rules**. The Edit Roll Up Rules dialog box opens (Figure 71-8).

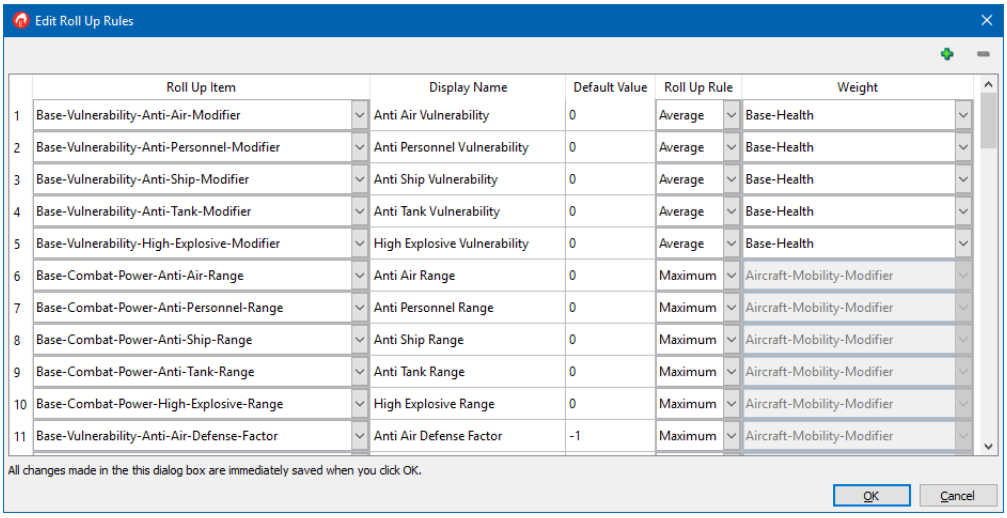


Figure 71-8. Edit Roll Up Rules dialog box

2. Click the Add button (+). A new row is added to the list.
3. Select the type of roll up item from the list.
4. Specify a Display Name.
5. Specify a default value.
6. Select an option in the Roll Up Rule list.

7. If you are adding an Average roll up type rule, select an option in the Weight list. This creates a weighted average based on the item you want to give greater weight, such as health or vulnerability.
8. Add additional items as desired.
9. Click OK.

71.6. Rolling Up Assemblies

Once you assign a set of assemblies to a unit, you roll them up so that the full set of resources is applied to the unit. Assemblies are rolled up according to roll up rules.

When you roll up assemblies, the resources and modifiers in the assemblies overwrite the current values for the unit. If you have edited any of these resources or modifiers manually, your changes are lost. Any parameters that are not part of an assembly or roll up rules do not change.

To roll up assemblies for a unit:

1. Select the unit.
2. Choose **Edit** → **Roll Up All**, or in the Assemblies area of the Simulation Object Editor window, click Roll Up. You are prompted to confirm the command.
3. Click Yes.

71.7. Adding Activities

You can assign simulation objects an activity. The Simulation Object Editor lets you add activities. When you add an activity, it is included in the list of activities for the Activity set data request. However, unless you add code or a Lua script that takes advantage of the new activity setting, assigning an activity to a simulation object will have no effect.

To add an activity to the Activities list:

1. Choose **Edit** → **Activities**. The Edit Activities dialog box opens (Figure 71-9).

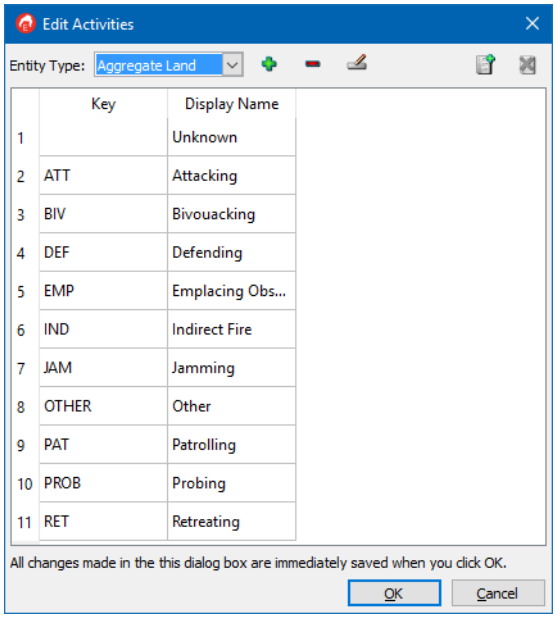


Figure 71-9. Edit Activities dialog box

2. Select an object type in the Entity Type list, or click the Add button (+) to add a new type.
3. Click the Add Attribute button (). A new entry is added to the list of activities. The Key cell is active for editing.
4. Type a key value for the activity. This is the value that the code or Lua script uses to identify the activity.
5. Press **Tab** or double-click in the Display Name cell to make it editable.
6. Type the text to display in the Activities dialog box.
7. Click OK.

71.8. Object Parameters for Aggregate-Level Modeling

The aggregate warfare model is data driven. It relies on the comparative strengths of opponents given their resources and movement posture. The data used is from an extensive set of parameters that are configured for each simulation object in the Simulation Object Editor.

Most parameters are in one of the following categories:

- Real-world absolute values, such as liters of diesel fuel, number of personnel, and amount of equipment and ammunition.
- Modifiers. Modifiers change the base values for the various indicated conditions, such as posture, MOPP level, or footprint. All modifiers are multipliers. For example, speed modifiers are applied to the maximum speed. If the maximum speed for a simulation object is 20 KM/hr and the modifier for overlapping another unit's footprint is .9, the entity will not be able to go faster than 18 KM/hr while it overlaps another unit.
- Systems. Movement system, sensor systems, other systems.
- Abstract values. Health and strength are abstract values. They represent the relative ability of simulation objects to damage other simulation objects and absorb damage.

For the simulation objects in *AggregateLevel.sms* provided with VR-Forces, real-world values are based on publicly available information about the types of simulation objects. The various modifiers applied to simulation object behavior are derived from customer requirements for the aggregate warfare model. Modifier values have been set from experimentation to provide what we believe to be realistic results. Values for health and strength have also been developed on a relative scale to obtain reasonable results.

If you add new simulation objects to *AggregateLevel.sms*, it is important that you try to set health and strength to values that fit with the existing simulation objects. If you create an entirely new SMS for aggregate-level scenarios, you are responsible for developing your own consistent way to represent the relative strengths of the simulation objects.

Parameters are organized in a set of tabbed pages. The following sections describe the parameters on each tab. The Additional Systems tab does not include aggregate warfare parameters. It lets you specify systems that did not logically belong on the other tabs.

71.8.1. General Parameters

The General tab lets you specify identifying information for a unit, such as its name, enumeration, tags, and model.

71.8.2. Physical Parameters

Physical parameters describe a simulation object's footprint and how it faces the external world for each posture, which is a simulation object's overall mode of interaction with its environment. Air units and ground units have different parameters.

Table 71-1: Physical parameters for ground units

Parameter	Description
Health	A quantification of the overall health of the entity.
Physical Footprint	The area notionally occupied by the simulation object. The footprint can vary for different postures based on the Footprint Modifier.
Height	Height of the physical footprint.
Footprint Modifier	The amount by which the footprint changes for the indicated posture.
Front Sector Size	The number of degrees, centered on the heading, within which forward-facing sensors and interactions take place.
Rear Sector Size	The number of degrees, centered on the heading -180^{o}, within which rear-facing sensors and interactions take place.
Flank Sector Size	The number of degrees, centered on the heading +/- 90^{o}, within which side-facing sensors and interactions take place.

71.8.3. Movement Parameters

A unit has a movement system. You can specify the movement system on this tab and edit its parameters. You can also specify posture transition times.



Aircraft units do not have a Movement tab.

Movement can be affected by:

- Environmental factors. These properties are specified per movement system.
- Posture. A simulation object's posture can affect its speed.
- MOPP level transition time. The amount of time it takes for a unit to change from MOPP level 0 to one of the other MOPP levels.

The Movement tab also has settings for embarkation. These settings are the same as for simulation objects in an entity-level scenario. For details, please see Chapter 72, *Configuring Embarkation*.

Table 71-2 describes the movement parameters.

Table 71-2: Movement parameters

Parameter	Description
Max Speed Modifier When Overlapping Another Unit	The amount to modify a simulation object's maximum speed when its footprint overlaps another simulation object's footprint.
Max Speed Modifier (Posture)	The amount to modify the maximum speed based on the posture.
Health Percent For Rout	The amount to modify health when in the rout posture.
Transition Time from Level 0	The amount of time it takes to transition from MOPP Level 0 to a higher MOPP level.
Max Speed Modifier (MOPP Level)	The amount to modify the maximum speed based on the MOPP Level.

Editing Movement System Properties

Table 71-3 describes movement system properties. Systems do not all have the same properties.

Table 71-3: Movement system properties

Property	Description
Rain Modifier by Intensity	The amount speed is affected by the intensity of rain. The lower the modifier value, the more slowly the simulation object moves as rain intensity increases.
Snow Modifier by Intensity	The amount speed is affected by the intensity of snow. The lower the modifier value, the more slowly the simulation object moves as snow intensity increases.
Visibility Degrades Max Speed When Below	If visibility drops below this value, maximum speed is reduced.
Maximum Amount Visibility Can Degrade Speed	The maximum amount that speed decreases as visibility decreases. Higher values decrease speed more.
Wind Degrades Max Speed When Above	If wind speed exceeds this value, the maximum speed of the entity is reduced. (Aircraft only.)
Wind Degrades Until	Degrade speed due to wind until the wind speed reaches this value. Then degrade speed at the maximum rate. (Aircraft only.)
Maximum Amount Wind Can Degrade Speed	The maximum amount that wind can degrade speed. (Aircraft only.)
Retrograde Movement Penalty	Speed is reduced by this amount when moving in retrograde. (Ground movement only.)

To edit movement system properties:

1. Click the Edit Properties button (). The Edit Properties dialog box opens.
2. Edit the values for the various properties.
3. Click OK.

Editing Posture Transition Times

VR-Forces models the amount of time it takes for a unit to transition from one posture to another. You can specify the transition times for each movement system.

To edit posture transition times:

1. Select the Movement tab.
2. Click Edit Transition Times. The Edit Transition Times dialog box opens (Figure 71-10).

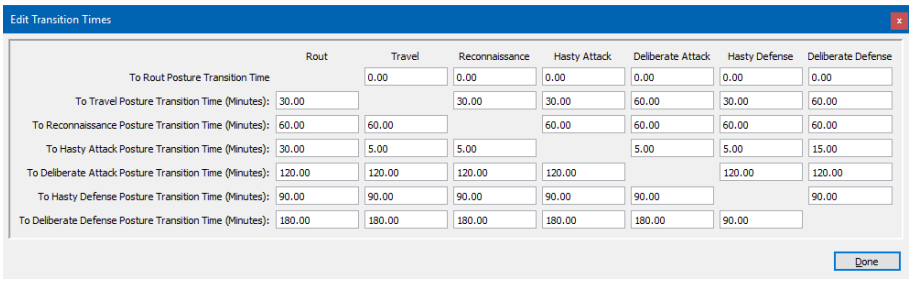


Figure 71-10. Edit Transition Times dialog box

3. Change the times, in minutes, as desired. Changes take effect immediately.
4. Click Done.

71.8.4. Sensor Parameters

The Sensors tab lets you specify the sensors that a simulation object has. You can add and remove sensor systems and set their properties using the standard Simulation Object Editor procedures. (For details, please see "Editing a Simulation Object's Systems," on page 69-15.)

You can specify the simulation object's sensor signatures by choosing from a preconfigured set of signatures or by defining a custom set. The Sensor Signatures parameters specify the distance at which other simulation objects can sense the simulation object you are editing using the different types of sensors. The preconfigured signatures make it easy to consistently apply the same set of signatures to similar object types.

As with movement, a unit's posture and MOPP level affect its ability to use its sensors. You can specify signature modifiers for each posture and MOPP Level. (Posture and MOPP level modifiers do not apply to air units.)

71.8.5. Attack Parameters

Simulation objects have combat power in five domains: Anti-Air, Anti-Tank, High Explosive, Anti-Personnel, and Anti-Ship. The Attack tab specifies the parameters that determine a unit's ability to attack an opponent in these domains. It also lets you add weapon systems to the simulation object.



Aircraft units do not have an Attack tab.

Table 71-4: Attack parameters

Parameter	Description
Range	The distance at which a simulation object can attack in each domain.
Hit Factor	When compared to the defense factor for a target simulation object, determines the probability of the attack hitting the target.
Strength	The simulation object's strength in each type of combat. This is an abstract value.
Ammunition Usage	The number of rounds per minute used for each type of combat. This value is apportioned among ammunition of the same type. For example if the ammunition usage for Anti-Personnel combat is 1000 rounds per minute and a simulation object has 12.7mm ammunition and 7.62mm ammunition, both of which are type Anti-Personnel, then the simulation object uses 500 rounds per minute of each type of ammunition.
Ammunition Auto-Resupply Rate	The amount of ammunition that is automatically resupplied each day if auto-resupply is enabled for this simulation object.
Ammunition	The types of ammunition this simulation object has, the amount of each type, and whether or not pacing or tracking is enabled.
Distribution	The proportion of casualty types from an attack.
Sector Combat Power Modifiers	The amount the base combat power is modified depending on which sector is being attacked.
Posture Combat Power Modifiers	The effect on combat power of each posture.
Combat Range Modifiers	The effect on the range at which a simulation object can attack based on its posture.
CID Level Combat Power Modifiers	The effect on combat power of the enemy detection and classification level.
MOPP Level Combat Power Modifiers	The effect on combat power of the MOPP Level.

Table 71-4: Attack parameters

Parameter	Description
If Morale is Less Than	A percentage of morale below which combat power is affected.
Adjust Combat Power By	The combat power modifier to apply if morale is less than the specified amount.
Environmental Conditions Combat Power Modifier	The combat power modifier to apply if precipitation intensity is at its maximum or if there is no illumination.
Weapon System Auto-Resupply Rate	The number of rounds per day supplied to each weapon system if auto-resupply is enabled.

71.8.6. Defense Parameters

The Defense tab lets you specify modifiers to a simulation object's vulnerability. Vulnerability is specified within each weapon domain.



Aircraft units do not have a Defense tab.

Table 71-5: Defense parameters

Parameter	Description
Vulnerability	The amount that a simulation object is vulnerable to the different types of attack.
Defense Factor for Guided Munitions	The ability to defend against guided munitions.
NBC Vulnerability	Vulnerability to nuclear, biological, or chemical contamination.
Posture Vulnerability Modifier	The effect of different postures on the simulation object's ability to defend itself.
Sector Vulnerability Modifier	The ability of the simulation object to defend itself to the front, rear, and flanks.
MOPP Level Vulnerability Modifier	The ability of the simulation object to defend against NBC contamination at different MOPP Levels.

71.8.7. Combat Parameters

Air units have a Combat tab. This tab lets you configure weapon systems, combat systems, loadout, and health parameters.

Table 71-6: Combat parameters

Parameter	Description
Weapon Systems	Add, remove, and edit properties for the weapon systems this unit has.
Combat Systems	Add, remove, and edit properties for other combat-related systems.
Loadout	The type of loadout this unit uses. (Loadouts are specific to each unit. There is no library of preconfigured loadouts available to select from.)
Loadout Items	The specific munitions in the loadout.
Distribution	The percent chance that a casualty caused by this unit will be of a specific casualty type.
Defense Factor for Guided Munitions	The defense factor to apply when this unit is struck by a guided munition.
Weapon System Attack Interval	The amount of time between each weapon system attack by this unit.
Health Percent for Firepower Kill	The level of health below which this unit will not be able to fire.
Health Percent for Mobility Kill	The level of health below which this unit will not be able to move.
Health Percent for Sensor Kill	The level of health below which this unit will not be able to sensor other objects.

71.8.8. Electronic Warfare (EW) Parameters

The EW tab lets you specify a simulation object's ability to defend against electronic warfare and how much it depends on communications. You can add EW systems.

71.8.9. Personnel and Equipment Parameters

The Personnel and Equipment tab lets you specify the order of battle for the simulation object. These values are not used by the back-end in the simulation. They are for reporting purposes. The configured personnel and equipment amounts are multiplied by the simulation object's health to arrive at the current amount.

Table 71-7: Personnel and Equipment parameters

Parameter	Description
Personnel Officers	Number of officers.
Personnel WOs	Number of warrant officers.
Personnel NCOs	Number of non-commissioned officers.
Personnel Enlisted	Number of enlisted persons.
Equipment	List of equipment types, the number of each, and whether it is flagged for pacing, tracking, or none.
Weapons	List of weapon types, the number of each, and whether it is flagged for pacing, tracking, or none. (Ground units only.)

71.8.10. Supplies Parameters

For ground units, the Supplies tab lets you specify the amount, use per day, use per kilometer, pacing/tracking, and auto-resupply rate for the following supplies:

- Food.
- Water.
- Oil.
- Lubricant.
- Motor gas.
- Aviation fuel.
- Diesel fuel.
- Other Supplies. You can add any other type of supply that you want. The list of items in the Supply drop-down list is drawn from *./data/simulationModelSets/AggregateLevel/gui/rwKeywordValuePairs.xml*. (For information about this file, please see "Editing Reader/Writer Key/Value Pairs," on page 71-27.) You can type in any other supply type that you want to.

  The supplies added here are only used by engineering objects. To use a supply, you configure it here and in the engineering object. For example, a combat engineering company has other supplies called Construction-Material and Explosives.

This tab also lets you specify the conditions under which a simulation object can receive supplies.

For air units, you can specify the amount of aviation fuel, the aerial refueling system, and other supplies.

71.9. Configuring Combat Engineering Objects

Combat engineering objects (CEOs) can affect mobility of simulation objects and can cause attrition. Therefore, their simulation models include the information needed for these effects. Since CEOs can be constructed by units, their models include parameters that specify the types of materials needed to create them and sizing information.

The actual process of configuring CEOs is similar to that of other objects. Table 71-8 describes the parameters specific to CEOs. Some parameters do not apply to all CEOs.

Table 71-8: Combat engineering object parameters

Parameter	Description
Short Name	The name displayed on the map and sent as marking text over the network.
Height	The height of the object.
Maximum length	Maximum length of single constructed objects. If a longer object is specified, it is created as several objects.
Width	The width of a line object.
Minimum Completion Percentage for Effect	The minimum completion percentage before the object affects simulation objects.
Minimum Completion Percentage to Detect	The minimum completion percentage before simulation objects can detect the object.
Attrition	The rate of attrition to an affected simulation object in each domain.
Mobility Modifiers	The amount that mobility of an affected simulation object is modified in each type of movement.
Vulnerability Modifiers	The amount that an affected simulation object's vulnerability is modified in each domain.
Signature Modifiers	For object types that can be affected by this object, the amount that each type of sensor signature is affected.
Resource Required	The type of resource needed to construct this object.
Resource Amount	The amount of the required resource needed to build this object.

Table 71-8: Combat engineering object parameters

Parameter	Description
Is Concealed	If selected, the object is detected based on a random probability in the detecting sensor. If cleared, the object is always detected by the appropriate sensor.
Maximum Relative Size	For areal objects, the maximum size of a simulation object, relative to the object, for it to be affected by the object. For example, if the maximum relative size for a fortified area is .75, a simulation object must be 75% of the size of the area or smaller for it to be protected.

71.10. Editing Reader/Writer Key/Value Pairs

Some display text is not automatically included in the *untranslated.ts* file provided with VR-Forces. The Simulation Object Editor lets you map some internal keywords to display text so that you can translate it. These internal keywords are called reader/writer keys and they are managed using key/value pairs. These mappings are stored in *./data/simulationModelSets/AggregateLevel/gui/rwKeywordValuePairs.xml*. If you edit this file, do not change the `paramName`; change the string mapped to it.

Besides allowing for translation, you can add new entries to this file. Table 71-9 describes the effect of adding new keyword:value pairs.

Table 71-9: Effect of editing rwKeywordValuePairs.xml parameters

Parameter	Result
Type	Added to the Type list for the Edit Ammunition and Equipment dialog box, Ammunition tab, in the Simulation Object Editor.
Category	Added to the Category list for the Edit Ammunition and Equipment dialog box, Weapon tab and Equipment tab, in the Simulation Object Editor.
Posture	Added to the list of Postures in the Posture set data request. However, selecting such a posture would not have any effect unless complementary code is written to update the simulation engine.
Resource-Item	Added to the Supply list for configuring Other Supplies on the Supplies tab for simulation objects in the Simulation Object Editor.



72. Configuring Embarkation

This chapter explains how to configure entities for embarkation. It includes a tutorial for configuring embarkation slots.

Enabling Embarkation 72-2
Configuring Slots 72-5
Adding Slots 72-5
Configuring Slot Parameters 72-7
Configuring Slot Exclusions 72-9
Specifying Ingress and Egress Techniques 72-10
Adding Ingress and Egress Points 72-11
Embarkation Slot Tutorial 72-14
Configure the SuperTanker for Embarkation 72-14
Configure Slots for the Supertanker 72-16
Add Ingress Points 72-21
Add Egress Points 72-22
Save the New Configuration 72-23
Test the Embarkation Configuration 72-23

72.1. Enabling Embarkation

If an entity allows other entities to embark on it, you can configure:

- Allowed entity types. The types of entities that can embark.
- Slots. The location on an entity where embarked entities move to. The number of slots on an entity controls how many entities can embark on it using the Embark task. For entities that are visible when they are embarked, such as humans on a truck or aircraft on an aircraft carrier, it is important to specify sensible looking slots.

  Slots can have a type and a name. Use of slot types and names allows you to tell a simulation object to embark in a specific slot.
- Ingress points. The locations on an entity where entities embark before moving to a slot.
- Egress points. The locations on an entity from which entities disembark.



Units can support embarkation, but the usual use of embarkation is for entities to embark on other entities. In entity-level scenarios, if a unit embarks, the individual members of the unit carry out the embarkation task.

To enable embarkation for an entity:

1. In the Simulation Object Editor object list, select the entity that you want to configure embarkation for.
2. Select the Movement tab (Figure 72-1).

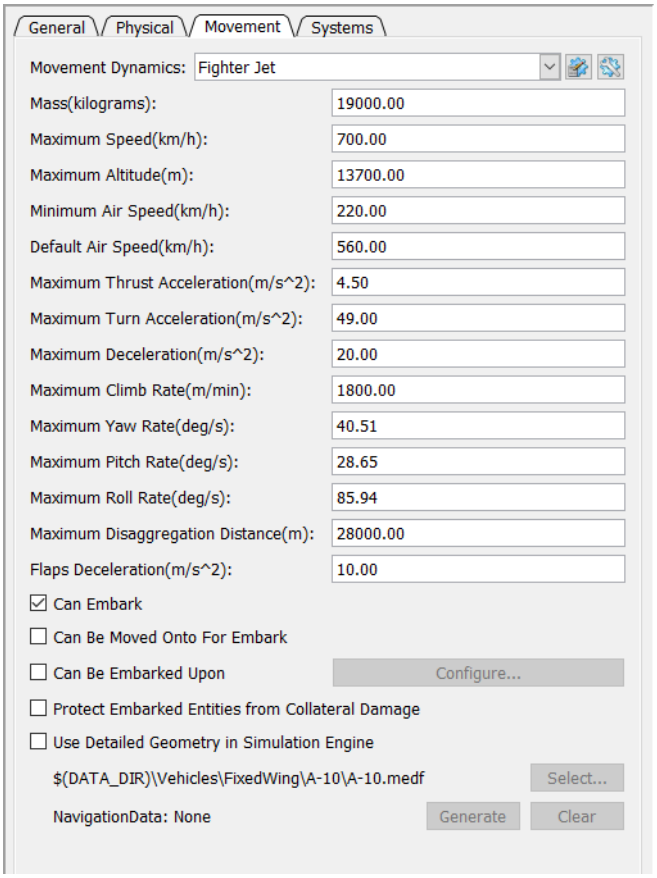


Figure 72-1. Movement tab

3. If Can Be Embarked Upon is not selected, select it.
4. Optionally, select Can Be Moved Onto For Embark. This option lets ground-clamped entities identify entities that they can embark on. They can then walk or drive onto the entity and automatically become embarked.
5. Optionally, select Protect Embarked Entities from Collateral Damage. If selected, embarked entities do not get damaged if the parent entity sustains collateral damage.
6. Click Configure. The Configure Embarkation dialog box opens (Figure 72-2).

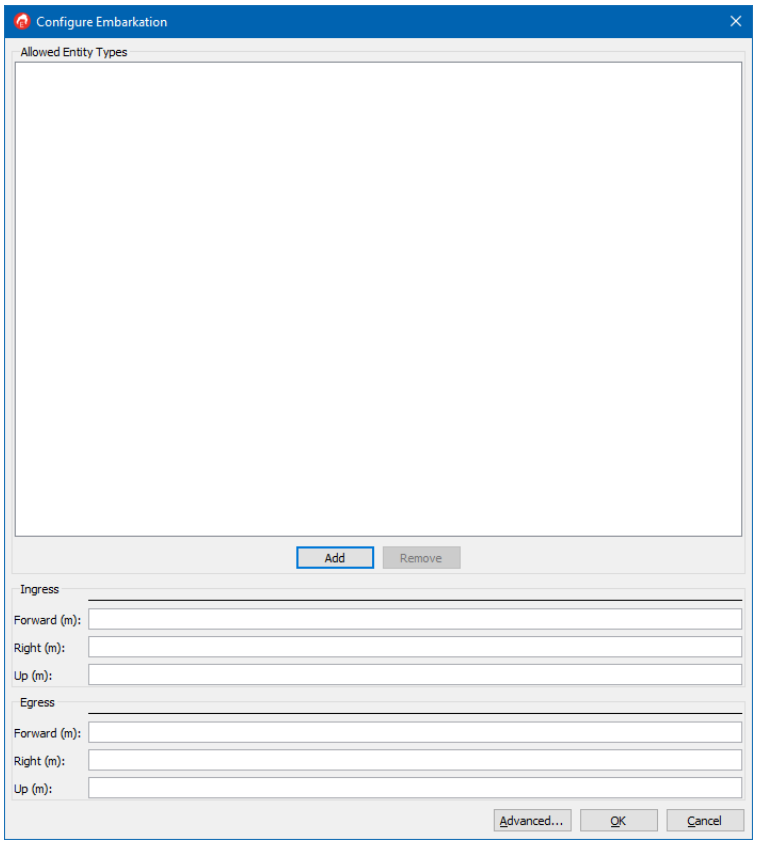


Figure 72-2. Configure Embarkation dialog box

7. Click Add. The Add Type dialog box opens.
8. Select a type of entity that can embark.
9. Optionally, add additional entity types.
10. For each entity type listed, specify ingress and egress points. Values are in meters relative to the center of the parent entity. For more information, please see "Adding Ingress and Egress Points," on page 72-11.
11. Optionally, configure slots. For more information, please see "Configuring Slots," on page 72-5.
12. Click OK.

72.2. Configuring Slots

In the simple embarkation configuration process, VR-Forces adds one slot, located at the center of the parent entity. You can edit that slot. You can also add additional slots if you want embarked entities to be in specific locations on the parent.

72.2.1. Adding Slots

To add a slot:

1. In the Simulation Object Editor object list, select the simulation object that you want to configure embarkation for.
2. Select the Movement tab (Figure 72-1).
3. Select the Can Be Embarked Upon check box.
4. Click Configure. The Configure Embarkation dialog box opens (Figure 72-2).
5. If you have not already specified the simulation object types that can embark, specify a simulation object type (as described in "Enabling Embarkation," on page 72-2.
6. Click Advanced. The dialog box redisplays. It shows a 2D view of the bounding volume of the simulation object for which you are configuring embarkation (Figure 72-3).

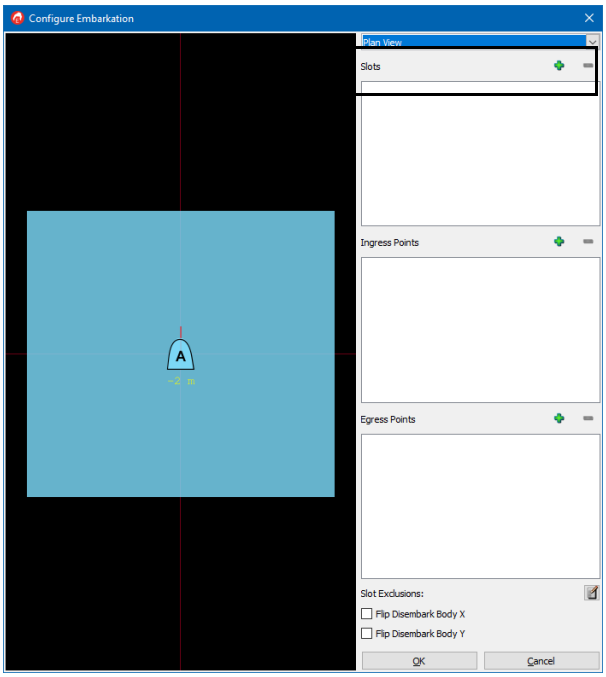


Figure 72-3. Configure Embarkation Advanced

7. Optionally, change the view mode to Stealth to see a 3D model of the entity.
8. Optionally, use the mouse or keyboard to zoom in or out or otherwise change your view of the entity.
9. Click the Add button above the Slots window (Figure 72-3). A slot is added to the list and the dialog box redisplays to show the slot's parameters (Figure 72-4).

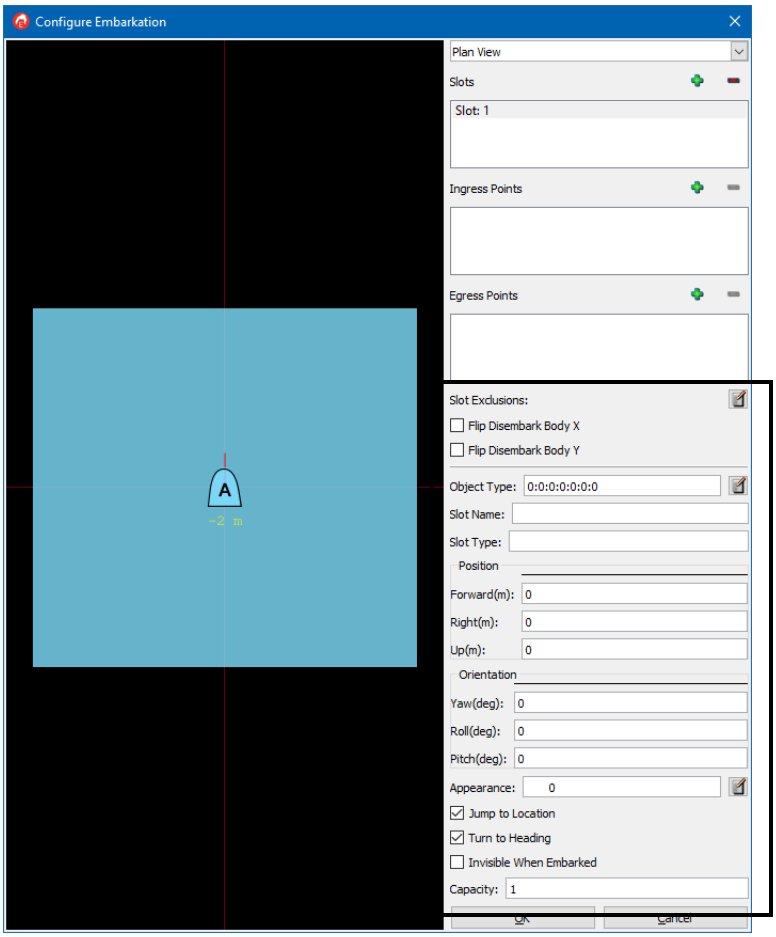


Figure 72-4. Slot parameters

10. Optionally, configure Slot Exclusions as described in "Configuring Slot Exclusions," on page 72-9.
11. Optionally, configure other slot parameters as described in "Configuring Slot Parameters," on page 72-7.

72.2.2. Configuring Slot Parameters

You can specify the following parameters for a slot:

- The object type that can use it. You can configure different object types in different slots on a simulation object.

  For each field in the object type enumeration, you can specify more than one value. For example, the category value for a helicopter can be 20-25 for military helicopters, 50 for unmanned, and 90-92 for civilian. If you just want to allow a specific category for a given slot, specify that category. If you want to specify a range of values, separate them with a hyphen, for example, 21-24. To specify several specific categories, separate them with commas, for example, 21, 25, 50. You can also combine these options. To specify all helicopters, you could set the object type to 1:2:-1:20-25,50,90-92:-1:-1:-1.
- A name.
- A slot type.
- The location on the parent entity relative to the center of the entity's bounding volume.
- The orientation of the slot relative to the center of the entity.
- The appearance of the simulation object type when it is in the slot. For example, a human embarked on a truck might have its posture set to Sitting.
- For slots that accept humans, an animation to use while embarked.
- Whether or not the embarking entity should just jump to the slot, rather than using normal entity movement. For example, an aircraft will taxi to a slot on a carrier deck.
- Whether or not the embarking entity should turn to the heading of the parent.
- Whether or not the embarking entity should be invisible while it is embarked.
- How many objects the slot can hold.

To specify slot parameters:

1. To change the side of a parent simulation object that the embarked simulation objects exit on, select the Flip Disembark Body X check box, the Flip Disembark Body Y check box, or both.
2. If you have added a new slot, in the Object Type field, specify the object enumeration for the type of simulation object that can use this slot. If this is a previously created slot, you can edit the object type. You can configure different object types in different slots on a simulation object.
3. If you want to give the slot a name, type it in the Slot Name text box.
4. If you want to categorize this slot as a specific named type (a text string, not an object type enumeration), type it in the Slot Type text box.
5. In the Position group box, specify the location of the slot relative to the center of the entity's bounding volume.

6. In the Orientation group box, specify the orientation of the slot relative to the center of the entity.
7. Optionally, specify the appearance of the simulation object type when it is in the slot. If you want to specify an animation for a human, specify a posture that supports animations, such as Upright, Standing Still.
8. Optionally, specify an animation that the entity will use when it embarks.



Animations only apply to humans. The available animations depend on the entity's posture, as specified by the Appearance parameter.

9. Optionally, specify an ingress technique. For details about specifying an ingress technique, please see "Specifying Ingress and Egress Techniques," on page 72-10.
10. Optionally specify an egress technique. For details about specifying an egress technique, please see "Specifying Ingress and Egress Techniques," on page 72-10.
11. Optionally, for lifeforms only, if the Appearance specifies a posture that has animations, click the Edit button and select an animation.
12. If you want the embarked simulation object to immediately jump to the slot from the ingress point, select Jump To Location. If this check box is clear, the simulation object will move to the slot.
13. If you want the embarked simulation object to have the same heading as the parent entity, select Turn To Heading.
14. If you want the embarked simulation object to be invisible in 3D observer modes, select Invisible When Embarked.
15. In the Capacity text box, specify the number of entities that can use this slot. -1 means an unlimited number.

72.2.3. Configuring Slot Exclusions

A slot exclusion specifies that if a given slot is occupied, other slots cannot be occupied. For example, an entity might allow both tanks and dismounted infantry entities to embark. If no tanks are embarked, the entity can hold 20 DIs. But if a tank is embarked, it uses up 10 slots and now only 10 DIs can embark.

To configure slot exclusions:

1. On the Slot Exclusions line, click Edit. The Edit Slot Exclusions dialog box opens (Figure 72-5).

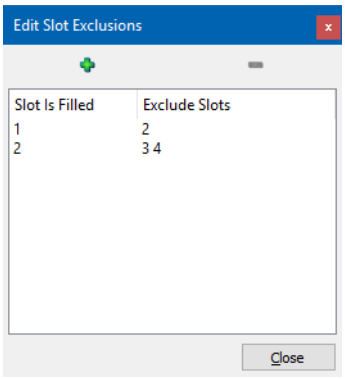


Figure 72-5. Edit Slot Exclusions dialog box

2. Click the Add button (). A blank line is added to the list.
3. Double-click the cell in the Slot Is Filled column. The cell becomes editable.
4. Type the number of the slot that you want to configure.
5. Double-click the cell in the Exclude Slots column.
6. Type the numbers of the slots that cannot accept entities if the slot you are configuring is filled. You can enter more than one slot to be excluded. Separate the slot numbers with a space.
7. Repeat this procedure to add additional exclusions. You can also edit existing exclusions and delete them.

72.2.4. Specifying Ingress and Egress Techniques

When you configure a slot you can specify an ingress and egress technique. An ingress or egress technique associates a string with a script that gets run when the entity embarks. The technique string gets associated with a script ID in the OPE file for the entity type.

To specify a technique in an OPE file, add the `technique-map` block to the embarkation controller. The *Human.ope* file has the following technique map:

```
(embarkation-controller
  ...
  (invisible-at-origin True)
  (technique-map
    (GetInCar "get_in_car")
    (GetOutOfCar "get_out_of_car")
  )
)
```

In this case putting the string GetInCar in the Ingress parameter tells VR-Forces that when the entity embarks, it should run the script that has the script ID get_in_car.

VR-Forces includes two scripts, Get In Car and Get Out Of Car (in the Civilian Vehicle folder) that show how to write a script that implements an ingress or egress technique. As shown in the comments in these scripts, there is a set of standard task variables. Do not define the variables. The script gets their values from the embarkation code.

72.3. Adding Ingress and Egress Points

An ingress point specifies where an entity embarks onto another entity. An ingress point can specify a path that the entity follows to get to the ingress point and how it moves once it is on the entity. For example, a jet embarking on an aircraft carrier can taxi to its slot.



If you do not specify an ingress point, entities will not move towards the parent before embarking. They will teleport to the slot.

An egress point specifies where an entity disembarks from the parent entity.

You can specify that an ingress point should only support some of the slots on the entity. You might want some types of entities embarking at one ingress point and moving to certain slots, while other entities embark at a different location and use other slots. For example, if you are simulating a ferry, cars and trucks would enter at the front or rear and move below deck. Foot passengers would enter at the sides and move to the passenger decks.

When you add an allowed entity type, the Simulation Object Editor automatically creates a slot for that entity type. The ingress and egress points that you configure for that entity type on the basic version of the Configure Embarkation dialog box are automatically associated with the same slot.

In Advanced mode, you can add additional ingress and egress points and associate them with whichever slot you want. You do not have to add them in pairs.



This procedure describes configuring an ingress point. To configure an egress point, substitute egress for ingress in each step.

To add an ingress point:

1. In the Simulation Object Editor object list, select the entity that you want to configure embarkation for.
2. Select the Movement tab (Figure 72-1).
3. Select the Can Be Embarked Upon check box.
4. Click Configure. The Configure Embarkation dialog box opens (Figure 72-2).
5. Click Advanced. The dialog box redisplays. It shows an entity icon in Plan View mode.
6. Optionally, change the view mode to Stealth to see a 3D model of the entity.
7. Optionally, use the mouse or keyboard to zoom in or out or otherwise change your view of the entity.
8. Click the Add button (+) above the Ingress Points list. An ingress point is added to the list. A vertex is added to the display window at the center of the entity bounding volume (Figure 72-6). The coordinates for the ingress point are displayed at the bottom of the dialog box.

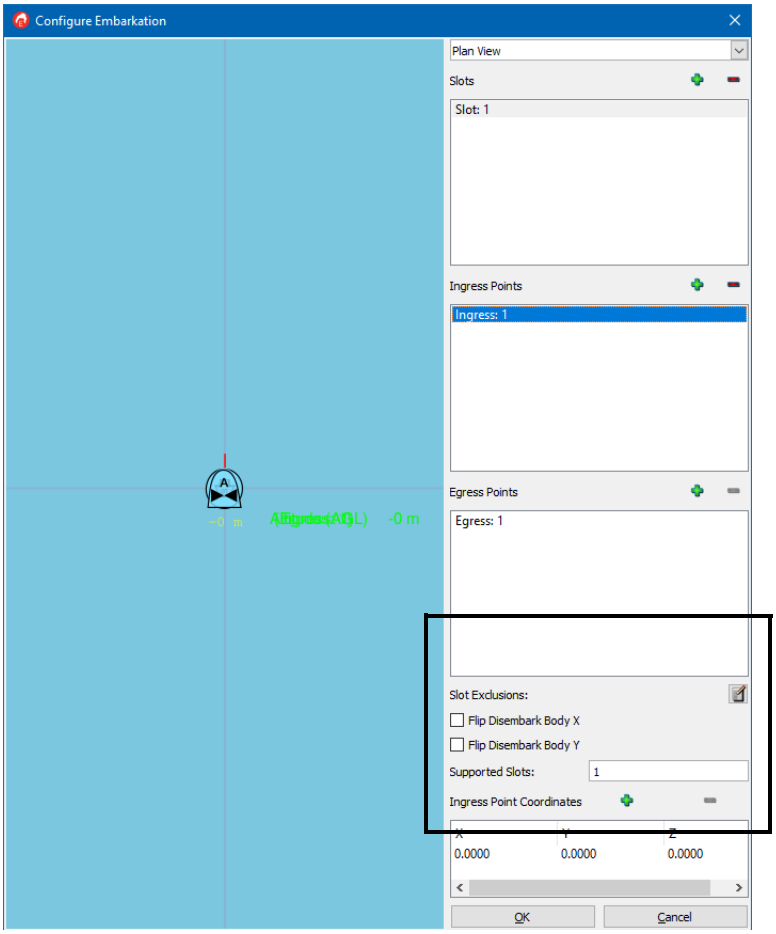


Figure 72-6. Ingress Point

9. Double click the X value for the Ingress point to make it editable.
10. Enter the X value.
11. Repeat for the Y and Z values.
12. If you want to specify a path for the embarking entity
    a. Click the Add button for the Ingress Point coordinates list. A new coordinate is added.
    b. Specify the X, Y, and Z values for the new coordinate. The path is displayed in the window (Figure 72-7).

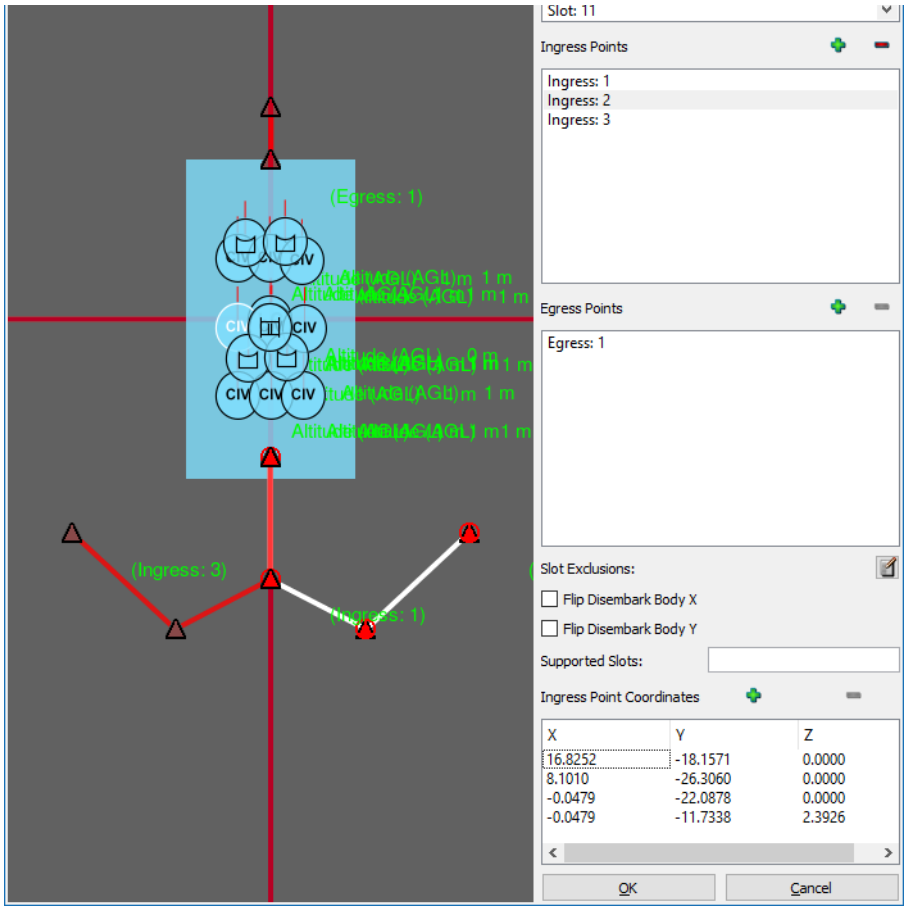


Figure 72-7. Ingress point configuration

13. Optionally in the Supported Slots field, type the numbers for the slots that should support this ingress point. If you leave the field blank, it supports all slots. If you specify more than one slot, separate the numbers with a space.
14. To change the side of a parent simulation object that the embarked simulation objects exit on, select the Flip Disembark Body X check box, the Flip Disembark Body Y check box, or both (Figure 72-3).
15. Click OK.

72.4. Embarkation Slot Tutorial

This tutorial shows how to configure an entity for embarkation. The supertanker does not have slots configured for embarkation in the default VR-Forces *EntityLevel.sms*. However, it has two helipads on the deck, which makes it ideal for embarkation. We will add embarkation slots, ingress points, and egress points to the supertanker. Then we will demonstrate the new slots in a scenario.

72.4.1. Configure the SuperTanker for Embarkation

To configure the supertanker for embarkation:

1. Start the Simulation Object Editor.
2. Choose **File** → **New Simulation Model Set**. The New Simulation Model Set dialog box opens.
3. Select the New Simulation Model Set option.
4. Click Next. The Simulation Model Set Type dialog box opens.
5. Select the Extend Existing Simulation Model Set option.
6. Click Next. The Extend Existing Simulation Model Set dialog box opens.
7. Select *EntityLevel.sms*.
8. Click Next. The New Model Set Name dialog box opens.
9. Specify a name for the new SMS.
10. Click Finish. The new SMS is created and loaded.
11. In the list of entities, select Super Tanker. It is read-only because it is included from *EntityLevel.sms*.
12. At the top of the Simulation Object Editor window, click Promote. The Super Tanker is now editable.
13. Select the Movement tab (Figure 72-1).
14. Select the Can Be Embarked Upon check box.
15. Click Configure. The Configure Embarkation dialog box opens (Figure 72-2).

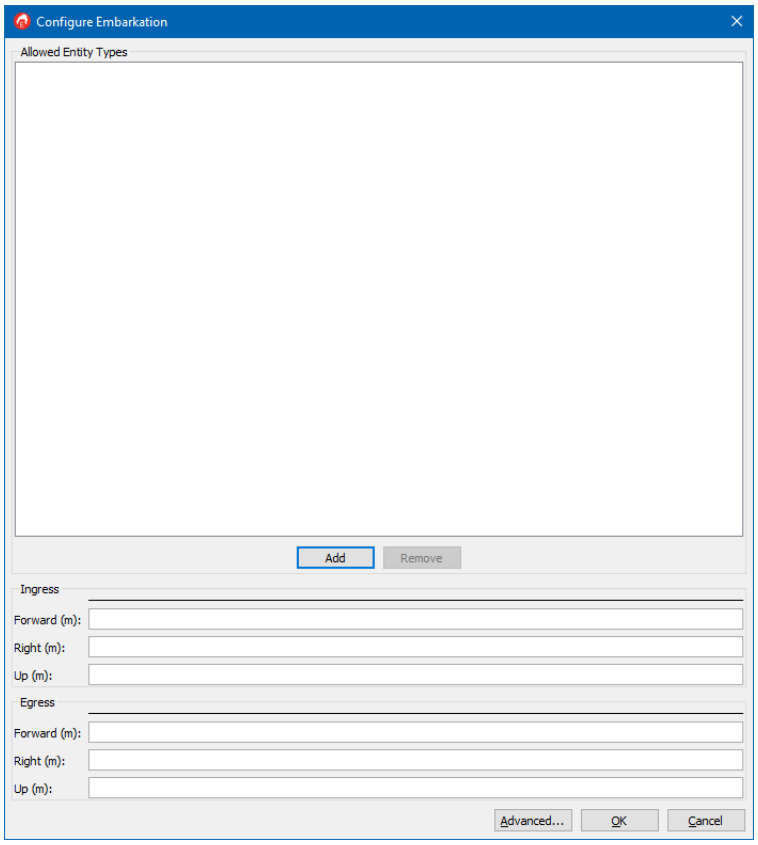


Figure 72-8. Configure Embarkation dialog box

16. Click Add. The Add Type dialog box opens.
17. In the list of entity types, select Rotary Wings (Platform). Rotary Wings (Platform) is listed in the Allowed Entity Types window.
18. Click Advanced. The Configure Embarkation dialog box redisplays (Figure 72-9). We are now ready to configure slots.

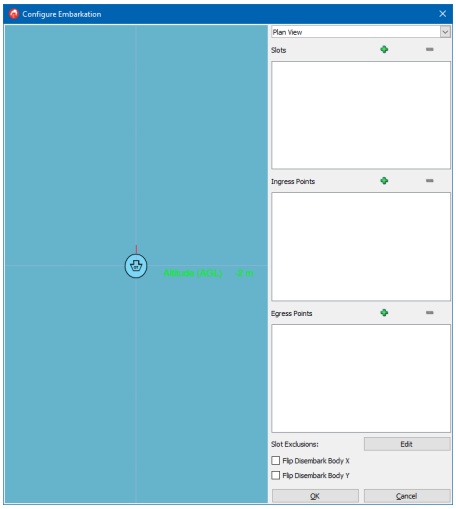


Figure 72-9. Configure Embarkation 2D

72.4.2. Configure Slots for the Supertanker

We want to configure specific slots for the supertanker. We cannot do that in the basic Configure Embarkation dialog box. The key parts of configuring a slot are specifying the object type and the location. Other parameters can provide more specific behaviors.

Add a Slot

To add a slot for the supertanker:

1. In the list at the upper right of the dialog box, change the observer view from Plan View to Stealth. You will now see a zoomed in view of the supertanker.
2. Zoom out so that you can see the forward half of the supertanker. In particular, you want to see the two helipads (Figure 72-10).



Figure 72-10. Stealth view

3. Click the Add button (+) above the Slots window. Slot 1 is added to the list.
4. Select Slot 1.

Specify the Object Type

We need to specify the object types of the entities that can use the slot. In this case, we just need to specify helicopters. However, helicopters have several object type enumerations. Helicopters can have category values of 20-25 (military), 50 (UAV), and 90-92 (civilian). You can specify multiple values in an enumeration using dashes for ranges and separating values with commas. For the helicopter, we want an enumeration that looks like this: 1:2:-1:20-25,50,90-92:-1:-1:-1. We will let the Simulation Object Editor build a basic object type. Then we will customize it.

To: specify the object type:

1. On the Object Type line, click the Select button. The Change Enumeration dialog box opens.

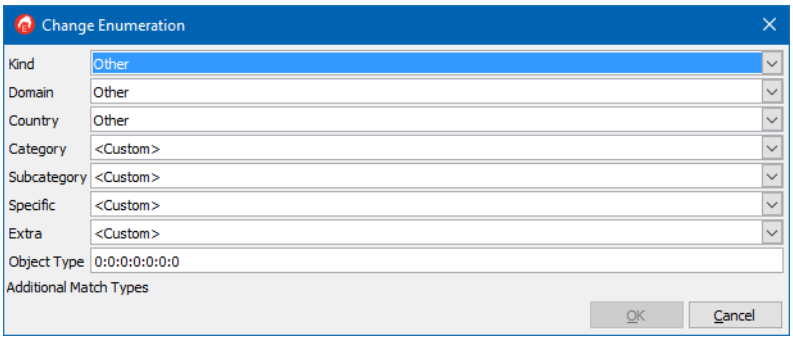


Figure 72-11. Change Enumeration dialog box

2. In the Object Type dialog box, in the Kind list, select Platform (1).
3. In the Domain list, select Air (2).
4. In the Country list, select All (-1).
5. In the Category list, leave as Other (0).
6. For all the other lists, select All (-1). The results in an enumeration of 1:2:-1:0:-1:-1:-1 (Figure 72-12).

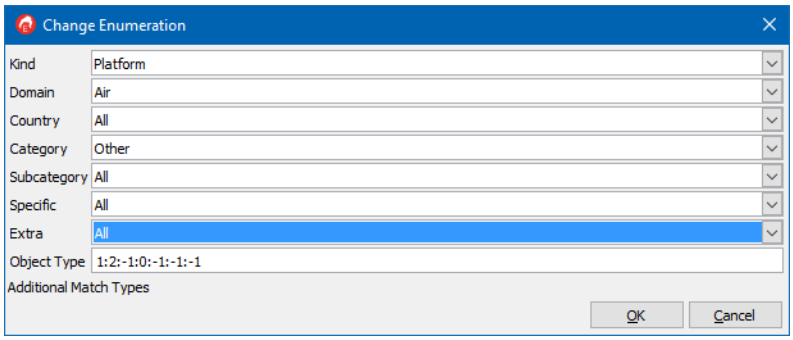


Figure 72-12. Completed base enumeration

7. Click OK. The enumeration is listed in the Object Type box.
8. In the category field, replace 0 with 20-25,50,90-92. The completed object type is: 1:2:-1:20-25,50,90-92:-1:-1:-1. This object type will allow any type of helicopter to land in this slot.

Specify the Location of the Slot

We must specify the X, Y, and Z location of the slot relative to the origin of the supertanker. You will see a yellow number (-2) at the center of the ship.

To specify the slot location:

1. Take a look at the location of the helipad to the right of the center line of the supertanker (starboard). Estimate the distance from the middle of the ship to the helipad.
2. Type your estimated distance for the Forward (Y) value. A yellow number is displayed on the center line of the ship.
3. Adjust the forward value to get to the proper distance (69 is a good value)
4. For the Right (X) coordinate, type an estimated value for the distance to the right of center (starboard). (20 is a good value.) You should now see the number centered on the H of the helipad .
5. Type a value for Up (Z). If you leave this blank, the helicopter will clamp to the deck of the tanker, but first it will sink to 0 meters, which doesn't look too good. 14 meters is a good value for this coordinate.
6. Clear the Jump to Location check box. The dialog box should look like Figure 72-13.



Figure 72-13. Slot 1

Add a Second Slot

Add a second slot. Use the same object type and forward and up values. The Right coordinate should be -20.

72.4.3. Add Ingress Points

Ingress points define locations relative to the parent entity that an embarking entity moves to before it moves to its slot. While you do not have to define one, it is important that you do so. If there are no ingress points, when you task an entity to embark, it teleports directly to a slot. It does not simulate movement towards the parent.

To add an ingress point:

1. Click the Add button (+) above the Ingress Points window. A red vertex marker is added to the center of the ship.
2. In the Ingress Points Coordinates window, double-click the X value to make it editable.
3. Set it to 20. The vertex marker moves to this location.
4. Set the Y value to 69.
5. Set the Z value to 40. This will cause a helicopter to fly to a point above the helipad (the ingress point) and then land on the helipad.
6. Add a second ingress point. Set the X, Y, and Z to -20, 69, and 40. The completed ingress points look like Figure 72-14.

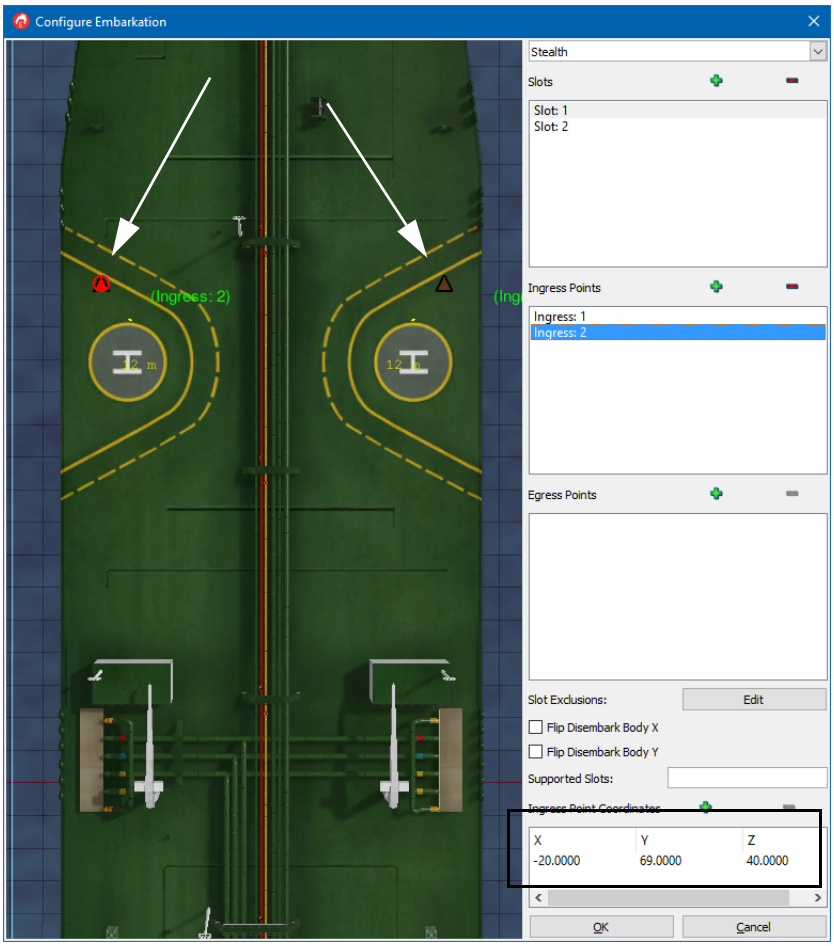


Figure 72-14. Completed ingress points

72.4.4. Add Egress Points

Egress points define locations relative to the parent entity that a disembarking entity moves to after it leaves its slot. As with ingress points, egress points help manage realistic looking behavior for entities as they disembark. Without them, the entity simply stays where it is in its slot, but changes its state to disembarked.

To add an egress point:

1. Click the Add button (✚) above the Egress Points window. A red vertex marker is added to the ship.
2. Specify X, Y, and Z values. Set the X, Y, and Z to 20, 69, and 40. This will cause a helicopter to disembark and fly to a point above the helipad before beginning its next task. The vertex marker moves to show the location of the egress point.
3. Add a second egress point. Set the X, Y, and Z to -20, 69, and 40.

72.4.5. Save the New Configuration

To save the configuration:

1. In the Configure Embarkation dialog box, click OK.
2. In the main SOE window, choose **File** → **Save Simulation Model Set**.

72.4.6. Test the Embarkation Configuration

Now you can test the new embarkation configuration for the supertanker.

To test the embarkation configuration:

1. Start VR-Forces.
2. Create a new scenario using *EntityLevel.sms* and a terrain that has an ocean area, such as Makland or Ala Moana.
3. Place a supertanker in the water.
4. Add three helicopters. Put one on one side of the tanker and two on the other side. Use a variety of military and civilian helicopters. Set their altitude to 60 - 100 meters.

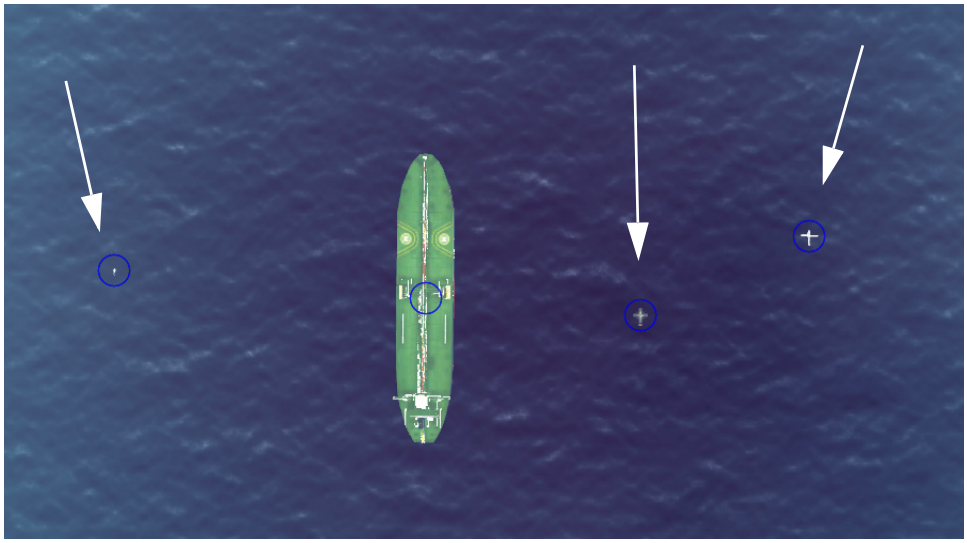


Figure 72-15. Test scenario

5. Give each helicopter a plan to embark on the supertanker. Stagger the times at which they start their embark task so that they do not all start flying at once.
6. Run the scenario. Two helicopters should be able to embark in the two slots. The third should send a message to its console that it cannot complete its task because there are no slots available (Figure 72-16).

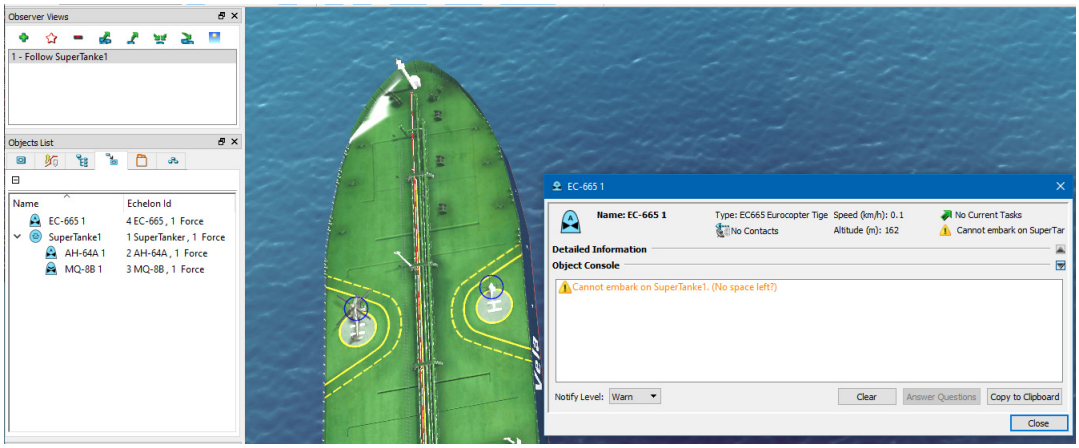


Figure 72-16. Helicopters embarked on Super Tanker



73. Adding DI-Guy Models to VR-Forces

This chapter explains how to take DI-Guy models created with the DI-Guy SDK and integrate them into VR-Forces.

Adding New DI-Guy Models to VR-Forces 73-2
Copy DI-Guy Files to the Data Directory 73-2
DI-Guy Configuration Files 73-3
Adding a New DI-Guy Animation 73-3
How VR-Forces Maps DI-Guy Models 73-5
Add a New DI-Guy Entity in the SOE 73-5

73.1. Adding New DI-Guy Models to VR-Forces

VR-Forces supports DI-Guy characters and includes many lifeform characters. However, MAK customers sometimes want to add their own DI-Guy characters. The general process for adding a new character is as follows:

1. Create the new character using DI-Guy software and copy the required files to the *./data* directory.
2. Add a model definition and element definition for the new character.
3. If the character has animations, add them to *character_animations_table.mtl*.
4. Create a new VR-Forces character in the Simulation Object Editor.

73.1.1. Copy DI-Guy Files to the Data Directory

VR-Forces stores files for DI-Guy characters in *./data/Lifeforms/DIGuy* and its subdirectories. After you create a new character using DI-Guy software, you must copy the files for this character to these directories.

DI-Guy documentation recommends that all customer content be placed in *$(DIGUY)/custom*. To follow this convention, create a directory called *./data/Lifeforms/DIGuy/custom* and put your files in the appropriate subdirectories under it.

VR-Forces can visualize custom DI-Guy characters based on DIS enumerations mapped to element definitions or by recognizing custom DI-Guy PDUs.

- If you want to map your custom characters to element definitions, then you must add new simulation models in the Simulation Model Editor. This will automatically create the complementary model and element definitions and object type mappings.
- If you want to visualize characters based on custom DI-Guy PDUs without creating element definitions or object type mappings, place the DI-Guy files in the *./data/Lifeforms/DIGuy/custom* directory as described in the previous paragraph. No further configuration is needed.

73.2. DI-Guy Configuration Files

VR-Forces can specify the character, appearance, and animation values for DI-Guy models used in 3D visualization applications such as VR-Forces and VR-Vantage. A model's character determines the kinds of movement it can make. The appearance determines what it looks like. The animation specifies the movements that the character makes.

In the VR-Forces front-end, you can specify the animation with the DI-Guy Animation task. You can set the appearance with the DI-Guy Appearance set data request. For details about specifying DI-Guy characteristics in the front-end, please see "DI-Guy Animation," on page 28-7, "DI-Guy Characteristics (Appearance, Head, Body Weight)," on page 34-15, and "DI-Guy Hand Item," on page 34-15.

You can specify the default character, appearance, and animation for a lifeform entity in the Simulation Object Editor. (For details, please see "DI-Guy Animation," on page 28-7 and "Editing a Lifeform's Visual Model and Animation," on page 69-11.) The DI-Guy mappings are stored in the entity's *.entity* file.

The DI-Guy animations available to lifeforms are controlled by *./appData/settings/vrfSim/character_animations_table.mtl*. This information is used to populate the lists in the Simulation Object Editor. You can edit this file to add additional DI-Guy characters to those already configured for VR-Forces. The syntax of entries is described in the file.

Any characters you add to *character_animations_table.mtl* must be valid DI-Guy characters. You cannot arbitrarily make up new types of characters. If you create new characters using a supported version of the DI-Guy SDK, VR-Forces will support these characters.

73.2.1. Adding a New DI-Guy Animation

This section explains how to add a new DI-Guy animation to VR-Forces. It assumes that you have the DI-Guy software and have created the new animation.



The new animation must be made with the same version of DI-Guy supported by VR-Forces. Please see *VR-Forces Release Notes* for the currently supported version.

To add a new DI-Guy animation:

1. Create the new animation in DI-Guy. You will create an animation and add it to a character's action table. Make sure that the name of the animation file matches the name of the action you added to the action table. For example, if the name of the animation is layEgg, but you call the action lay_egg, it will not work.



There is a 1-1 relationship between characters and their action tables. You cannot use an animation based on a chicken character in a cow character's action table.

2. Add the *motion_name*.bdm file for the animation to *./data/Lifeforms/DIGuy/motions*.
3. Add the *action_table*.cfg file to *./data/Lifeforms/DIGuy/config/diguy*.
4. Add the new animation to *./appData/settings/vrfSim/character_animations_table.mtl*. This table determines whether an animation can be assigned to a character using the DI-Guy Animation task (`is-taskable-animation  True`) or whether it is used based on the entity's behavior. For example, if a character is moving faster than a certain speed, it is given a running animation. An entry for our lay egg example would look like the following:

```
(chicken
  (display-name "Chicken")
  (animations
    (animation
      (animation-name "lay_egg")
      (name-to-be-displayed "Lay an egg")
      (is-taskable-animation  True)
      (animation-variant "normal")
    )
    (animation
      (animation-name "feeding")
      (name-to-be-displayed "Feed")
      (is-taskable-animation  True)
      (animation-variant "normal")
    )
...
```

73.3. How VR-Forces Maps DI-Guy Models

Ordinarily, VR-Forces chooses animations to play for a DI-Guy based on the lifeform's speed, position, and similar attributes. However, VR-Forces can specify the DI-Guy state and appearance with the Set DI-Guy Appearance set data request and the Set DI-Guy Animation task. Once you explicitly set the DI-Guy appearance or animation, VR-Forces no longer tries to choose animations and expects to be told how to render that lifeform for the remainder of the simulation.

73.4. Add a New DI-Guy Entity in the SOE

Add a new entity in the Simulation Object Editor, as described in "Creating a New Simulation Object," on page 68-26.

When you specify the 3D model, the list of character names is drawn from the display-name attribute in the entries in *./appData/settings/vrfSim/character_animations_table.mtl*. Therefore, it must have been updated before you can specify the 3D model in the Simulation Object Editor.



74. The Object Parameter Database

This chapter provides a detailed description of the object parameter database. It will be most useful to VR-Forces users who need to develop and configure new components.

The Object Parameter Database 74-2
VR-Forces Classes and Object Parameters 74-3
Object Types 74-4
Parameter Types 74-8
Local and Remote Subcomponents 74-9
Component Descriptors and Parameters 74-13
Common Elements of Component Descriptors 74-13
Configuring the Priority of Components 74-16
Platform Files 74-16
System Definition Files 74-17
Creating a System Definition File 74-17
Configuring System Connections 74-18
Adding Variable Bindings to Platform and System Files 74-22
Adding a Variable Binding 74-23
Making Variables Editable in the Simulation Object Editor 74-24
Adding Platform Variable Bindings to the SOE 74-24
Adding System Definition File Variables to the SOE 74-26
State Properties and Parameter Properties 74-26
Adding State Properties and Parameter Properties 74-27
Getting and Setting State Properties Using Lua Scripts 74-29
Displaying State Property Values in the Information Dialog Box 74-29
An Example of Adding State Properties 74-30
Configuring Units 74-39
Configuring Tactical Graphics 74-39
Pathname Interpretation in Simulation Model Set Files 74-40

74.1. The Object Parameter Database

VR-Forces simulation objects and tactical graphics have many parameters. Parameter values are stored in files organized in simulation model sets (SMS) that VR-Forces reads when you load a scenario or create new objects. We refer to these files collectively as the object parameter database (OPD). The object parameter database specifies both the characteristics of a simulation object and starting state values. You can modify the behavior of a simulation object by changing the parameters for that type of simulation object. You can add new types of simulation objects and new behaviors to VR-Forces by adding new files to the object parameter database.

Before you make any changes to the object parameter database, make a backup copy of the file you are changing.

Figure 74-1 illustrates the organization of an entry in the OPD. The primary definition of a simulated object, such as an entity, a tactical graphic, or a cultural feature, is stored in a file with the extension *.entity*. The *.entity* file specifies the object type enumeration, marking text, organizational information, and other parameters. It references a platform file (extension *.ope*), that provides a set of generic components for the object type, such as ground platform, surface platform, or line object. Each *.entity* file also specifies the systems that allow it to interact with the simulated environment, such as sensors, weapons, and movement systems.

Systems are configured in system definition files. System definition files group related component and connection specifications into convenient packages. The components reference additional configuration files. The system definition files also make it possible to add systems to simulation objects in the Simulation Object Editor.

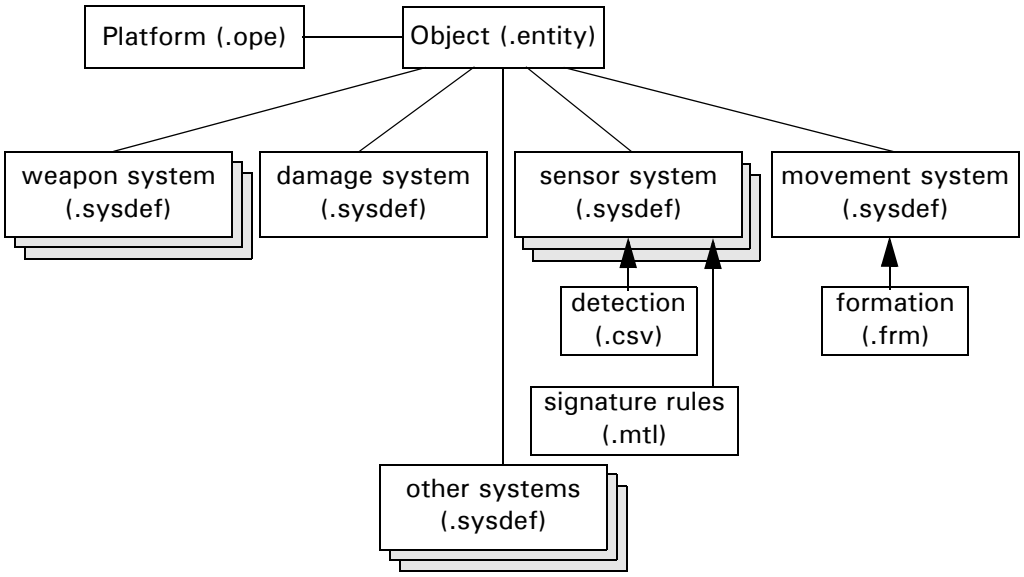


Figure 74-1. Object definition

74.1.1. VR-Forces Classes and Object Parameters

When you create any simulated object in VR-Forces, an instance of *DtVrfObject* is created in the simulation engine. Different types of simulation objects require different parameters for simulation and publishing, which are initialized by a subclass of *DtVrfObjectParameters*, and stored in a subclass of *DtVrfObjectBaseStateRepository*. Each *DtVrfObject* "has a" *DtVrfObjectParameters* and a *DtVrfObjectBaseStateRepository*. The type of parameters object to use for simulation object creation is specified in the simulation object's platform file by the **parameter-type** parameter. The state repository is specified in the **state-repository** parameter in the local-objects block. For example, when an entity that references the platform *Ground_Vehicle.ope*, which has a parameter type **ground-vehicle-param**, is created, a *DtVrfGroundVehicleParameters* object is created to read in the initial values from, and a *DtGroundVehicleStateRepository* is used to store runtime values.

74.1.2. Object Types

VR-Forces uses an eight-digit enumeration scheme for specifying objects. The enumerations are based on the seven-digit enumerations used by DIS and the RPR FOM. The additional field distinguishes between individual and unit objects.

This enumeration scheme describes a tree that goes from a more general specification to a more detailed one. Figure 74-2 illustrates a partial object type tree. To build an object type enumeration, start at the Kind and work your way down the tree, adding the number for each element. So, for example, if you follow the tree down to M1A2 you build the enumeration: 1 1 225 1 1 3 0. Since M1A2 is a leaf in the tree, the last field is 0. For more information about the object enumeration scheme, please see the SISO Enumerations Document (*SISO Reference for Enumerations for Simulation Interoperability*).

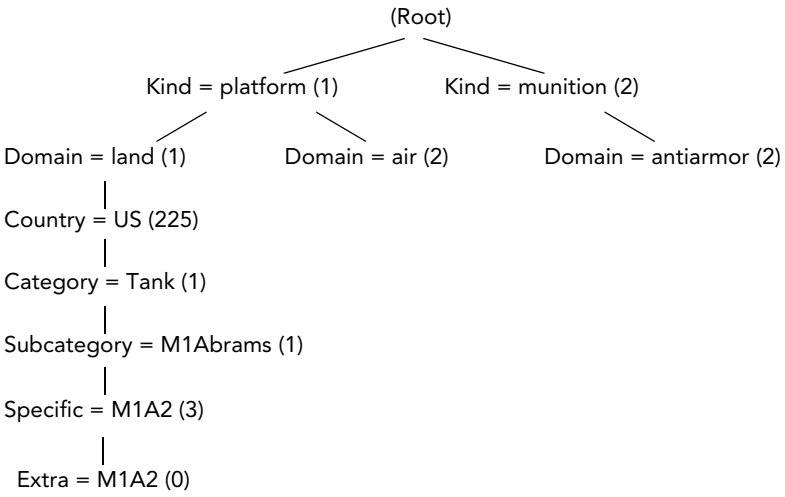


Figure 74-2. Object type enumeration tree

The Object Super-Type

VR-Forces adds a field to the standard 7-digit entity enumeration so that we can manage objects that are not part of the DIS/RPR FOM enumeration scheme, such as units and tactical graphics. We enclose the seven-digit enumeration in parentheses and preface it with the object super-type (Figure 74-3). For example, the object enumeration for an M1A2, described in the previous paragraph, becomes the object type: 1 (1 1 225 1 1 3 0).

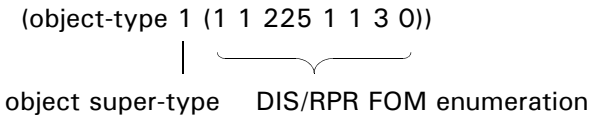


Figure 74-3. VR-Forces object-type parameter

Table 74-1 lists the values VR-Forces can use for the object super-type. For more information, please see *objTypeEnums.h* or "Object Type Enumerations," on page C-9.

Table 74-1: Object super-type field values

Object super-type	Meaning
0	Other.
1	Individual (platform entities, individual tactical graphics).
2	Unit.
3	Disaggregated unit (a unit composed of other simulation objects).
4	Aggregated unit (a unit that does not have subordinate simulation objects).

Published Object Types and Matching Object Types

Each simulation object in the OPD has two object types – the published object type and the matching object type.

The published object type is used as follows:

- When the front-end sends a create simulation object message to the back-end, the back-end uses the object type to determine which model to create.
- When the back-end publishes a simulation object on the network it uses the published object type as the object enumeration.

Each field of a published object type must have a specific value.

The matching object type is used as follows:

- When the back-end receives a request to create a simulation object and it cannot find an exact match for the object type that it is sent, it finds the best match among matching object types to determine which model to create. (For details about the best match method, please see "The Best Match Method," on page 74-7.)
- When the back-end discovers a simulation object on the network, it finds the best match among matching object types so that it can create local representative *DtVrf-Objects*.

The matching type can be the same as the published object type or it can have wildcards (-1).

For example, in VR-Forces, the published object type for an A-10 Thunderbolt is 1:2:225:2:4:1:0. If the matching type for the A-10 Thunderbolt is also 1:2:225:2:4:1:0 and VR-Forces discovers a remote simulation object with that same object type, it can match it to the A-10 Thunderbolt simulation model and can represent it properly in the simulation. However, if VR-Forces discovers a remote simulation object with the object type 1:2:225:2:4:1:1 (which is just a variation of an A-10), it will not be able to match the remote entity to an A-10 model because there is no A-10 match for that object type. (VR-Forces would end up matching the remote entity to a more generic fighter model, as explained in "The Best Match Method," on page 74-7.) But if the A-10's matching object type is 1:2:225:2:4:1:-1, then VR-Forces will be able to match the remote entity to the A-10 model using the best match method.

Object types are specified in the *.entity* file as follows:

```
<simObject objectType="1:1:2:225:2:4:1:0" matchType="1:1:2:225:2:4:1:-1"
  platform="@(platforms-dir)/Fixed_Wing_Aircraft.ope">
```

The **objectType** attribute specifies the published object type; **matchType** specifies the matching object type.

The Best Match Method

As explained in "Object Types," on page 74-4, object types describe a simulation object in increasing specificity as the object type fields move from left to right. Objects in the object parameter database are specified in a hierarchy in which entries at higher levels of the hierarchy have wildcard values (-1s) in object type fields that do not need to be specified at that level of the hierarchy. When VR-Forces looks up an object type, it begins at the root of the tree, working its way down until no better matches are found (the best match method.)

Suppose the object type hierarchy for fighter jets looks like this:

- -1:-1:-1:-1:-1:-1:-1:-1 – match any object.
- 1:1:2:-1:-1:-1:-1:-1 – match any fixed-wing entity.
- 1:1:2:225:2:-1:-1:-1 – match any U.S. attack fixed-wing.
- 1:1:1:2:225:2:4:1:-1 – match any A-10 Thunderbolt. (In VR-Forces, this is the A10 Thunderbolt matching object type.)
- 1:1:1:2:225:2:4:1:0 – match a specific A-10 Thunderbolt. (In VR-Forces, this is the A10 Thunderbolt published object type.)

As you replace each wildcard value (-1) with a specific enumeration, the scope of the object type narrows. Given this hierarchy, if VR-Forces discovers a simulation object with the object type 1:1:2:225:2:11:0:0, it starts trying to find the best match as follows:

1. The discovered object type matches the top level - any object, because all fields are wild cards.
2. The discovered object type matches the fixed-wing object type, because the first three fields match and the rest are wild cards.
3. The discovered object type matches the U.S. attack fixed-wing object type.
4. The discovered object type does not match the A-10 Thunderbolt matching object type. Although the first five fields match, the sixth is different and is not wild carded.

The best match is the U.S. attack fixed-wing object type, so VR-Forces will use the generic U.S. attack fixed-wing model to represent the discovered simulation object.

74.1.3. Parameter Types

In addition to specifying an **object-type** for each object, you must specify a **parameter-type** for each object. The **parameter-type** determines the parameters and components that can be specified for an object. For example, a rotary-wing or fixed-wing entity can have a **nose-support** parameter, but a ground vehicle cannot. Figure 74-4 illustrates the object parameter type hierarchy.

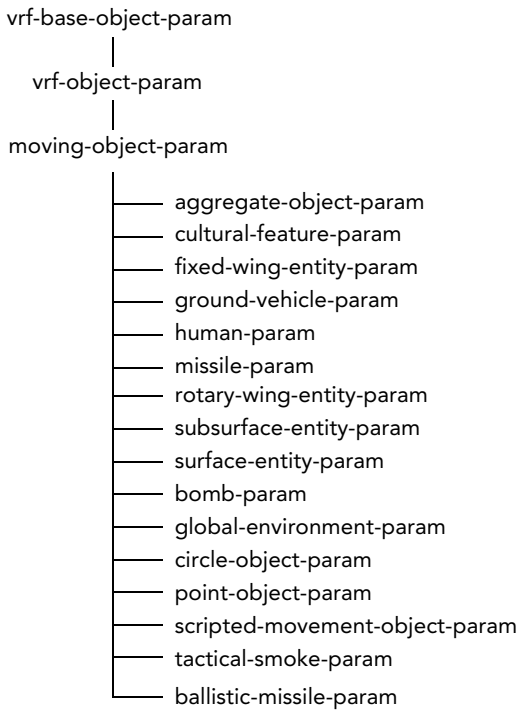


Figure 74-4. Object parameter hierarchy

The **parameter-type** specifies the set of parameters appropriate to a simulation object, and the object state repository maintains state information for a simulation object, so the **parameter-type** and state repository must match. For more information, please see "The State Repository Hierarchy," on page 74-12.



The **object-type** is specified in an object's *.entity* file. The parameter-type is specified in the platform (*.ope*) file.

74.1.4. Local and Remote Subcomponents

As explained in "Local Objects and Remote Objects," on page 3-9, each VR-Forces back-end simulates objects itself (local objects), and maintains state information about remote objects. All of these objects are instantiated as *DtVrfObjects*. The *DtVrfObject* class has a "has a" relationship with the following classes:

- *DtSimObjectStateRepository*
- *DtSimObjectNetInterface*
- *DtSimTaskManager*
- *DtSimComponentManager*
- *DtVrfObjectPlanManager*.

These classes are referred to as subcomponents of the object. Table 74-2 maps the subcomponent classes to the parameters that specify them.

Table 74-2: Subcomponent classes and parameters

Class	Parameter
DtSimObjectStateRepository	state-repository
DtSimObjectNetInterface	net-interface
DtOrganizationManager	organization-manager
DtSimTaskManager	task-manager
DtSimComponentManager	component-manager
DtVrfObjectPlanManager	plan-manager

VR-Forces creates different subcomponents for an object depending on whether it is a local object or a remote object. The subcomponents are specified in the **local-objects** and **remote-objects** parameter blocks for that object. Table 74-3 lists the permissible entries for each parameter listed in Table 74-2.

Table 74-3: Permissible subcomponent parameters

Parameter	Permissible values
state-repository	◆ ballistic-missile-entity-state-repository ◆ bomb-entity-state-repository ◆ cultural-feature-state-repository ◆ fixed-wing-entity-state-repository ◆ ground-entity-state-repository ◆ human-state-repository ◆ missile-entity-state-repository ◆ rotary-wing-entity-state-repository ◆ surface-entity-state-repository ◆ subsurface-entity-state-repository ◆ vrf-aggregate-state-repository ◆ vrf-moving-object-state-repository ◆ vrf-object-base-state-repository ◆ vrf-object-state-repository ◆ vrf-overlay-object-state-repository ◆ vrf-tactical-smoke-state-repository
net-interface	◆ aggregate-local-entity-net-interface ◆ engineering-object-local-net-interface ◆ individual-local-entity-net-interface-with-properties ◆ local-control-object-net-interface ◆ local-control-object-net-interface-with-properties ◆ local-control-object-net-interface ◆ local-environmental-net-interface ◆ local-tactical-smoke-net-interface ◆ pseudo-aggregate-local-entity-net-interface-with-properties ◆ engineering-object-remote-net-interface ◆ vrf-aggregate-remote-entity-net-interface-with-properties ◆ vrf-individual-remote-entity-net-interface ◆ vrf-individual-remote-entity-net-interface-with-properties ◆ vrf-remote-environmental-net-interface ◆ vrf-remote-environmental-net-interface-with-properties ◆ vrf-remote-tactical-smoke-net-interface
task-manager	◆ local-task-manager
component-manager	◆ component-manager
plan-manager	◆ vrfobject-plan-manager

Every object must have a state repository specified. Objects that should be published need a net interface. Other subcomponents are optional. However, for most simulation objects, you will want to specify a task manager, plan manager, and component manager so that you can control the simulation object and give it tasks.

The local and remote objects parameters are specified in the platform (*.ope*) files. Therefore they apply to all simulation objects that reference a particular platform. Use the existing entries in the object parameter database as guidance if you create a new platform or change an existing platform. A typical entry for local and remote objects is:

```
(local-objects
  (state-repository  "ground-entity-state-repository")
  (state-repository-min-tick-period  -1.000000)
  (state-repository-min-tick-period-variance  -1.000000)
  (net-interface  "individual-local-entity-net-interface-with
  properties")
  (net-interface-min-tick-period  -1.000000)
  (net-interface-min-tick-period-variance  -1.000000)
  (task-manager  "local-task-manager")
  (component-manager  "component-manager")
  (plan-manager  "vrfobject-plan-manager")
  (nav-interface "active-bot")
)
(remote-objects
  (state-repository  "vrf-moving-object-state-repository")
  (state-repository-min-tick-period  -1.000000)
  (state-repository-min-tick-period-variance  -1.000000)
  (net-interface  "vrf-individual-remote-entity-net-interface-with
  properties")
  (net-interface-min-tick-period  -1.000000)
  (net-interface-min-tick-period-variance  -1.000000)
  (task-manager  "")
  (component-manager  "")
  (plan-manager  "")
  (nav-interface "dynamic-obstacle")
)
```

The State Repository Hierarchy

Each VR-Forces object must have a state repository to maintain essential state information about the object, such as its current location, orientation, appearance, and so on. Therefore, each object must specify a state repository and the state repository must match the **parameter-type** for the object. (The state repository and **parameter-type** are specified in the platform file.)

Table 74-4 lists parameter types and the state repository type required.

Table 74-4: State repository and parameter types

State Repository Type	Parameter Type
ballistic-missile-entity-state-repository	ballistic-missile-param
bomb-entity-state-repository	bomb-param
cultural-feature-state-repository	cultural-feature-param
fixed-wing-entity-state-repository	fixed-wing-entity-param
ground-entity-state-repository	ground-vehicle-param
human-state-repository	human-param
missile-entity-state-repository	missile-param
rotary-wing-entity-state-repository	rotary-wing-entity-param
surface-entity-state-repository	surface-entity-param
subsurface-entity-state-repository	subsurface-entity-param
vrf-aggregate-state-repository	aggregate-object-param
vrf-moving-object-state-repository	moving-object-param
vrf-overlay-object-state-repository vrf-object-state-repository	vrf-object-param
vrf-tactical-smoke-state-repository	tactical-smoke-param

The Net Interface Hierarchy

VR-Forces manages network relationships of local and remote simulation objects through the network interface classes. Choose the set of local and remote network interface types that most closely represent the type of object being simulated. For example, for an individual vehicle such as a tank, choose individual-local-entity-net-interface for the local network interface and vrf-individual-remote-entity for the remote network interface. Please see Appendix C, *Object Parameters* for tables that list the network interfaces used for the simulation objects provided with VR-Forces.

74.2. Component Descriptors and Parameters

The parameters for a simulation object include a list of all the sensor, controller, and actuator components it uses. (For details, please see "VR-Forces Uses a Component Architecture," on page 13-7.) These components might be in the OPE file or in a system definition file. If there are no components of a certain type (for example, no sensors), you can omit the list. (In other words, you do not need a placeholder list for unused components.)

A component descriptor is a text block in a platform or system definition file that describes a component (see Example 74.1). A component descriptor must include the type of the descriptor and the type of the component that it is describing. A component descriptor can also contain parameters for the component. "Common Elements of Component Descriptors," on page 74-13 describes the construction of a component descriptor.

Components can receive data from, and convey data to, other components through ports. These ports are specified in platform files and system definition files.

A component can create a port group, which groups a set of related ports together. For example, the automotive-control port group groups the throttle, steering, and brake ports. Use of port groups simplifies specification of connections because you just have to specify a port group instead of many individual ports. For a detailed discussion of ports and port groups, please see API documentation.

The port group also provides a mechanism for ensuring that only one data provider attempts to put values on those ports each simulation frame. For more information, please see "Configuring the Priority of Components," on page 74-16.

74.2.1. Common Elements of Component Descriptors

The common elements of component descriptors are:

- Name – The first element in a component descriptor specifies the component's name. This name can be whatever you want, but it must be unique within the simulation object or the system the component is defined in. It gets used in the connections section of the simulation object's parameters. Other components may use this name to find the specified component. In Example 74.1, the name of the first component is move-along (line 2).
- component-descriptor-type – The component descriptor factory creates the type of component descriptor specified by this type. This lets the system know what types of additional data may be present. In Example 74.1, the component descriptor type for the move-along component is ground-move-to-descriptor (line 3).
- component-type – The component factory creates the type of component specified in this entry. In Example 74.1, the component type is ground-auto-move-along-controller (line 4).



The component-descriptor-type and component-type values are specified in code. You cannot make them up. Please see *compTypes.h* and *compDescTypes.h*.

- connections (optional) – The connections (Example 74.1, line 44) entry specifies how the components connect to one another (through ports, port groups, or both). Components attach to each other by connecting output and input ports to one another, or output port groups to input port groups. Since a component can have more than one port, port group, or both, the name of the port or port group to use is also specified. If a simulation object does not have any components that need to connect to one another, then you can omit this entry. Items in the connections entry can be in one of the following forms:
  - Specify a port-to-port connection:

    ```
    connect component_name:output_port component_name:input_port
    ```

  - Specify a port group-to-port group connection:

    ```
    connect component_name:output_port_group
      component_name:input_port_group
    ```

    The connections in Example 74.1 use this format.

  - Specify a port-to-port connection, in which one or both of the ports exists inside of a group, as follows:

    The input port belongs to an input port group:

    ```
    connect component_name:output_port
      component_name:input_port_group:input_port
    ```

    The output port belongs to an output port group:

    ```
    connect component_name:output_port_group:output_port
      component_name:input_port
    ```

    Both input and output ports belong to groups:

    ```
    connect component_name:output_port_group:output_port
      component_name:input_port_group:input_port
    ```



The port and port group names are defined in code for each component. Please see class documentation for the names of port and port groups used by a given component.

Example 74.1 shows component descriptors for the Move Along and Patrol Between Points controllers and the Powertrain actuator for a ground vehicle (from the *ground-tracked.sysdef* file.) The Powertrain actuator simulates movement of a ground vehicle based on the throttle, brake, and steering inputs it receives from the automotive controllers. The example also shows how the output port groups for the controllers are connected to the input port group of the powertrain.

Example 74.1

```
1    (controllers
2      (move-along
3        (component-descriptor-type "ground-move-to-descriptor")
4        (component-type  "ground-auto-move-along-controller")
5        (min-tick-period  -1.000000)
6        (min-tick-period-variance  -1.000000)
7        (tick-period-uses-real-time False)
8        (process-state-repository-name  "")
9        (process-state-repository-type  "")
10       (is-enabled True)
11       (near-distance  25.000000)
12       (at-distance  1.000000)
13       (approach-speed  2.000000)
14     )
15     (patrol-between-points
16       (component-descriptor-type "ground-move-to-descriptor")
17       (component-type  "ground-auto-patrol-between-controller")
18       (min-tick-period  -1.000000)
19       (min-tick-period-variance  -1.000000)
20       (tick-period-uses-real-time False)
21       (process-state-repository-name  "")
22       (process-state-repository-type  "")
23       (is-enabled True)
24       (near-distance  25.000000)
25       (at-distance  1.000000)
26       (approach-speed  2.000000)
27     )
28   ) ;; end controllers
29   (actuators
30     (powertrain
31       (component-descriptor-type "automotive-component-descriptor")
32       (component-type  "automotive-actuator")
33       (min-tick-period  -1.000000)
34       (min-tick-period-variance  -1.000000)
35       (tick-period-uses-real-time False)
36       (process-state-repository-name  "")
37       (process-state-repository-type  "")
38       (is-enabled True)
39       (art-part-list )
40       (check-terrain-collisions False)
41       (use-high-fidelity-ground-clamping True)
42     )
43   ) ;; end actuators
44   (connections
45     (connect move-along:automotive-control powertrain:automotive-
         control)
46     (connect patrol-between-points:automotive-control
         powertrain:automotive-control)
47   ) ;; end connections
```

The connection in line 47 would read: the move-along component sends output to its automotive-control output port group. The powertrain receives the move-along controllers output through its automotive-control input port group.

74.2.2. Configuring the Priority of Components

Each simulation frame, VR-Forces ticks each simulation object's components in the order in which they are listed in the platform and system definition files. If a component believes it should be providing data, it takes control of the receiving component's ports or port group. Once a component acquires the ports, no other component can send data to that component during that frame. Therefore the order in which components are listed provides a natural priority scheme.

For example, among the ground movement controllers, the collision avoidance controller is placed on the controller list before any of the other movement controllers. If this controller needs to provide input to the ground automotive actuator to avoid a collision, it takes control of the automotive-control port group on the next tick and overrides the controller that previously owned the port group (such as Move To or Move Along).

74.3. Platform Files

Platform files (extension *.ope*) are the foundational configuration files for all simulated objects. They provide the basic description of each type of simulation object that you can create, such as a ground vehicle, surface vessel, or human. They also define types of tactical graphics. All platform files are part of an SMS. A platform file includes all of the components that every simulation object of that type needs to have, such as collision avoidance, radios, script controllers, and so on.

A platform file does not include systems that vary among specific examples of the platform, such as weapon systems, movement systems, and damage systems.

A platform file uses variables for the parameters that need to be different for specific examples of a platform. For example, all ground vehicles have a maximum speed, but tanks and civilian vehicles have a different value for this parameter. The values for these variables get set in the Simulation Object Editor.

You can create and edit platform files using a text editor. Platform files use MTL syntax. Most VR-Forces users will not need to create new platform files. You may need to add components or change parameters to be variables.

74.4. System Definition Files

The Simulation Object Editor allows VR-Forces users to easily add component systems, such as weapons, sensors, and movement, to simulation objects without needing to understand the components, configuration files, and connections required to make these systems work. These systems are defined in system definition files (*.sysdef*). System definition files are referenced in a simulation object's *.entity* file. They are read by the SOE and can be edited in it.

System definition files have two major sections – data and metadata. The data is a list of the components and connections required to implement the system for a simulation object.

The metadata explains to the Simulation Object Editor how to present the system to the user and how to integrate it into the entity model.

Weapon systems and damage systems have two system definition files, a template file (*.template_sysdef*), which specifies components shared by similar types of weapons or damage systems, and a weapon or platform-specific system definition file (*.sysdef*).

74.4.1. Creating a System Definition File

If you want to create a new system definition file, MAK recommends that you copy an existing file and use it as a starting point for adding the components that you want to configure into a system. You can use the SOE to do this or a text editor.

The format of a system definition file is as follows:

```
(system-name
  (controllers
  )
  (actuators
  )
  (connections
  )
  (resources
  )
  (meta-data
    (system-name "Spot Report Generator")
    (system-description "Allows the entity to send spot reports though
 its radio when its sensors detect other entities in the scenario.")
    (allowed-state-repository-types "all")
    (system-categories "other")
    (meta-data-entry-list
    )
    (parameter-data-list
    )
  )
)
```

As you can see, system definition files use MTL syntax. (For information about MTL syntax, please see "MAK Technologies Lisp (MTL) Files," on page A-3.) The system-name is an arbitrary name that uniquely identifies the system. The component sections list the various controllers, actuators, sensors, and resources that are needed to build the system. The connections section lists the connections among the components.

You can call the system whatever you want, but the name must not start with a number.

74.4.2. Configuring System Connections

Within each system definition file, you list the connections among the components in the file. (For general information about connections, please see "Common Elements of Component Descriptors," on page 74-13.) For systems that are self-contained, for example movement systems, there is no need to connect to components in other systems. However, some systems need to connect to components outside their system. For example, a visual sensor needs a list of object types to detect. This section explains how to make these out-of-system connections.

To understand how system definition files and *.entity* files are integrated, we will look at:

- A `componentSystem` element in a *.entity* file.
- The related system definition file.
- The connection section in the system definition file.

The M1A2, which uses the Ground_Vehicle platform, is configured with a visual sensor (in *M1A2_Abrams_MBT.entity*) as follows:

```
1    <simObject objectType="1:1:1:225:1:1:3:0"
     platform="platforms\Ground_Vehicle.ope">
2    ...
3    <componentSystem systemName="sensor"
       platform="@(system-dir)/sensors/visual-sensor.sysdef">
4      <bodyPosition paramName="sensor-position" right="0.57"
         forward="-1.14" up="2.1"/>
5      <real paramName="max-range">4000</real>
6    </componentSystem>
```

The sensor component references the system definition file *visual-sensor.sysdef*. The parameters specified provide values for the variables in the system definition file. (For more information, please see "Adding Variable Bindings to Platform and System Files," on page 74-22.)

The system definition file has the following connections:

```
1     (visual-sensor-system
2         ...
3       (connections
4         (connect system:object-types-to-detect
            visual-sensor:object-types-to-detect)
5         (connect visual-sensor:detected-objects system:detected-objects)
6         (connect system:sensor-offset visual-sensor:sensor-offset)
7       )
8       (resources )
9       (meta-data
10        (system-name  "Visual Sensor")
11        (system-description  "Allows an entity to detect other objects
        through visible light.")
12        (allowed-state-repository-types "all")
13        (system-categories "sensor")
14    ...
15    )
```

The system definition file defines connections for its components. The first connection in the list (line 4) means that some external component (`system`) is providing input from its `object-types-to-detect` port to the visual sensor's `object-types-to-detect` port. The second connection specifies that the visual sensor's `detected-objects` port sends data to some external component's `detected-objects` port. When VR-Forces creates a simulation object, it needs to substitute an actual component for the generic "system" component specified in the system definition file. It does this by referring to an auto-connection block in the OPE file

The platform file has a set of auto-connection blocks like the following (from *Ground_Vehicle.ope*):

```
(auto-connection
  (source-controller "target-selection")
  (source-port-name "object-types-to-detect")
  (destination-category "sensor")
  (destination-port-name "object-types-to-detect")
)
```

The auto-connection block in the OPE file is a template that VR-Forces uses to match systems of a specified category to the appropriate controllers. In this example the `target-selection` controller sends data through its `object-types-to-detect` port to the `object-types-to-detect` port of any system that is in the sensor category. Since the visual sensor is in the sensor category, it substitutes the `target-selection` controller for the generic system in building its connections.

Figure 74-5 illustrates the relationships of the OPE file, *.entity* file, and system definition files referenced in this section and how connections are made.

The M1A2 entity, which uses the Ground_Vehicle platform, has a visual sensor (callout 1). The visual sensor has several connections. The one illustrated in the figure (2) takes input from an external source's `object-types-to-detect` port and sends it to the visual sensor's `object-types-to-detect` port (3). In this connection, the external source is represented by `system`. Before the sensor can actually work, this unspecified system must be determined.

When the entity is created, VR-Forces checks all the auto-connections in the OPE file (4). In this case, there is an auto-connection from the target-select controller's `object-types-to-detect` port to the `object-types-to-detect` port of a component whose destination category is `sensor` (5). To make this connection, the unspecified sensor must be determined.

VR-Forces matches the `destination-category` value "sensor" in the OPE file to the `system-categories` value "sensor" in the system definition file and the connections can now be completed (6).

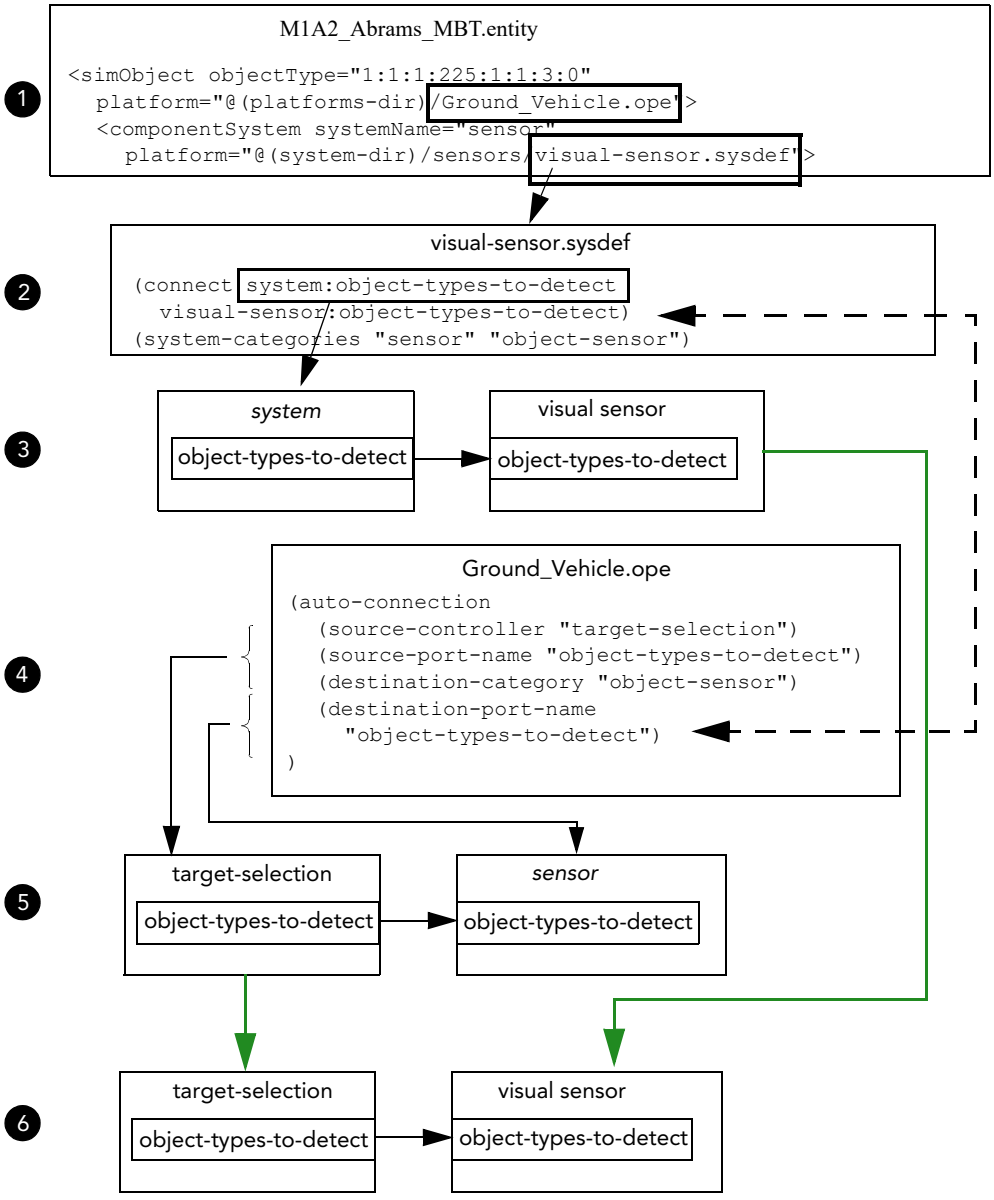


Figure 74-5. External component connection

Systems that use templates must pass information between template files and individual system files. The “all” connection supports this exchange of information, as in the following example:

```
(connections
  (connect system:all main-14_5mm-gun:all)
  (connect main-14_5mm-gun:all system:all)
)
```



Most VR-Forces users add new systems by copying existing ones and changing their parameters. If you add new systems in this way, you will not need to worry about adding connections, since the existing auto-connection system will handle them automatically. If developers add new types of components that require intersystem connections, they will need to add new auto-connection blocks to the appropriate OPE files for them to connect.

74.5. Adding Variable Bindings to Platform and System Files

Platform files and most system definition files are used by multiple simulation objects. However, the values of some parameters may need to vary for the different simulation objects that use them. This is handled by using variables in the platform and system definition files and specifying the values for these variables in the *.entity* files. VR-Forces uses variable bindings for the parameters that you are most likely to want to change from simulation object to simulation object. If there are other parameters that you want to vary, you can add your own variable bindings.

When you add a variable binding, you must:

1. Change the hard-coded value to a variable in the platform or system definition file.
2. Add the variable to the *.entity* file.
3. Update the Simulation Object Editor so that you can edit the value for the new variable.

The following example shows a variable binding for fuel in a movement system. The variable is specified in the system definition file as follows:

```
(resources
  (fuel
    (resource-type "real-resource")
    (amount  $fuel-amount)
    (full-amount  $fuel-amount)
  )
)
```

The *.entity* file for a simulation object that uses this movement system specifies a value for the variable:

```
<componentSystem systemName="movement"
  platform="@(system-dir)/movement/ground-tracked.sysdef">
  <real paramName="fuel-amount">465</real>
</componentSystem>
```

As you can see, the parameter defined as **fuel-amount** in the *.entity* file is referenced as `$fuel-amount` in the system definition file.

74.5.1. Adding a Variable Binding

The procedure for adding a variable binding is the same for platform files and system definition files.

To add a variable binding to a platform file or system definition file:

1. In the system definition file, substitute a variable name, in the format $*variable-name* for the hard-coded value that you want to replace. For example, the *Ground_Vehicle.ope* file has a value is-organized:

```
(groundParams
  …
  (is-organized True)
  …
)
```

   To make this a variable, replace the value with a variable name, as follows:

```
(groundParams
  …
  (is-organized $entity-is-organized)
  …
)
```

   You can do this by hand in the OPE file .

2. Optionally, specify a default value for the variable. The default value is used if no value is specified for an entity type that uses this OPE file. The syntax for specifying a default value is:

```
(default "value")
```

   In this example, to set the default value to True, do the following:

```
(groundParams
  …
  (is-organized $entity-is-organized (default "True"))
  …
)
```

3. In each *.entity* file that uses this platform file or system definition file, add an element to specify a value for the variable. The attribute must specify the type of the value, the name of the variable, and the value for the variable, for example.

```
<simObject objectType="1:1:1:225:1:1:3:0"
  platform="platforms\Ground_Vehicle.ope">
  <bool paramName="entity-is-organized">True</bool>
...
</simObject>
```

   Although you can add these entries by hand, it is highly recommended that you do so in the Simulation Object Editor by applying platform updates. When you apply platform updates, all changes to platform and system definition files get applied to all simulation objects that use them. For details, please see "Applying New and Updated Variables," on page 67-25.

If you do not add an attribute to an *.entity* file that references the modified platform or system definition file, a default value will be used, which may result in unexpected behavior.

74.5.2. Making Variables Editable in the Simulation Object Editor

When you add new variable bindings, you can make them editable in the Simulation Object Editor (SOE). Variables in platform files are edited in the General, Physical, and Movement tabs. They include variables such as speed, mass, and bounding volume. Variables in system definition files are edited in the Properties dialog boxes for the individual systems. The procedures for adding these two types of variables to the Simulation Object Editor are different.

74.5.3. Adding Platform Variable Bindings to the SOE

Variable bindings for platform files are edited in the General, Physical, and Movement tabs. The layout of these tabs is controlled by *.ui* files. The Simulation Object Editor has a *.ui* file for each platform. The files are in *./data/simulationModelSets/<model_set>/gui/ui*. You can edit *.ui* files in a text editor or in Qt Designer. Qt Designer is included with VR-Forces (*./bin64/designer.exe*).



You do not need a developers license to use Qt Designer.

In "Adding Variable Bindings to Platform and System Files," on page 74-22, we added a new variable binding to the *Ground_Platform.ope* file. The corresponding entry in *ground-vehicle-param.ui* file would be:

```
<item>
 <widget class="QCheckBox" name="entity_is_organized">
   <property name="text">
   <string>Is Organized</string>
   </property>
   </widget>
</item>
```

Notice that the variable name *entity-is-organized* in the OPE file is expressed as `entity_is_organized` in the *.ui* file. You must use underscores in the *.ui* file for the variable name.

If you are adding an attribute to the Simulation Object Editor and editing the .ui file by hand, the easiest approach is to copy an existing item, paste it where you want, and edit it for the new attribute. Many of these attributes have the **display-units** and **source-units** properties. For attributes that are expressed in units, such as meters or kilograms, these properties specify the unit. The unit value gets expressed in the attribute label, for example (m) or (kg). If your new attribute is not expressed in units, leave these fields blank.

UI File Details

Whether you edit a *.ui* file by hand or using Qt Designer, you need to understand how data entry elements are defined. This varies depending on the type of widget used, such as text box, check box, option button, or list.

It is beyond the scope of this manual to explain Qt Designer and GUI layout. However, at the most basic level, the controls in the GUI are specified in item elements. Each item has a widget that has a **class** and an **objectName**. (The XML attribute is **name**. In Qt Designer it is **objectName**.)

- The **objectName** for an item is the platform variable name with dashes changed to underscores. For example the **objectName** for the `short-name` variable is `short_name`.
- If an item has a label, the **objectName** must be `<variable_name>_label`. For example, the **objectName** for the Short Name label is `short_name_label`.

Each data entry widget creates a data entry field that is appropriate for the variable type (integer, real, string). Other elements (systems, vector entry) create more complex controls.

Depending on the type of information displayed, you can assign additional properties to each data entry field. These properties can be:

- element_type. String property that can be assigned a particular reader/writer (variable) type (for example, DtRwReal) or any other internal type that has been registered for element creations. These types are:
  - single_system_selector
  - multi_system_selector
  - sensor_signatures
  - aggregate_composition
  - aggregate_formation_editor
  - push_button
  - scripted-movement-system-selector.
- enable_if_checked. String property that defines a check box element. If the check box is selected, the variable is enabled. If clear, it is disabled.
- disable_if_checked. String property that will define a check box element. If the check box is selected, the variable is disabled. If clear, it is enabled.
- disable_if_multiple_selection. Boolean property. If selected, when the user selects multiple simulation objects, this item cannot be edited.
- hide_by_default. Boolean element. If selected, this item is hidden by default and is visible only if the Advanced button is clicked.
- signal_slot. String element. If defined for a push button, a particular slot is called when that signal is sent. The format is:

  ```
  <signal>;<referencedControl>;<slot>released();bounding_volume;
    onCalculateSupportPointsChanged()
  ```

  The referenced control is the name of an existing control in the *.ui* file.

74.5.4. Adding System Definition File Variables to the SOE

When you add a system to a simulation object in the Simulation Object Editor, you can edit its properties. The Properties dialog box is not configured in a *.ui* file. It is generated based on the metadata section of the system definition file. Suppose that you want to change the rounds per minute parameter in *120mm-gun.sysdef* to a variable binding. You would create the variable binding as described in "Adding Variable Bindings to Platform and System Files," on page 74-22 and you would add details to the metadata as shown in the following example:

```
(weapon-120mm-gun
  ...
  (rounds-per-minute $rounds-fired-per-minute)
  ...
  (meta-data
    (parameter-data-list
      (bool-parameter-data
        (parameter-name  "rounds-fired-per-minute")
        (variable-type  "DtRwInt")
        (display-label  "Rounds Fired Per Minute")
        (display-units  "")
        (source-units  "")
        (default-value  True)
      )
...
```

74.6. State Properties and Parameter Properties

All simulation objects and tactical graphics have properties, such as ordered speed, mass, and force, that are stored in their state repository. You can add the following types of properties to these objects without using the VR-Forces Toolkit:

- State properties. A property that can store state information locally or can be published on the network. You can access the property using Lua scripts or the C++ API and you can change the value.
- Parameter properties. A state property whose value is static and cannot be changed at runtime.

The VR-Forces GUI and simulation components do not know anything about these state properties unless you write code to integrate them.

74.6.1. Adding State Properties and Parameter Properties

You define state properties and parameter properties in the `state-data` section of the OPE files for the simulation object types that you want to have them. The basic syntax for defining a state property is:

```
(type property_name default [publish record_type])
```

where:

- *type* is a data type. Please see Table 74-5 for a list of supported data types.
- *property_name* is the name of the property, such as NumberOfWheels.
- *default* is a default value.
- `publish` is an optional parameter that specifies that the state property should be published.
- *record_type* is an optional parameter to the publish flag. For details, please see Section 4.8, Publishing User-Created State Properties, in *VR-Forces Developers Guide*.

The following example specifies two state properties:

```
(state-data
   (DtRwInt TargetsMissed 0)
   (DtRwInt NumberOfWheels 4 publish)
)
```

Both properties are of type DtRwInt. The TargetsMissed property is initialized with the value 0. It is not published. The NumberOfWheels property has a default value of 4. It is published to the network.

You can bind state properties to variables. This lets you specify different values for different simulation objects. For a complete discussion of binding properties to variables, please see "Adding Variable Bindings to Platform and System Files," on page 74-22. The format for specifying a state property and binding it to a variable is:

```
(type property_name variable [(default default_value) pub-
   lish record_type])
```

where:

- *type* is a data type. Please see Table 74-5 for a list of supported data types.
- *property_name* is the name of the property, such as NumberOfWheels.
- *variable* is a variable name that you use to add the property to platform and system files.
- `(default` *default_value*`)` specifies a default value for a variable. (Optional.)
- `publish` specifies that the state property should be published. (Optional.)
- *record_type* is a parameter to the publish flag. For details, please see Section 4.8, Publishing User-Created State Properties, in *VR-Forces Developers Guide*. (Optional.)

The following example adds a state property with a variable binding:

```
(state-data
  (DtRwInt NumberOfWheels $number-of-wheels (default 4) publish)
)
```

You specify parameter properties in the `parameter-data` section of the OPE file. The syntax is the same.

Table 74-5 identifies the data types supported for state properties. The Data Type column indicates the name of the data type used in the OPE file definitions. (The data types used by C++ are slightly different.) The Description column describes the format of this type. The HLA Representation/Encoding column indicates what type should be used in an HLA 1516 FOM to match the specification in the OPE. For information about HLA and state properties, please see Section 4.8, Publishing User-Created State Properties, in *VR-Forces Developers Guide*.

Table 74-5: Data types supported by state and parameter properties

Data Type	Description	HLA Representation/Encoding
DtRwInt, DtRwInt32	A 32 bit integer.	RPRunsignedInteger32BE, HLAinteger32BE
DtRwInt16	A 16 bit integer.	RPRunsignedInteger16BE, HLAinteger16BE
DtRwInt8	An 8 bit integer.	RPRunsignedInteger8BE, HLAoctet
DtRwReal, DtRwReal64	A 64 bit float.	HLAfloat64BE
DtRwReal32	A 32 bit float.	HLAfloat32BE
DtRwBoolean	A Boolean.	RPRboolean
DtRwString	An ASCII string.	RPRnullTerminatedArray of HLAASCIIchar
DtRwStructure	A fixed structure with multiple named fields.	HLAfixedRecord with matching fields.
DtRwList	A variable sized list of items.	HLAvariableArray
DtRwMap	A variable sized map of items.	HLAvariableArray of HLAfixedRecords, where the first field in each record is the key.

74.6.2. Getting and Setting State Properties Using Lua Scripts

You can set and get the values for state properties using scripted tasks and sets or through C++ code. For details about accessing them using C++, please see *VR-Forces Developers Guide*. This section describes how to access state properties using Lua.

To get the value of a state property from Lua, use the getStateProperty() function of a SimObject. Use getParameterProperty() to retrieve parameter properties.

```
local hits = this:getStateProperty("TargetsHit")
```

To set the value of a state property from Lua, use the setStateProperty() function.

```
this:setStateProperty("TargetsHit", 2)
```

The setStateProperty() function has an optional third argument that specifies that the property should be created if it does not already exist. If not set, this argument defaults to false. When a state property is added this way, VR-Forces must try to determine what the type of the property is, and it might not make the choice you intended, especially for properties with complex types. Therefore, if you are defining a complex property we recommend that you add it to an OPE file. Properties added using this argument cannot be published to the network.

```
this:setStateProperty("SomeNewProperty", 1.2, true)
```

74.6.3. Displaying State Property Values in the Information Dialog Box

You can add user-created state properties to the State Data page in the Information dialog box.

To add state properties to the State Data page:

1. Open *./appData/settings/vrfGui/default_guiConfiguration.gui* in a text editor.
2. Find the DtEntityInformationPage `pageitem` element.
3. Add an `informationitem` element for each state property you want displayed, in the following format:

   ```
   <informationitem
     informationitemname="DtStatePropertyPageItem:MyStateProp:Display
     Name"/>
   ```

   where:

   – *`MyStateProp`* is the name of the state property as specified in the OPE file.

   – *`Display Name`* is the text that you want to display for the property in the Information dialog box.

74.6.4. An Example of Adding State Properties

This section is an example of how you can add state properties, configure them in the Simulation Object Editor, set them with a scripted set, and use them in a scripted task.

For this example, let's assume a scenario in which ground vehicles need to travel from one location to another. There are several routes they can follow. One route can only be traveled by smaller wheeled vehicles. A second can be traveled by large wheeled vehicles, defined as any vehicles with more than four wheels. A third can be traveled by tracked vehicles.

To make this all work, we need to know if a vehicle has wheels, and if so, how many. We will start by adding state properties to hold this information.

Add the State Properties

We want to add two state properties. `hasWheels` indicates that a vehicle has wheels. `numWheels` holds the number of wheels. We will add these properties to *./data/simulationModelSets/EntityLevel/vrfSim/platforms/Ground_Vehicle.ope* as variables so that we can add them to the Simulation Object Editor. `numWheels` has a default value of 4.

```
(state-data
  (DtRwBoolean hasWheels $has-wheels publish)
  (DtRwInt numWheels $number-of-wheels (default "4") publish)
)
```

Add the State Properties to the Simulation Object Editor

You do not have to make state properties settable in the Simulation Object Editor, but if you want to use them for a lot of simulation objects, it is convenient to be able to set them when you add new simulation objects or update existing ones.

Ground vehicle variables are configured in *.data/simulationModelSets/EntityLevel/gui/ui/ground-vehicle-param.ui*. Each variable is configured as an `item` element. We want to add the state properties so that we set `hasWheels` with a check box. If the check box is enabled, the `numWheels` field becomes enabled and we can set that.

If you take a look at the variables available to a ground vehicle in the Simulation Object Editor, you will see that many of them have check boxes and an associated value. You will also see that the Configure button is disabled unless Can Be Embarked On is selected. Using the existing item elements as a guide, we add the following code after Maximum Pivot Speed, so that it is listed after Maximum Pivot Speed:

```
<item>
  <layout class="QHBoxLayout" name="horizontalLayout_66">
    <item>
      <widget class="QCheckBox" name="has_wheels">
        <property name="text">
          <string>Has Wheels</string>
        </property>
        <property name="element_type" stdset="0">
          <string notr="true">DtRwBoolean</string>
        </property>
      </widget>
    </item>
    <item>
      <widget class="QLineEdit" name="number_of_wheels">
        <property name="enable_if_checked" stdset="0">
          <string notr="true">has_wheels</string>
        </property>
        <property name="source-units" stdset="0">
          <string notr="true"></string>
        </property>
        <property name="display-units" stdset="0">
          <string notr="true"></string>
        </property>
      </widget>
    </item>
  </layout>
</item>
```

We set the name for the `layout` element to a unique name (horizontalLayout_66).

The QCheckBox widget has the name `has_wheels`. This is the variable for the `hasWheels` property (`$has-wheels`), with the hyphen replaced with an underscore. (This convention is described in "Making Variables Editable in the Simulation Object Editor," on page 74-24 and "Adding Platform Variable Bindings to the SOE," on page 74-24.) The label for the check box is Has Wheels and it is of type DtRwBoolean, which matches the data type we set in the OPE file.

The QLineEdit widget (the data entry field) has the name `number_of_wheels` to reference the `$number-of-wheels` variable. The **enable_if_checked** property is set to `has_wheels`. This means if `has_wheels` is checked, `number_of_wheels` is enabled. We left the other properties blank since we do not express the number of wheels in any unit of measure.

You can also edit *ground-vehicle-param.ui* in Qt Designer, which is included with VR-Forces (*./bin64/designer.exe*). When you add a widget for a variable, the objectName property should be set to the variable name, for example, `number-of_wheels`.

Update the Platform in the Simulation Object Editor

Now we need to update the Simulation Object Editor to use the updated *ground-vehicle-param.ui* file.

1. Start the Simulation Object Editor.
2. Load *EntityLevel.sms*.
3. Choose **File** → **Regenerate SMS Files**. An information prompt opens.
4. Click OK. The Simulation Object Editor gets updated so that all ground vehicles now have the Has Wheels check box and the number of wheels data entry field (Figure 74-6).

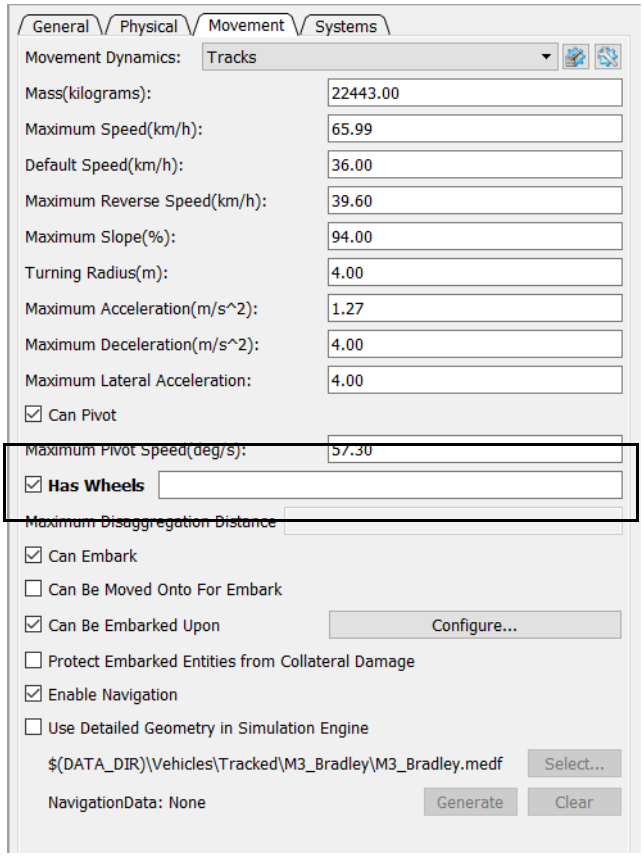


Figure 74-6. Has Wheels check box

5. You can now configure vehicles to have wheels and set the number of wheels.

Add the State Properties to the Information Dialog Box

If you want to see the values for the state properties in the Information dialog box, add the following lines to the DtEntityInformationPage `pageitem` element in *./appData/settings/vrfGui/default_guiConfiguration.gui*:

```
<informationitem
  informationitemname="DtStatePropertyPageItem:hasWheels:Has Wheels"/>
<informationitem
  informationitemname="DtStatePropertyPageItem:numWheels:Number of
  Wheels"/>
```

Figure 74-7 shows the Information dialog box for an entity that is set to have wheels.

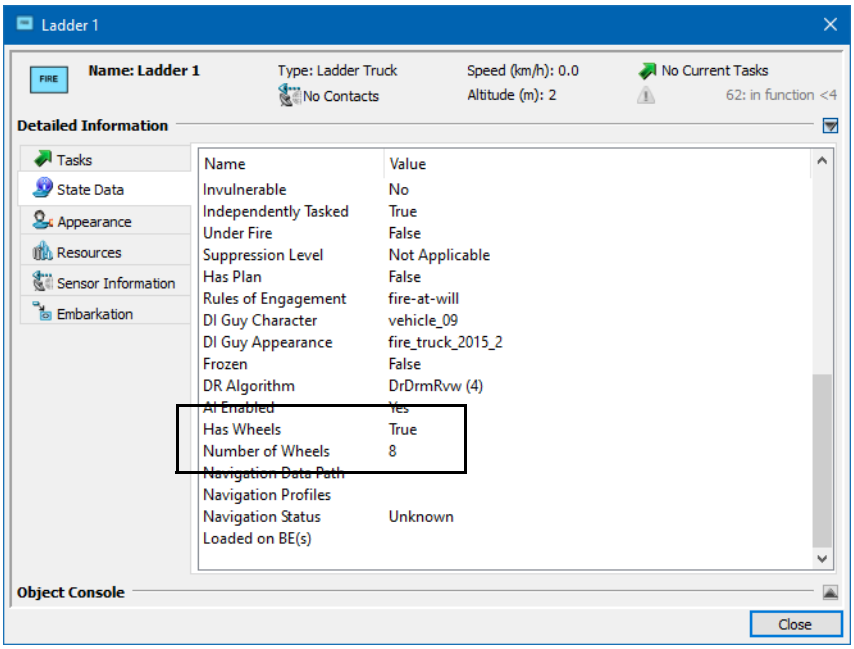


Figure 74-7. Information dialog box with user state properties added

Write a Scripted Set to Set the Number of Wheels

You might not want to configure every ground vehicle in the Simulation Object Editor with its wheel status. Maybe you just want to be able to set this in the GUI. To do that you can write a scripted set data request.

Figure 74-8 shows the setup for the Set Wheels scripted set. It has two parameters that match the state properties we added. It is valid for land entities and is on the Set menu.

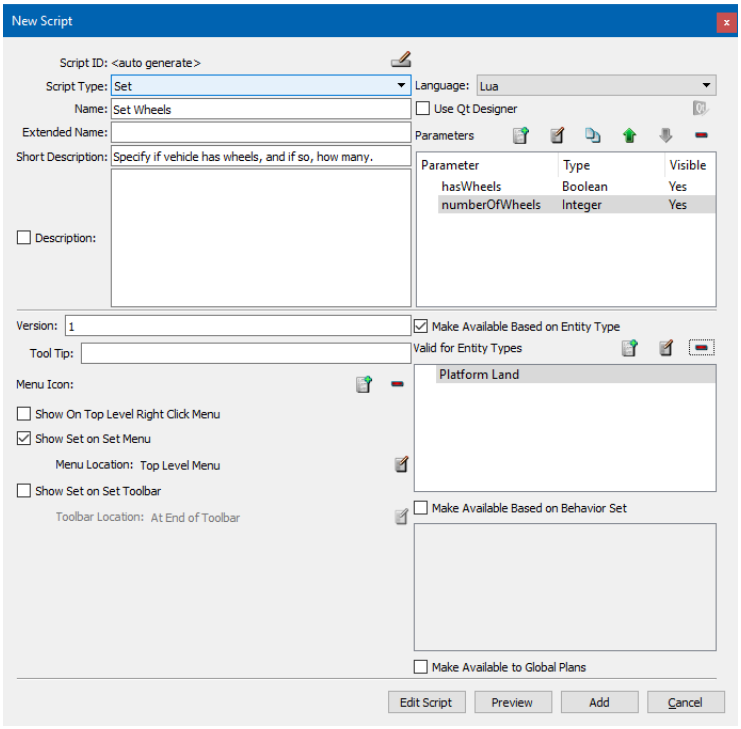


Figure 74-8. Set Wheels meta data

The Lua script for Set Wheels is as follows. It starts out by getting the current values for the two state properties and printing them to the object console. Then it takes in the input from the Set Wheels dialog box and sets the state properties to the new values. Finally, it gets the state properties and prints them to the console to confirm that it worked. We have left out the generic comments and the unused functions and commented the code.

```
require "vrfutil"
function init()
  vrf:setTickWhilePaused(true)
-- get the current values for the state properties
  local haswheels = this:getStateProperty("hasWheels")
  local numwheels = this:getStateProperty("numWheels")
-- if the entity has wheels, print out that it has them and how many
-- this tests that we are getting the state properties correctly
  if haswheels then
    print("This vehicle has wheels.")
    print("Number of wheels is: ", numwheels)
  else
-- if no wheels print this instead
    print("This vehicle does not have wheels.")
  end
  finished = 0
  vrf:setTickPeriod(0.5)
end
function tick()
-- this happens after we complete the Set Wheels dialog box and click OK
-- get the haswheels value from the dbx and set the state property
  haswheels = setParameters.hasWheels
  this:setStateProperty("hasWheels", haswheels)
-- get the number of wheels from the dbx and set the state property
  wheels = setParameters.numberOfWheels
  this:setStateProperty("numWheels", wheels)

-- test it again. get the property values and print out the results.
  newhaswheels = this:getStateProperty("hasWheels")
  newnumwheels = this:getStateProperty("numWheels")

  if newhaswheels then
    print("This vehicle has wheels.")
    print("Number of wheels is: ", newnumwheels)
  else
    print("This vehicle does not have wheels.")
  end
  finished = 1

  if finished == 1 then
    vrf:endSet()
  end
end
```

Write a Task that Uses the State Properties

Now we will write a script that uses the state properties. Remember that the premise of the exercise was based on the need for different vehicles to travel on different routes to reach a destination. In the scenario, we create three routes – Tracked, Heavy, and Light, that lead to the destination (Base). Our task, Move to Base, gets the `hasWheels` and `numWheels` state properties and based on what it finds gives a Move Along Route task to the entity using the appropriate route. There is no input to the task, so the New Scripted Task dialog box just sets up the name of the task and puts it on the menu (Figure 74-9).

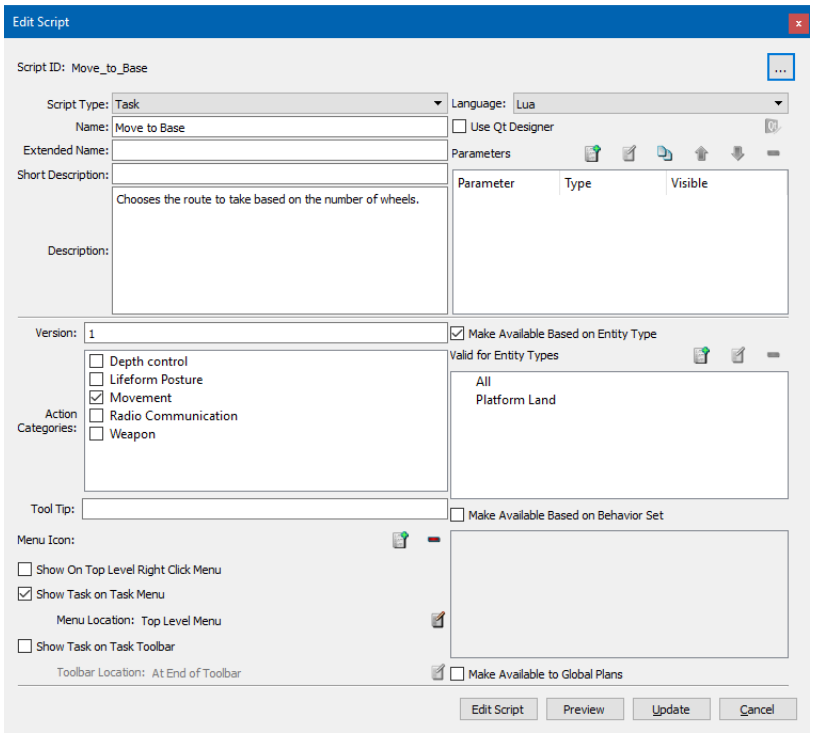


Figure 74-9. Move to Base meta data

Here is the Lua script:

```
require "vrfutil"
function init()
  moveAlongRouteSubtaskID = -1

-- get the properties and print to the console for test purposes
  haswheels = this:getStateProperty("hasWheels")
  numwheels = this:getStateProperty("numWheels")
  if haswheels then
    print("This vehicle has wheels.")
    print("Number of wheels is: ", numwheels)
  else
    print("This vehicle does not have wheels.")
  end
  vrf:setTickPeriod(0.5)
end

function tick()
  if moveAlongRouteSubtaskID == -1 then
-- see if the vehicle has wheels. If yes, give proper move along task
-- based on the number of wheels
    if haswheels then
      if numwheels <= 4 then
        moveAlongRouteSubtaskID = vrf:startSubtask("move-along",
          {route = "Light"} )
      end
      if numwheels > 4 then
        moveAlongRouteSubtaskID = vrf:startSubtask("move-along",
          {route = "Heavy"} )
      end
-- if no wheels, assume tracked and send on tracked route.
    else
      moveAlongRouteSubtaskID = vrf:startSubtask("move-along",
        {route = "Tracked"} )
    end
  end
  if vrf:isSubtaskComplete(moveAlongRouteSubtaskID) then
    vrf:endTask(true)
  end
end
```

In your scenario, create a few ground vehicles. If necessary, specify the number of wheels they have. Give them the Move to Base task and they will pick the correct route to move along.

74.7. Configuring Units

All units have the **parameter-type** `aggregate-object-param`.

In entity-level scenarios, most functionality resides in entities and units have relatively few individual parameters. The principle function of the various elements is to specify controllers and formations, and to specify the **echelon-level** for the different levels of units.

In aggregate-level scenarios, aggregated units can have many systems and parameters. Disaggregated units, like their entity-level counterparts, have limited capabilities. Both types of unit have the **parameter-type** `aggregate-object-param`.

74.8. Configuring Tactical Graphics

The object parameter database includes parameters for tactical graphics. Tactical graphics provided with VR-Forces have the **parameter-type** vrf-base-object-param. You should not change the parameters for the objects provided with VR-Forces. However, if you are using the VR-Forces toolkit, you could create other types of tactical graphics. You would have to include parameters for them in the object parameter database.

74.9. Pathname Interpretation in Simulation Model Set Files

The files that make up a simulation model set, such as *.entity* files and system definition files, usually contain references to other files in the SMS. These supporting files may also have references to other configuration files. VR-Forces supports the file relationships by specifying files using relative paths and by defining directory variables. This allows you to maintain custom simulation model sets outside of the VR-Forces file hierarchy and not have to move them every time you install a new version of VR-Forces.

Examples of directory variables are in the SMS file and OPD file. The following lines from the *EntityLevel.sms* file show a binding (model-set-directory) and its use:

```
(simulation-model-set
  (variable-bindings
    (DtRwString
      (model-set-directory  "EntityLevel")
    )
    (DtRwString
      (opd-dir  "$(model-set-directory)")
    )
  ...
  )
  (opd-file  "$(model-set-directory)\vrfSim.opd")
  (physical-world  "$(model-set-directory)\physicalWorldParams.mtl")
  ...
)
```

Similarly, the *vrfSim.opd* file has bindings, which are then used in *.entity* files, as shown.

From *vrfSim.opd*:

```
(variable-bindings
  (DtRwString
  ...
    (system-dir "$(opd-dir)/vrfSim/systems"
    )
    (DtRwString
      (ammoselect-dir "$(opd-dir)/vrfSim/ammoselect"
    )
    ...
)
```

From *M1A2_Tank.entity* (**system-dir** variable):

```
<componentSystem systemName="damage"
  platform="@(system-dir)/damage/ground-heavy-armor.sysdef"/>
```



75. Configuring Sensors

This chapter explains how to configure sensors and sensor signatures.

Configuring Sensors 75-2
Adding a New Sensor to a Simulation Object 75-2
Editing a Sensor System Definition File 75-4
Specifying a Sensor Signature 75-5
Specifying Sensor Detection Probabilities 75-9
Adding Detection Types (Why Doesn't My Sensor Sense Anything?) 75-11
Adding a New Sensor Type 75-11
Illumination and Time of Day 75-14
Configuring the Effects of Rain on Radar 75-15
Detection Tables 75-16

75.1. Configuring Sensors

VR-Forces includes several sensor systems and you can add them to simulation objects using the Simulation Object Editor. This section provides additional details about how sensors are configured. A complete sensor configuration requires the following:

- One or more `componentSystem` elements in the *.entity* file for sensor systems.
- A list of sensor signature values in the *.entity* file for the sensor types that can detect the entity type.
- A signature propagator entry in the *./data/simulationModel-Sets/<model_set>/physicalWorldParam.mtl* file (one per sensor type).

VR-Forces includes sensors such as radar, infrared, visual, magnetic anomaly detector (MAD), and sonar. The simulation objects configured in the object parameter database are configured with appropriate sensors and sensor signatures. For details about sensor components and how they work, please see API documentation.

75.1.1. Adding a New Sensor to a Simulation Object

It is recommended that you use the Simulation Object Editor to add new sensors to a simulation object. If you want to do it manually, you can edit its *.entity* file.

All sensors provided with VR-Forces are signature-object-sensors. They are distinguished by the domain in which they operate (for example, visual, radar, infrared, MAD, or sonar).

To add a sensor to a simulation object by editing its *.entity* file:

1. Copy an existing `componentSystem` element in the *.entity* file and paste it into the file.
2. Change the `systemName` to be unique, for example, sensor-2.
3. Change the name of the system definition file to point to the sensor you want to add.
4. Change the values for the variables that must be specified for the sensor.

Connecting a Sensor

If you add a sensor that has been configured to use the auto-connection feature (all sensors provided by MAK support auto-connections), its connections are handled automatically. (For details, please see "Configuring System Connections," on page 74-18.)

If you add a new type of sensor, you must connect it to input through its input port and then send the data to its output port for the appropriate controller. The sensor gets input through its `object-types-to-detect` input port. It sends that data out through its `detected-objects` output port. The connections for the visual sensor, in *visual-sensor.sysdef* are:

```
(connect system:object-types-to-detect visual-sensor:object-types-to-
  detect)
(connect visual-sensor:detected-objects system:detected-objects)
(connect system:sensor-offset visual-sensor:sensor-offset)
```

If you were to create an acoustic sensor, its connections would look like this:

```
(connect system:object-types-to-detect acoustic-sensor:object-types-to-
  detect)
(connect acoustic-sensor:detected-objects system:detected-objects)
(connect system:sensor-offset acoustic-sensor:sensor-offset)
```

75.1.2. Editing a Sensor System Definition File

The following example is from the visual sensor system definition file:

```
(visual-sensor
  (component-descriptor-type "signature-sensor-descriptor")
    (component-type  "signature-sensor")
    (min-tick-period  2.000000)
    (min-tick-period-variance  0.100000)
    (tick-period-uses-real-time False)
    (process-state-repository-name  "")
    (process-state-repository-type  "")
    (is-enabled True)
    (detect-only-hostile-forces False)
    (detection-types )
    (detect-destroyed-objects False)
    (sensor-geometry
      (in-range
        (range  $max-range)
      )
    )
    (sensor-domain  "visual")
    (sensor-offset  $sensor-position)
    (sensor-positional-error  0.000000)
    (detection-level-determinator
      (determinator-type  "signature-detection-level-determinator")
      (detection-level-to-set-hostility  3)
      (combat-identification-level-table-file
        "$(detection-dir)\std-visual-detection-table.csv")
    )
  )
)
(controllers
)
(actuators
)
(connections
  (connect system:object-types-to-detect
    visual-sensor:object-types-to-detect)
  (connect visual-sensor:detected-objects system:detected-objects)
  (connect system:sensor-offset visual-sensor:sensor-offset)
)
```

As you can see from viewing the file, most of the parameters that you would want to vary from entity to entity are variables whose values are set in the *.entity* file.

75.1.3. Specifying a Sensor Signature

A sensor signature indicates that a simulation object can be detected by other simulation objects that have a sensor for that signature. The value of the signature represents the maximum distance, at which the entity can be detected. However, this value is affected by modifiers and the calculations made by the propagator. (Technically, the signature does not assume a particular measurement unit. As implemented, distance is measured in kilometers.)

The modifiers specify multipliers for the base value, based on the current state of the object. For example, a simulation object might have a base value of 5.0 for its visual signature, and a modifier of 1.5 when moving. This means the signature of the entity is 7.5 when moving and 5.0 when not moving.

The following is the sensor signature block from a DI with M4:

```
(sensor-signatures
  (visual
    (base-signature  $visual-signature)
    (modifiers
      (file
        (external-file
          (filename  "signatureRules/lifeform-visual-rules.mtl")
        )
      )
      (file
        (external-file
          (filename  "signatureRules/visual-illumination-rules.mtl")
        )
      )
    )
  )
  (infrared
    (base-signature $infrared-signature)
    (modifiers )
  )
  (radar
    (base-signature $radar-signature)
    (modifiers )
  )
)
```

This sensor-signature block shows that a DI M4 can be detected by visual, infrared, and radar sensors. The base-signature values are variables. Their values come from the *.entity* file.

```
<sensorSignatures>
  <real paramName="visual-signature">1.2</real>
  <real paramName="infrared-signature">1.2</real>
  <real paramName="radar-signature">0.25</real>
</sensorSignatures>
```

The visual sensor has two modifiers. The file *lifeform-visual-rules.mtl* is as follows:

```
(signature-modifiers
  (speed-less-than
    (speed 1.0)
    (multiplier 0.5)
  )
)
```

This indicates that if the entity's speed is less than 1 KMH, multiply the base signature by 0.5. A slowly moving entity is less likely to be seen.

For details about the visual illumination rules modifier, please see "Illumination and Time of Day," on page 75-14.

If you do not want a simulation object to be detectable by a particular type of sensor, remove the sensor signature from the *.entity* file. If you want it to be detectable by an additional sensor type, add its sensor signature to the *.entity* file. You can edit the values of sensor signatures in the Simulation Object Editor. You cannot add sensor signatures in the Simulation Object Editor.

VR-Forces has a set of predefined sensor signatures in *./data/simulationModelSets/EntityLevel/extra/sensor-signatures.csv*. These values are used only in the Simulation Object Editor to quickly set standard sensor signatures for entities in the various sensor domains. The standard sensor signatures are also used if you edit the properties of a sensor system in the Simulation Object Editor.

Table 75-1 lists the sensor signatures that VR-Forces uses for general categories of simulation objects in *EntityLevel.sms*. (These categories do not match the tags list in the Simulation Object Editor. They represent a more granular approach to categorizing simulation objects for the purpose of applying a consistent set of sensor signatures.)

Table 75-1: Entity-level sensor signatures (Sheet 1 of 4)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
Military Fixed Wing - stealth technology	2.5	4	6	N/A	N/A	N/A
Military Fixed Wing - Large Jet (Cargo)	7	30	30	N/A	N/A	N/A
Military Fixed Wing - Small Jet (Fighter)	4	6	20	N/A	N/A	N/A
Military Fixed Wing - Large Prop	5	25	15	N/A	N/A	N/A
Military Fixed Wing - Small Prop	4	20	15	N/A	N/A	N/A
Military Fixed Wing - Large UAV (Predator)	5	20	16	N/A	N/A	N/A

Table 75-1: Entity-level sensor signatures (Sheet 2 of 4)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
Military Fixed Wing - Small UAV	2	2	1.5	N/A	N/A	N/A
Military Fixed Wing - Micro UAV (Hand Launch)	1	0.5	0.5	N/A	N/A	N/A
Civilian Fixed Wing - Large (Passenger)	10	40	35	N/A	N/A	N/A
Civilian Fixed Wing - Medium (regional)	7	30	30	N/A	N/A	N/A
Civilian Fixed Wing - Small (Cesna)	5	20	12	N/A	N/A	N/A
Small Rotary Wing	4	20	20	N/A	N/A	0.3
Medium Rotary Wing	5	25	25	N/A	N/A	0.3
Large Rotary Wing	7	30	30	N/A	N/A	0.3
Military/Civilian Large Cargo Ship (tanker/cargo)	35	40	30	10	8	N/A
Military Aircraft Carrier	35	40	30	10	8	N/A
Military Cruiser	25	30	25	8	6	N/A
Military Destroyer	20	28	25	8	6	N/A
Military Frigate	18	25	25	8	6	N/A
Military Large Patrol (Corvette)	18	25	25	6	4	N/A
Military Small Patrol (covered)	10	25	25	3	2	N/A
Military Small (zodiac)	4	6	8	2	1.5	N/A
Civilian Large Commercial Passenger (Cruise)	45	50	50	10	8	N/A
Civilian Medium Commercial	25	40	35	8	5	N/A
Civilian Small Commercial (tug)	15	30	35	4	3	N/A
Civilian Private Medium	15	25	25	4	3	N/A

Table 75-1: Entity-level sensor signatures (Sheet 3 of 4)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
Civilian Private Small (24->30ft day cruiser)	10	25	25	3	2	N/A
Civilian Private Micro (jet ski)	5	9	10	1	2	N/A
Civilian Sailboat (J-24)	15	20	10	2	0.5	N/A
Small Lifeboat (bright colors)	7	12	15	1	1	N/A
Large Lifeboat (bright colors)	10	15	15	1	1	N/A
Military Submarine - Large (Ohio)	2	3	2	1	0.01	N/A
Military Submarine - Medium (LA)	1.5	3	2	0.75	0.01	N/A
Military Submarine - Small	1.5	3	2	0.5	0.25	N/A
Military Submarine - Torpedo (non decoy)	0	0	0	1	2	N/A
Human (no camou-flage)	1.2	0.25	1.2	N/A	N/A	N/A
Human (with camouflage)	0.5	0.25	0.75	N/A	N/A	N/A
Human in Water (with bright vest)	2	2	1.5	N/A	N/A	N/A
Human in Water (no vest)	0.75	2	1.5	N/A	N/A	N/A
Civilian Passenger Car	4	0.5	4	N/A	N/A	N/A
Civilian Truck/Bus	6	1	6	N/A	N/A	N/A
Military Artillery - large	3	0.75	4	N/A	N/A	N/A
Military Artillery - small	1	0.5	2	N/A	N/A	N/A
Military Large vehicle (Logistics)	5	1	5	N/A	N/A	N/A
Military Medium vehicle (Tank)	3	0.5	4	N/A	N/A	N/A

Table 75-1: Entity-level sensor signatures (Sheet 4 of 4)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
Military Small vehicle (Jeep/Technical)	1	1	2	N/A	N/A	N/A
Military Micro vehicle (human sized or smaller)	0.5	0.25	1	N/A	N/A	N/A

75.1.4. Specifying Sensor Detection Probabilities

Sensor systems can have a detection probability table. If a sensor system does not have a detection probability table, it always detects targets (assuming they meet detection criteria such as being in line-of-sight, within the detection range, are not being jammed, and so on) and the combat identification level (CID) always advances as the apparent signature increases over time. By adding a detection probability table to a sensor, you can add variability to a sensor's ability to detect targets and update information about them.



The systems provided with VR-Forces do not have detection probability tables.

A detection probability table associates apparent signatures with detection probabilities. (Apparent signature refers to the radar return strength, if the sensor is a radar, or to the apparent brightness of a target on the sensor in question. For more information about apparent signatures, please see "Detection Tables," on page 75-16.) The detection probability is applied each detection period of the sensor. For radar, the detection period models the sensor sweep, such that the probability is applied individually for each sensor sweep.

The probability of detection affects detection as follows:

- Initial detection. If an object is not on the current detection list, the probability of detection is applied after the apparent signature is calculated. If a random draw against the probability passes, the target passes to the CID detection table calculation to determine the CID level. If the random draw fails, then the object is treated as if it was not detected. It is processed again on the next sensor detection run.
- Continuing detection. If an object is on the detected list, a random draw is done against the detection probability for its current apparent signature. If this passes, the entity remains detected as normal. Its location and CID level are updated if it is appropriate to do so. If this check fails, the entity remains on the detection list, however the data in the contact information is not updated with any new data, including location, CID level, and so on. On the next sensor detection run, this target is processed again as part of the continuing detection section of the sensor.

The following example shows how you could add a sensor detection probability table to the active-radar-sensor system (*./data/simulationModelSets/EntityLevel/vrfSim/systems/sensors/active-radar-sensor.sysdef*). (The system included with VR-Forces does not have a sensor detection probability table.)

```
(active-radar-sensor-system
  (systems )
  (sensors
    (radar-sensor
      (component-descriptor-type "radar-signature-sensor-descriptor")
      (component-type  "radar-signature-sensor")
      ...
      (detection-period  2.000000)
      (detection-probability
        (signature 1.0
          (probability 0.2)
        )
        (signature 1.5
          (probability 0.2)
        )
        (signature 2.0
          (probability 0.2)
        )
        (signature 2.5
          (probability 0.2)
        )
        (signature 3.0
          (probability 0.2)
        )
        (signature 3.5
          (probability 0.2)
        )
        (signature 4.0
          (probability 0.2)
        )
      )
      (detection-period-variance 0.5)
      ...
    )
  )
)
```

75.1.5. Adding Detection Types (Why Doesn't My Sensor Sense Anything?)

Sometimes VR-Forces users add a sensor to a simulation object and then wonder why it does not detect anything. This problem occurs when a simulation object does not have a weapon. In this case, the sensor's `object-types-to-detect` parameter is empty. To have a simulation object that does not have a weapon detect other simulation objects, specify object types to detect in the `detection-types` parameter in the sensor's system definition file, for example:

```
(detection-types
  (entity-type 1 (1 3 -1 -1 -1 -1 -1))
  (entity-type 1 (1 4 -1 -1 -1 -1 -1))
)
```

You can also debug an object's failure to fire on another object by comparing them with the Damage Probabilities utility. For details, please see "Viewing Inter-Object Damage Probability Results," on page 77-14.

75.1.6. Adding a New Sensor Type

You can easily add new sensor types to your simulation objects. Suppose that you want to add an acoustic sensor. You would do the following:

1. Add a block to the signature-propagator-params block in *physicalWorld-Params.mtl*.
2. Add a sensor `componentSystem` element to each entity that can detect targets acoustically.
3. Add a sensor signature to each simulation object that can be detected acoustically.

Configuring a Signature Propagator

The *physicalWorldParams.mtl* file specifies the list of sensors available and the propagator to use for each. The following is a portion of this file:

```
(physical-world-params
  (signature-propagator-params
    (radar
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "standard-propagator")
      (check-line-of-sight True)
    )
    (visual
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "visual-propagator")
      (check-line-of-sight True)
      ...
    )
  )
)
```

If you edit the file to add an acoustic sensor, it might look like the following:

```
(physical-world-params
  (signature-propagator-params
    (visual
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "visual-propagator")
      (check-line-of-sight True)
    )
    (acoustic
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "standard-propagator")
      (check-line-of-sight False)
    )
    ...
  )
)
```

You do not have to be able to see a simulation object to hear it, so you do not need to check line of sight.

Adding a Sensor Component

To add a new sensor component, follow the procedure in "Adding a New Sensor to a Simulation Object," on page 75-2. A new acoustic sensor might look like the following:

```
(acoustic-sensor
  (component-descriptor-type "signature-sensor-descriptor")
    (component-type  "signature-sensor")
    (min-tick-period  2.000000)
    (min-tick-period-variance  0.100000)
    (tick-period-uses-real-time False)
    (process-state-repository-name  "")
    (process-state-repository-type  "")
    (detect-only-hostile-forces False)
    (detection-types )
    (detect-destroyed-objects False)
    (sensor-geometry
      (in-range
        (range  $max-range)
      )
    )
    (sensor-domain  "acoustic")
    (sensor-offset  $sensor-position)
    (sensor-positional-error  0.000000)
    (detection-level-determinator
      (determinator-type  "signature-detection-level-determinator")
      (detection-level-to-set-hostility  3)
      (combat-identification-level-table-file
        "$(detection-dir)\std-acoustic-detection-table.csv")
    )
  )
)
(controllers
)
(actuators
)
(connections
  (connect system:object-types-to-detect
    acoustic-sensor:object-types-to-detect)
  (connect acoustic-sensor:detected-objects system:detected-objects)
)
```


You can call the sensor whatever you want, but the name must not start with a number.

Adding a Sensor Signature

If you add a new sensor, you need to add a signature for it to each simulation object that can be detected by the sensor. If you add an acoustic sensor, a sensor signature might look like the following:

```
(sensor-signatures
  (visual
    (base-signature  4.000000)
    (modifiers )
  )
  (acoustic
    (base-signature  2.000000)
    (modifiers )
  )
  ...
)
```

You can probably see simulation objects at a greater distance than you can hear them, so the acoustic sensor has a lower base-signature value than the visual sensor.

75.1.7. Illumination and Time of Day

Visual sensors take into account the time of day. They do this through an illumination value that modifies the visual sensor signature. The modification value is configured in *./data/simulationModelSets/<model_set>/vrfSim/signatureRules/visual-illumination-rules.mtl*:

```
(signature-modifiers
  (environmental
    (modifier-coefficients 1.0 0.1)
  )
)
```

"Specifying a Sensor Signature," on page 75-5 shows how this file can be referenced in a sensor signature. Given this file, the modifier coefficients are applied to the illumination value, as follows:

```
Multiplier = 1.0 * illumination + 0.1
```

The illumination value is derived from the time of day based on parameters in *Global-Environment.ope*:

```
...
(actuators
  (global-environment-actuator
    ...
    (sunrise-time  21600.000000)
    (sunset-time  64800.000000)
    (daylight-illumination  1.000000)
    (night-illumination  0.100000)
    (day-night-transition-time  7200.000000)
  )
)
```

where:

- sunrise-time is the time at which the daylight model changes from night to day, in seconds past midnight.
- sunset-time is the time at which the daylight model changes from day to night, in seconds past midnight.
- daylight-illumination is the percent illumination during the full daylight hours.
- night-illumination is the percent illumination during the full night hours.
- day-night-transition-time is the period of time, centered on sunrise and sunset times, where the illumination value transitions from daylight to night, or night to daylight. During this time period, the illumination level is a linear progression between the night and daylight illumination levels.

75.1.8. Configuring the Effects of Rain on Radar

VR-Forces can take into account the effect of rain on radar sensors. It does so based on the amount of rainfall per hour, the range from the sensor to the target, and the rain-to-attenuation-coefficient parameter in the radar block of the *physicalWorldParams.mtl* file in an SMS:

```
(radar
  (propagator-descriptor-type "radar-propagator-descriptor")
  (propagator-type "radar-propagator")
  (check-line-of-sight True)
  (default-feature-los-query "MAK_LOF_OBSTACLE")
  (high-altitude-feature-los-query "NONE")
  (rain-to-attenuation-coefficient 0.05)
)
```

The higher the coefficient value, the more that rain reduces the effective range of the radar. Rain is measured in millimeters per hour, where the maximum rain setting in VR-Forces is equal to 20 mm/h.

75.2. Detection Tables

The detection tables determine the level of knowledge that a simulation object has of detected simulation objects. The detection level is used in the Detect Entity condition in plans.

Detection tables are comma-delimited files that are most easily read in a spreadsheet. As illustrated in Figure 75-1, each column represents a detection interval, in seconds, from 0 through 160. Each row represents an apparent signature from one through 4. (An apparent signature is a simulation object's absolute signature after being modified by factors such as terrain and distance. For details, please see API documentation.)

	A	B	C	D	E	F	G	H	I	J	K
1	CID Table-radar	Version 1.0		Detection	Interval	(seconds)	->				
2			0	1	5	10	20	40	80	160	
3	Apparent	1	1	1	2	2	2	3	3	3	
4	Signature	1.5	1	1	2	2	3	3	3	3	
5	I	2	1	1	2	2	3	3	3	4	
6	V	2.5	1	2	2	3	3	3	4	4	
7		3	1	2	3	3	3	4	4	4	
8		4	2	3	3	3	4	4	4	4	
9											

Figure 75-1. Detection table displayed in a spreadsheet

The detection level is determined by finding the intersection of the apparent signature and the detection interval. For example, if a simulation object with an apparent signature of 3 has been detected for 40 seconds, its detection certainty level is 4 (full knowledge).

The apparent signature is the ratio of the sensor signature given for the simulation object (in kilometers) to the actual range. So if the apparent signature = 1, the detection interval shows the increase in combat ID level over time when the target is at the range given by the sensor signature. If the apparent signature is 2, the detection interval shows the detection progression at 1/2 that range, and so on.

The numbers in the table represent the certainty levels as follows:

- 1 – Detected.
- 2 – Classified.
- 3 – Identified.
- 4 – Full Knowledge.

For details about CID detection levels, please see "Detecting Targets," on page 9-16.



76. Configuring Weapon Systems

This chapter describes configuration files for weapon systems in entity-level scenarios.

Introduction to Weapon Systems 76-2
The Weapon System System Definition Template File 76-3
The Weapon System System Definition File 76-5
Configuring Direct Fire Weapons 76-8
Sensing Targets 76-9
Configuring Ammunition 76-10
Hit Probability Tables 76-12
Damage Probability Tables 76-13
Configuring Missile Systems 76-14
Missile Launcher 76-15
Missile Projectile 76-16
Laser Guided Missile Launcher 76-16
Laser Guided Missile Projectile 76-17
Configuring Indirect Fire Weapons 76-17
Configuring Fixed Weapons 76-19
Configuring Bomb Bay Weapon Systems 76-20
Configuring Weapon Systems in the SOE 76-20
Specifying Target Types 76-20
Configuring Hit Probability 76-21
Configuring the Ammunition to Use 76-21
Using the Target Configuration Assistant 76-22
Configuring the Target Priorities 76-24
Configuring the Target Region 76-25
Configuring Missile Launcher Systems in the SOE 76-25
Configuring Explosive Devices 76-26

76.1. Introduction to Weapon Systems

Simulation objects in entity-level scenarios have many weapon systems. They all fall into one of the following types:

- Ballistic direct fire, such as rifles and the main gun on a tank.
- Self-propelled or guided munitions, such as missiles and rockets.
- Indirect fire, such as artillery, naval guns, mortars, and hand grenades.
- Fixed munition, such as mines and IEDs.
- Bomb bay. A weapon system for dropping bombs.

You can create and edit weapon systems by editing configuration files in a text editor or in the Simulation Object Editor. The preferred method is to use the SOE. It provides a point-and-click interface, handles all syntax issues, and includes error checking to ensure that all components required by a system are present.

Weapon systems are defined by a system definition template (*.template_sysdef*) file that includes component definitions (controllers, actuators, and sensors) and a system definition file (*.sysdef*) that contains system-specific parameter values and targeting information, such as ammunition, hit probabilities, and target priorities. Typically a weapon system template includes at least one controller that controls aiming and firing and an actuator that models the physical characteristics of the weapon. Some weapons create munitions when they fire, so the engagement process involves an additional simulation object with its own movement system, weapon (warhead) system, or both.

All objects with weapon systems can engage using a task such as Fire At Target, Fire For Effect, Throw Grenade, and so on.

Simulation objects often engage targets autonomously. Objects that engage autonomously require a sensor to detect targets. Weapon systems are configured to engage certain target types, and this information is passed to the sensor so that the appropriate enemy objects can be detected.

Automatic engagements are controlled by a target selection controller, which is a component separate from any one weapon system. This controller is specified in the platform (*.ope*) file. Weapons pass information to and from the target selection controller through a weapon interface. This is a process state repository used to keep track of targets for the weapon and the number of rounds to fire at that target. No port is used, no connections are necessary; it is created for you. The target selection controller keeps a list of all weapon systems' weapon-interfaces, one for each system.

Weapon systems generate the fire interactions that are published on the network for an engagement. Detonation actions may be generated by the weapon or by the munition it launches.

Once there is a detonation, damage is determined by the target. Damage may be configured in one of two ways: by specifying the power of the munition and the damage caused to a target by a detonation of a given power, or by specifying damage probabilities for each munition type against each target type. For more information, please see "Calculating Damage from Munitions," on page 77-2.



Throughout this chapter, we include snippets from system definition files and other configuration files, but usually without explaining what the various parameters mean. If you want to know more about the parameters, view the referenced systems in the Simulation Object Editor. It has tooltips for most parameters.

76.1.1. The Weapon System System Definition Template File

System definition template files (*.template_sysdef*) contain the sensors, controllers, and actuators that are common to a category of weapon or damage system. They contain variable bindings for attributes that are common to the system type and which are given specific values in the individual weapon system and damage system files. They also contain metadata that is used in the SOE.

The following code is a highly edited version of *handheld-launchers.template.sysdef*. It illustrates how components, resources, and meta-data are configured in a template file. Notice that the meta-data block includes parameter data for the Gun Location and Gun Part Number parameters. These parameters are editable on a per-entity basis when a handheld launcher system is added to an entity (Figure 76-1).

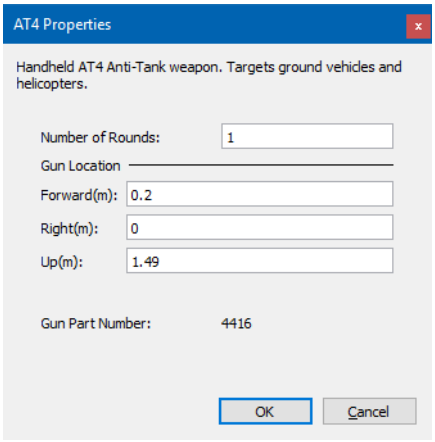


Figure 76-1. Properties for AT4 handheld weapon

The number of rounds parameter is a variable binding in the template file and is given a default value in the system files that use this template, for example, *handheld-AT4.sysdef* (discussed in "The Weapon System System Definition File," on page 76-5).

```
(weapon-handheld-AT4
  (controllers
    (human-acquire
      ...
      (target-acquire-criteria $target-acquire-criteria)
    )
    (main-gun-control
      ...
```

```
      (load-ammo-time  $load-ammo-time (default 5.00000))
      (unload-ammo-time  $unload-ammo-time)
      (disabled-by-suppression True)
      (targeting-control-table $targeting-control-table)
    )
    (aim-at-point
      ...
    )
    (aim-weapon
      ...
    )
  )
  (actuators
    (weapon-state
      ...
    )
    (main-gun
      ...
      (art-part-list
        (primary-gun
          (art-part-type $gun-art-part-type)
          (parent-part-type -1)
          (attach-point $attach-point)
          ...
          )
        )
      )
      (muzzle-speed  $muzzle-speed (default 250.0000))
      (hit-probability-table $hit-probability-table)
      (range-name  $range-name)
    )
  )
  (connections
    ...
  )
  (resources
    (M84mm
      (resource-type "integer-resource")
      (amount $num-rounds)
      (full-amount $num-rounds)
    )
  )
  (meta-data
    (system-name  "Handheld Launcher")
    (system-description  "Template for system will allow the Simulation
        Object to be given the ability to add a handled
        launcher to a human.")
    (system-categories "weapon")
    (allowed-state-repository-types "human-param")
    (parameter-data-list
      (vector-parameter-data
        ...
        (display-label  "Gun Location")
        (default-value  0.200000 0.000000 -1.490000)
        ...
      )
      (art-part-parameter-data
        ...
        (display-label  "Gun Part Number")
        ...
```

```
        )
      )
      (meta-data-entry-list )
    )
  )
```

76.1.2. The Weapon System System Definition File

System definition files provide weapon-specific values for the variable bindings in the template file. They reference the template file that they use so that VR-Forces can use the correct components and make the necessary connections for the weapon system. Weapon-specific system definition files also include a targeting control table and a hit probabilities table.



This section explains the layout of the weapon-specific system definition file for the benefit of advanced VR-Forces users. However, it is recommended that you edit weapon systems using the Simulation Object Editor.

The targeting control table lists each target type that this weapon can fire on and the ammunition types that it can use for that target. If it can use more than one ammunition type, it lists them from the highest to lowest priority. (For more information about targeting control tables, please see "Sensing Targets," on page 76-9.)

The hit probability table specifies the probability that the weapon can hit a target as a range list or a coefficient list. For more information about hit probability tables, please see "Hit Probability Tables," on page 76-12.



By default the metadata in a template file is merged with the metadata in the weapon-specific system file for use in the SOE. If you do not want a system file to merge its metadata with the template file, add the **merge-with-sub-systems** variable to the metadata and set it to False, for example:

```
(meta-data
  (system-name "Light Armor damage model")
  ...
  (meta-data-entry-list )
  (ui-filename "")
  (merge-with-sub-systems False)
)
```

The following example is an edited version of *handheld-AT4.sysdef*, which references *handheld-launchers.template.sysdef*. Notice how it provides values for the **load-ammo-time** and **unload-ammo-time** variables.

```
(weapon-handheld-AT4
  (systems
    (handheld-launcher
      (variable-bindings
```

```
(DtRwReal
  (load-ammo-time  5.000000)
)
(DtRwReal
  (unload-ammo-time  0.000000)
)
(DtTargetAcquireCriteria
  (target-acquire-criteria
    (target
      (entity-type  1 1 -1 1 -1 -1 -1)
      (time-to-acquire  1.500000)
      (check-maximum-speed-to-acquire True)
      (max-speed-of-target  10.000000)
      (check-elevation-angle-to-acquire False)
      (max-elevation-angle-to-acquire  1.570796)
      (check-elevation-to-acquire False)
      (max-elevation-to-acquire  500.000000)
    )
    ...
  )
)
(DtTargetingControlTable
  (targeting-control-table
    (ammunition-list
      (M84mm
        (amount $num-rounds)
      )
    )
    (entity-target-priorities
      (priority-entry
        (target-type 1 1 -1 1 -1 -1 -1)
        (priorities "M84mm")
      )
      (priority-entry
        (target-type 1 1 -1 -1 -1 -1 -1)
        (priorities "M84mm")
      )
      ...
    )
    (max-targets  1)
    (target-region
      (and
        (in-range
          (range  2100.000000)
        )
        (not
          (in-range
            (range  30.000000)
          )
        )
      )
    )
    (fire-with-spot-report-only False)
  )
)
(DtRwReal
  (muzzle-speed  285.000000)
)
(DtRangeProbabilityTable
  (hit-probability-table
```

```
                (entity-range
                  (entity-type  1 1 -1 -1 -1 -1 -1)
                  (range-determinant
                    (range-list
                      (range 100.000000
                        (probability 0.90000)
                      )
                      (range 200.000000
                        (probability 0.5000)
                      )
                      (range 500.000000
                        (probability 0.30000)
                      )
                      (range 750.000000
                        (probability 0.10000)
                      )
                    )
                  )
                )
                ...
              )
            )
            (DtRwString
              (range-name  "Handheld AT4")
            )
          )
          (system-definition
            (filename  "$(system-dir)\weapons\templates\handheld-
      launchers.template_sysdef")
          )
          (is-enabled True)
          (part-of-roll-up False)
          (use-parent-system-resource-manager True)
        )
      )
      (connections
        (connect system:all handheld-launcher:all)
        (connect handheld-launcher:all system:all)
      )
      (meta-data
        (system-name  "AT4")
        (system-description  "Handheld AT4 Anti-Tank weapon. Targets ground
              vehicles and helicopters.")
        (allowed-state-repository-types "human-param")
        (system-categories "weapon" "aimable-weapon")
        (parameter-data-list
          (int-parameter-data
            (parameter-name  "num-rounds")
            (variable-type  "DtRwInt")
            (display-label  "Number of Rounds")
            (display-units  "")
            (source-units  "")
            (default-value  1)
          )
        )
        (meta-data-entry-list )
      )
    )
```

76.2. Configuring Direct Fire Weapons

Entities that have direct fire weapons can fire in response to a Fire at Target task and they can detect targets through their sensors and automatically fire at the targets. Entities have to be able to:

- Know what targets they can fire at.
- Detect the targets.
- Determine the ammunition to use.
- Acquire the target and fire at it.

If you want to create a completely new type of weapon system, you must write code for the sensors, controllers, and actuators that it requires. However, if you need a system that is similar to an existing system, you can often create the new system by copying and modifying the configuration of the existing system. Although you can do so by directly editing configuration files, the recommended way to add and edit weapon systems is to use the Simulation Object Editor. To do so, you must understand how to:

- Configure a weapon system.
- Specify the simulation object types that can use the system.
- Specify the simulation object types that can be damaged by the system.
- Specify the amount of damage a simulation object receives from a munition.

The files that manage these relationships are:

- The weapon system template. The template configures the components common to similar types of weapons, such as ballistic guns or missile launchers.
- The weapon system. Defines a specific system, such as a 30mm gun, the simulation object types that it can target, ammunition types it can use, hit probabilities, and the simulation object types that can use it.
- The damage system. Specifies the types of munitions that a simulation object is vulnerable to and the damage file to use to determine the probability of damage.

A direct fire weapon does the following:

1. Acquires the target.
2. Selects the ammunition for the target using the ammo select file.
3. Checks that it has that resource.
4. Determines if it hits target.
5. Sends a fire interaction.
6. After a delay based on distance and muzzle speed, it sends a detonation interaction.

76.2.1. Sensing Targets

Simulation objects that have direct fire weapons detect targets through their sensors and automatically fire at the targets. Sensor systems can specify what types of targets to detect (using the **detection-types** parameter). The weapon systems can also tell the sensors what they want to target.

Each weapon system has an associated weapon interface. The **targeting-control** parameter takes a list of desired target types and creates a target selection criteria object, which is then set in the weapon interface. The following code shows the targeting control for the *120mm-gun.sysdef* file.

```
(DtTargetingControlTable
  (targeting-control-table
    (ammunition-list
      (M829A1-AP-120mm
        (amount $num-ap-rounds)
      )
      (M830-HEAT-120mm
        (amount $num-heat-rounds)
      )
    )
    (entity-target-priorities
      (priority-entry
        (target-type 1 1 -1 1 -1 -1 -1)
        (priorities "M829A1-AP-120mm" "M830-HEAT-120mm")
      )
      (priority-entry
        (target-type 1 1 -1 2 -1 -1 -1)
        (priorities "M829A1-AP-120mm" "M830-HEAT-120mm")
      )
      ...
    )
    (max-targets  1)
    (target-region )
    (fire-with-spot-report-only False)
  )
)
```

The platform file for the entity uses its target selection controller to iterate through each weapon interface to create a list of object types to detect, which it outputs on its `object-types-to-detect` port.

Each sensor on the entity connects to this port to know what types of objects to detect (Figure 76-2).

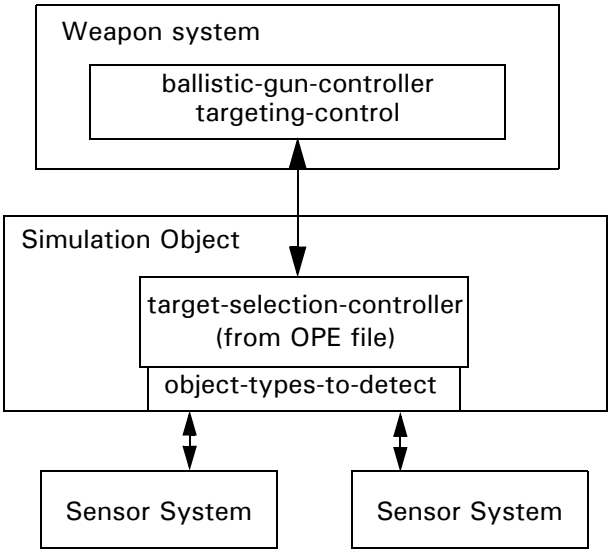


Figure 76-2. Direct fire sensor input

The simulation object gathers the detected objects from all the sensors and stores it in the entity's state repository. (A state repository is a read-write repository of information that is available to all entity components.) The target selection controller selects a target for the weapon.

76.2.2. Configuring Ammunition

The targeting control table ammunition-list block specifies the ammunition types that a weapon system can use. Each priority entry in the table specifies a target type (using the best match method) and a space-separated prioritized list of ammunition that should be used for that type of target (highest priority listed first). (The best match method is described in "The Best Match Method," on page 74-7. For an example of a targeting control table, please see "Sensing Targets," on page 76-9.)

The ammunition types that you can include in a weapon system's ammunition-list are specified in *./data/simulationModelSets/<sms>/ammunition-Params.mtl*. The Simulation Object Editor references this file when it displays the lists of ammunition that you can add to a weapon system. Each ammunition type entry in *ammunitionParams.mtl* specifies a set of parameters that define its characteristics and how it is represented in the Simulation Object Editor, as follows:

- munition-type – 7 digit object type enumeration. This is the object type that is used in the fire and detonation interactions created, and should NOT contain -1s.
- tracer-type – 7 digit object type enumeration. This parameter is optional. Use it if you want a weapon to support tracer rounds.
- warhead – VR-Link warhead enumeration value to use in fire and detonation interactions for this munition.
- guidance-mode. How the munition is guided, if applicable. Options are:
  - 0 - None.
  - 1 - Laser.
  - 2 - Optical Wire.
  - 3 - Radar.
  - 4 - Thermal.
- gui-label. The string used to identify the munition in the Simulation Object Editor.
- targets. Object type enumerations for the simulation objects that the weapon can target.
- tags. The tags used for filtering the munition in the Simulation Object Editor object lists.

The following code shows two entries in *ammunitionParams.mtl*, a missile and a ballistic munition:

```
(spiral-missile
  (munition-type 2 2 222 1 8 0 0)
  (warhead  0)
  (guidance-mode  0)
  (gui-label  "AT-6 Spiral Missile")
  (targets
    (target-type 1 1 -1 -1 -1 -1 -1)
    (target-type 1 3 -1 11 1 -1 -1)
  )
  (tags "3D Model" "Military" "Munition")
)
(AT-30mm
  (munition-type 2 2 222 2 2 0 0)
  (tracer-type 2 2 222 2 2 0 0)
  (warhead  0)
  (guidance-mode  0)
  (gui-label  "AT-30mm")
  (targets
    (target-type 1 1 -1 -1 -1 -1 -1)
  )
  (tags "mm gun")
)
```

76.2.3. Hit Probability Tables

A weapon system definition file includes a hit probability table. This table is used by the system's actuator to determine whether or not it hits a target that it fires upon. A hit probability table has one or more `entity-range` entries. The name of the `entity-range` entry is not used, and can be something descriptive. An `entity-range` entry has the form:

- **entity-type** – 7 digit entity type enumeration, which may use -1s as wildcards.
- **range-determinant** – a range determinant can be either a list of probability-range entries with the name "range-list", or a probability-coefficient list with the name "coefficients".
- **range-list** – a list of probability-range entries. A probability-range entry has a name, "range", a range value (in meters) and a list of probability entries. The entry applies to targets in the range from the last entry to the current one (or from 0.0 in the case of the first entry). Each probability entry has a name that the table user uses to find the probability value it is interested in and the value itself (a real number between 0.0 and 1.0.) The ballistic gun looks for the probability entry with the name "probability".



- For ranges that are greater than the last specified range, the probability is zero.
- The range for missiles is the range from the missile to the target, not from the entity that fired the missile.

- **coefficients** – a coefficient probability list is a list of coefficient probability entries. Each entry has a probability name and the coefficients of a standard polynomial that can be used with a given range to determine the specified probability. The order of the polynomial is determined by the number of coefficients provided. For example, an entry of:

  ```
  (probability -5.00000E-008 -300000E-012 1.0000)
  ```

  indicates that the formula to use for a given range *r* would be:

  $(-5.00000E-008 * r^2) + (-300000E-012 * r) + 1.0$

  Values calculated will be constrained to be between 0.0 and 1.0. Zero values are factored out, for example:

  ```
  (probability -5.00000E-008 0.0000 1.0000)
  ```

  indicates that the formula to use for a given range *r* would be:

  $(- 5.00000E-008 * r^2) + (0.0 * r) + 1.0$

The following is an example of the hit probability table for the handheld-AK47 system:

```
(DtRangeProbabilityTable
  (hit-probability-table
    (entity-range
      (entity-type  3 -1 -1 -1 -1 -1 -1)
      (range-determinant
        (range-list
          (range 10.000000
            (probability 1.00000)
          )
          (range 200.000000
            (probability 0.9000)
          )
          (range 550.000000
            (probability 0.70000)
          )
          (range 3600.000000
            (probability 0.01000)
          )
        )
      )
    )
  )
)
```

76.2.4. Damage Probability Tables

The damage actuator uses damage probability tables specified by its component descriptor to determine direct and indirect fire damage. Damage tables are in the damage systems in *./data/simulationModel-Sets/<sms>/vrfSim/systems/damage*. A damage probability table has a list of surface entries, which provide a sub-table for each surface upon which a round might detonate. The name of the surface entry is the surface name used in the damage model.

For the damage-by-power damage system, VR-Forces has damage probability tables for each power type. For more information, please see "Configuring the Damage File for Damage-By-Power," on page 77-8.

For the damage-by-ammo-type damage system, VR-Forces has damage probability tables for direct fire and collateral damage. A munition type does not need to have both types of probability table if one of the types is not appropriate. For example, an M16 does not have a collateral damage file. Conversely, an artillery shell does not have a direct fire probability table. For more information, please see "Configuring Damage-by-Ammo-Type," on page D-22.

76.3. Configuring Missile Systems

Missiles are launched by an object's missile launcher system, which creates a missile entity. The missile flies itself and detonates based on the use of a hit probability table or on the type of missile.

If a missile launcher system definition specifies a hit probability table, when the missile is created VR-Forces checks to see if the object type of the target is in the table. If it is, a random number is drawn and compared to the range-based probability. If it is determined that the missile should hit the target, the missile begins its normal fly out. If it collides with the terrain or another entity before reaching the target, it detonates and does not hit the target. However, if the missile does not collide with anything and hits or passes the target, it automatically counts as a hit. The missile detonates on the entity. The SA-21 missile launcher and Patriot missile launcher use hit probability tables.

If a missile launcher system definition has a hit probability table, but the target object type is not listed, or if the system does not specify a hit probability table, detonation is determined based on the type of missile, as follows:

- Direct flight. Missiles fly directly to a target. Accuracy is controlled by their maneuverability parameters.
- Laser guided. Missiles follow a laser designator to the target. If the laser is blocked or disrupted, the missile goes to the last known location of the target.
- Route following. Some missiles, such as cruise missiles, can follow a route to a target and detonate either at the target or at a proximity to the target.

The range for missiles is the range from the detonation location to the target, not from the entity that fired the missile.

76.3.1. Missile Launcher

The missile launcher for direct flight missiles works as follows:

1. The target selection controller chooses the best target for the weapon system and passes it to the weapon through the weapon's weapon-interface.
2. The weapon system's controller determines the right resource for that target type.
3. The missile resource is returned and the system loads the missile to fire.
4. The entity tries to acquire the target. It sends the desired rotation angle and elevation to acquire the target to the rotating mount actuator and elevation mount actuator.
5. Actuators move according to their slew rate speeds and send how far they went in a tick back to the weapon launcher so it can recalculate the remaining amount. It also publishes the articulated part numbers along with amount of rotation/elevation to the network.
6. When the feedback reaches the max angle off boresight, it can fire on the target. The missile object is created.
7. Firing state is sent to the launcher actuator.
8. The launcher actuator sends a fire interaction.

Parameters in the missile launcher's system affect whether or not it fires. For example, The Maverick Missile Launcher system includes the following parameters on the Attributes tab in the Simulation Object Editor:

- Load Ammo Time
- Unload Ammo Time
- Max Vehicle Speed to Fire
- Max Azimuth Angle Off Boresight
- Max Elevation Angle Off Boresight.

76.3.2. Missile Projectile

When the missile is created, it flies as follows:

1. Using the missile's movement parameters, the missile controller (track-entity if the target is an entity, or target-point if the target is a location) determines the values to be sent to the missile-maneuver actuator.
2. The missile-maneuver actuator calculates the new position, velocity, acceleration, and orientation of the missile.
3. The missile-collision actuator then checks to see if the missile is going to collide with the terrain or an entity. If so it sends out a detonation interaction.

Missile parameters affect how it detonates. The following code is from *./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/missile-warhead.sysdef*:

```
(actuators
  (warhead
    ...
    (detonate-on-ground-impact True)
    (detonate-on-water-impact True)
    (detonate-on-entity-impact True)
    (allow-direct-hit True)
    ...
  )
)
```

76.3.3. Laser Guided Missile Launcher

The missile launcher for laser guided missiles works as follows:

1. The target selection controller chooses the best target for the weapon system and passes it to the weapon through the weapon's weapon-interface.
2. The weapon system's launcher controller tells the lasing controller the targets.
3. The lasing controller looks for an actively lased target that matches the lasing code. If there are no lased targets, it checks to see if it can autonomously lase. If it can, it starts to lase the target.
4. Once the launcher controller has a lased target, it tells the movement system to rotate to the target.
5. The launcher controller checks for when the entity has rotated to line up with target, within the fire cone defined for it. It creates the missile object and fire interaction.

76.3.4. Laser Guided Missile Projectile

Once it is created, a laser-guided missile flies as follows:

1. Using the missile's movement parameters, the missile controller determines the values to be sent to the missile-maneuver actuator.
2. The missile-maneuver actuator calculates the new position, velocity, acceleration, and orientation of the missile.
3. The missile-collision actuator then checks if the missile is going to collide with the terrain or an entity. If so, it sends out a detonation interaction.

76.4. Configuring Indirect Fire Weapons

Indirect fire weapon systems use the indirect-fire-weapon-controller. Indirect fire weapons fire in response to an indirect fire task, such as Fire-for-Effect or Throw Grenade. They do not fire on their own, like many direct fire weapons do.

When a simulation object receives an indirect fire task, its indirect-fire-controller queues the task and checks for availability of an indirect fire weapon. It continues to check until a weapon becomes available. Then it sends the fire task to the weapon through the indirect weapon interface. If a task requires multiple weapons, the indirect-fire-controller continues this process until the task is complete. (The indirect-fire-controller and indirect weapon interface are analogous to the target-select-controller and weapon interface described for direct fire weapons.)

When an indirect-fire-weapon-controller gets a task, it reads the fire mission from the indirect weapon interface. The indirect fire weapon controller tries to align the turret and load the munition. When the weapon is ready to fire, it passes the information to its indirect-fire-actuator.

The munition is specified as part of the task. The indirect-fire-actuator creates the desired munition and sends out the fire interaction.

Figure 76-3 illustrates this process.

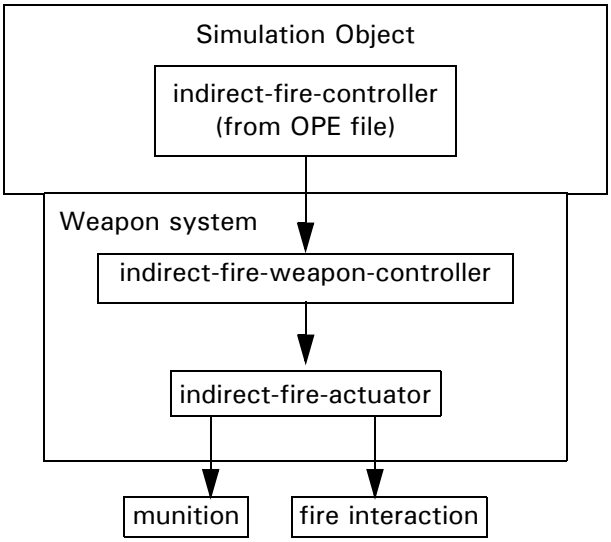


Figure 76-3. Indirect fire



Do not confuse the indirect-fire-controller and the indirect-fire-weapon-controller. The indirect-fire-controller is on the simulation object that has the weapon system (it is in the platform OPE file). The indirect-fire-weapon-controller is in the indirect fire weapon system. The controllers are illustrated in Figure 76-3.

Examples of indirect fire weapons include the M284-155mm cannon (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/M284-155mm-cannon.sysdef*), the M252-81mm mortar (*M252-81mm-mortar.sysdef*), and MK45 naval gun (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/MK45NavalGun.sysdef*). Also, although hand grenades would not normally be thought of as indirect fire weapons, the hand grenade tasks call the fire for effect tasks. Therefore, they use the same controllers and actuators as indirect fire weapons.

The munitions used by the indirect fire weapon systems are configured with explosive power in *detonationParams.mtl*. Therefore simulation objects that are susceptible to explosive power can be damaged by them. Their damage system definition files will point to the appropriate damage table files to assess damage from these munitions.

As with other weapon systems, the system definition files for indirect fire weapons have parameters that control when they can be used and how effective they are. You can set the values for these parameters in the Simulation Object Editor on the Attributes and Firing tabs. The Targets tab just has entries for ammunition, since there is no need for hit probability and target selection for indirect weapons.

76.5. Configuring Fixed Weapons

Fixed munitions, such as mines and IEDs can be triggered in the following ways:

- Immediately, by a detonate message.
- By proximity of a valid target. The weapon needs a sensor system or a proximity controller that can detect targets.
- Timed detonation, from when the system is armed.

Fixed munition have a detonation controller. The detonation controller processes set detonation messages to determine how it is to be triggered and whether or not it is armed. It tells the warhead-detonation-actuator when to arm the weapon. For immediate and timed detonation, it tells the warhead actuator to detonate the device.

For proximity detonations, the detonation controller takes input from the external sensor or the proximity sensor. The warhead-actuator checks for armed and trigger messages from the detonation controller. It also checks for collisions with the terrain and simulation objects. If a collision occurs or it is triggered by the detonation controller, it sends a detonation interaction over the network.

VR-Forces has two types of fixed weapons, naval mines and IEDs.

IEDs are simulation objects that have an explosive device system (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/explosive-device.sysdef*). These objects need a sensor to detect targets. The Roadside IED object has a visual sensor. You can specify the ammunition, targets to detect, and the type of fuse on the Targets tab for the system in the Simulation Object Editor.

Naval mines use the *./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/naval-mine-explosive-device.sysdef* system definition file. It has a proximity sensor built into the system definition, so it does not need an external sensor to detect targets. You can specify the ammunition, targets to detect, and the type of fuse on the Targets tab for the system in the Simulation Object Editor.

You can turn an entity into an IED by adding the explosive device system in the Simulation Object Editor. You can configure cultural features as IEDs by adding the Other Explosive Device system.

76.6. Configuring Bomb Bay Weapon Systems

The bomb-bay weapon system releases bombs in response to Release Bomb tasks. Bombs can drop directly onto targets or be laser guided. Once a bomb is released, it uses the *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement/smart-bomb-dynamics.sysdef* movement system to control movement. It has a warhead actuator that is similar to the warhead actuator in missiles. Its ability to navigate toward and hit its target is controlled by parameters that you can set in the Simulation Object Editor. The Simulation Object Editor also lets you specify the ammunition type for bomb bay systems. For more information about bombs, please see "Release Bomb Tasks," on page 27-20.

76.7. Configuring Weapon Systems in the SOE

Like other systems, weapon systems have a General tab for identification information and one or more tabs for physical parameters. Targeting is configured on the Targets tab. When you configure a direct fire weapon system, you must specify:

- The probability that the weapon will hit its target.
- The ammunition it should use.
- Target priorities.
- Target region.

The following sections explain each aspect of target configuration. However, sometimes it can be confusing to understand how to configure the minimum requirements for a target type. The SOE has two features that help you with this process — the Target Configuration Assistant and the Configuration Status window. For details, please see "Using the Target Configuration Assistant," on page 76-22.

76.7.1. Specifying Target Types

Weapon systems have one or more target types that they can fire on. You can add, edit, and remove target types. For each target type you must specify Hit Data. You might also need to specify an ammunition type that is valid for the target.

To add a target type:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) next to the Target Type label. The Add New Target Type to Hit Table dialog box opens.
4. Specify an object enumeration for this target type.

5. Select a creation option, as follows:
   - Copy Currently Selected. Use the same type of hit table as the currently selected target.
   - Create Empty Range Table. Create a range list table for configuring how this weapon can fire on the new target.
   - Create Empty Coefficient Table. Create a coefficient table for configuring how this weapon can fire on the new target.
6. Click OK. The target is added to the Target Type list and a Hit Data table is created based on the selected option.
7. Check the Configuration Status window to see if there are any configuration issues as a result of adding this target type.

76.7.2. Configuring Hit Probability

You can configure hit probability using a range list or a coefficient list. You can configure a range list by editing the table or by editing the graph directly. You configure a coefficient list by editing the table.

To edit a hit probability table:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Target Type window, select the Target Type that you want to configure.
4. Edit the probabilities as described in "Editing Probability Tables," on page 68-38.

76.7.3. Configuring the Ammunition to Use

You can specify the types of ammunition that a weapon system can use. You also specify the default amount. Some weapon systems can use more than one type of ammunition.

To add a type of ammunition for a weapon system:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) above the Ammunition window. The Choose Ammunition dialog box opens. It lists ammunition types that may be appropriate for this system. Ammunition types that are not supported for this weapon system have an alert icon next to their name. You can add them to the ammunition list, but they will not have any effect on targets unless you also configure them to work with the weapon system.
4. Select the ammunition type that you want to add.
5. Click OK. The ammunition is added to the list in the window.

Adding a New Ammunition Type from the Targets Page

You can add new ammunition types on the Munitions tab. However, you can also start the process from the Choose Ammunition dialog box that you open from the Targets tab.

To add a new ammunition type:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) above the Ammunition window. The Choose Ammunition dialog box opens.
4. Click Add New from Existing. The Create New Model dialog box opens.
5. Type a name for the new ammunition type.
6. On the Entity Type line, click Edit. The Change Enumeration dialog box opens.
7. Specify the object type for the new ammunition by selecting options from the lists or directly entering the object type.
8. Click OK. The new ammunition type is added to the list.
9. Select the Munitions tab.
10. Select the new ammunition type that you added.
11. Specify the parameters for this munition.

Removing an Ammunition Type

To remove an ammunition type from a weapon system:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Ammunition window, select the row for the ammunition type that you want to remove.
4. Click the Remove button (−) .

76.7.4. Using the Target Configuration Assistant

You can specify a target type, hit probabilities, and ammunition independently, as described in previous section, or you can specify them within one dialog box. This ensures that you have made the minimum configuration necessary for the weapon system to successfully fire on a target type.

To use the Target Configuration Assistant

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Hit Probabilities section, click the Target Configuration Assistant button. The Target Configuration Assistant opens (Figure 76-4).

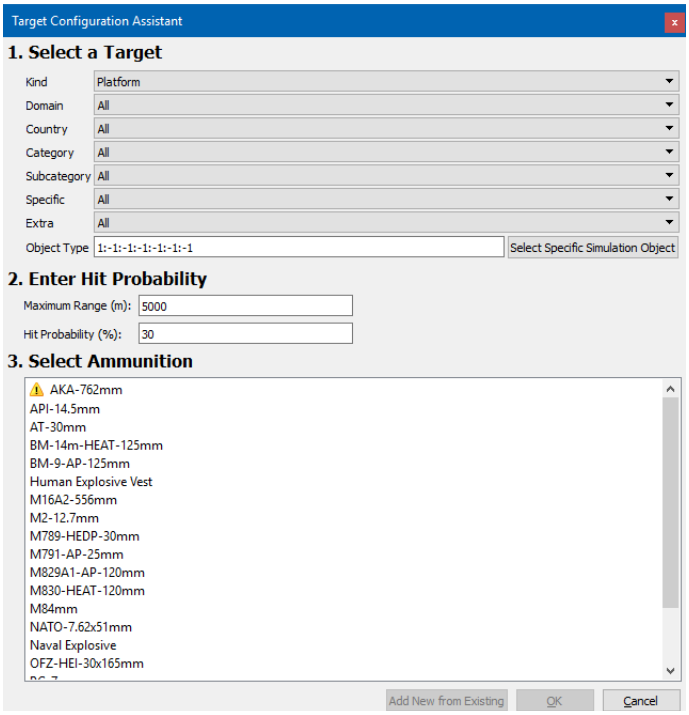


Figure 76-4. Target Configuration Assistant

4. Select a target type. You can build an object type by selecting options for the lists, you can directly type in an object type, or you can select a specific simulation object as the target.
5. In the Maximum Range box, type the maximum distance at which the weapon can hit the target. For range lists, the SOE will automatically divide up the range for the hit probability table into three segments.
6. In the Hit Probability box, type the probability of hitting the target at the maximum range.
7. In the ammunition list, select the ammunition to use for this target type. The window lists all possible ammunition types. If an ammunition type is not configured to fire on the target type, it is preceded by an alert button (as illustrated in Figure 76-4). You can still add this ammunition type, but you would have to configure it to fire on the target type or it will not work.
8. Click OK. The target type, hit probability, and ammunition are added to the page.
9. Check the Configuration Status window to see if there are any issues for this target type.
10. Optionally, specify weapon target priorities and the target region.

76.7.5. Configuring the Target Priorities

Weapons can only fire at objects that have been configured as possible targets. Within that set of possible targets, you can assign priorities. When the target controller identifies more than one possible target, it chooses its target based on the priorities.

You can specify the types of objects that a weapon can target and the priorities of those target types. For each target type you can specify the preferred ammunition to use.

To add a new target priority:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) above the Weapon Target Priorities window. The Add New Target Priority dialog box opens.
4. Specify the object type for the target you want to add.
5. Click OK. The target is added to the bottom of the list.

Changing a Target Priority's Object Type

You can change the object type of a target priority.

To change the object type of a target priority

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Weapon Target Priorities window, select the target priority you want to change.
4. Click the Edit button (✎) above the Weapon Target Priorities window. The Select Target Type dialog box opens.
5. Change the object type.
6. Click OK.

Changing the Priority of a Weapon Target

Weapon targets are listed in the Weapon Target Priorities window in order of priority, from highest to lowest. You can change the priority of a weapon target by changing the order in which it is listed.

To change the priority of a weapon target:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Weapon Target Priorities window, select the target whose priority you want to change.
4. Click the Up Arrow above the window to raise the target's priority. Click the Down Arrow to lower its priority.

76.7.6. Configuring the Target Region

You can configure a target region for a weapon system. (Many weapon systems do not have a value for this parameter.) If you specify a target region, the weapon system will not fire at a target unless it is within or without the region. For example, a visual sensor might detect a target at 500 meters, but you do not want the weapon to fire until the target is within 100 meters.

The target region can be specified as a range, a frustum, or both. You can specify a minimum range, maximum range, or both.

To specify the target region for a weapon system:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. To specify the target region as a range:
   a. Select the Minimum Range check box.
   b. Type a value for the minimum range.
   c. Select the Maximum Range check box.
   d. Type a value for the maximum range.
4. To specify the target region as a frustum:
   a. Select the Use Frustum check box..
   b. Specify the maximum and minimum values for the azimuth and elevation.
   c. Optionally, select the Outside Frustum check box, to specify the that the target region is outside of the frustum.

76.7.7. Configuring Missile Launcher Systems in the SOE

VR-Forces has several types of missile launcher systems. The types of missiles that they use determines the parameters that must be set on the Targets tab.

- Laser guided missile systems configure ammunition (as described in "Configuring the Ammunition to Use," on page 76-21) and weapon target priorities (as described in "Configuring the Target Priorities," on page 76-24).
- Cruise missile launchers configure ammunition.
- Other missile launcher systems can use hit probability tables or they can calculate hits based on the type of missile, as explained in "Configuring Missile Systems," on page 76-14. When you configure a missile system you specify how to calculate hit probability by choosing one of the following options:
  – Simulate Missile Guidance. Do not use a hit probability table. Calculate based on missile type.
  – Use Probability Table. Configure the launcher like a direct fire weapon system. Specify target type, hit data, ammunition, weapon target priorities, and target region.

76.7.8. Configuring Explosive Devices

Explosive devices, such as IEDs and naval mines, need to specify the ammunition type, detonations parameters, and objects to detect.

To configure an explosive device weapon system:

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Detonation Proximity Distance box, specify the distance to use for proximity detonation.
4. Select any of the check boxes that apply for types of impact this device will support.
5. Specify the ammunition, as described in "Configuring the Ammunition to Use," on page 76-21.
6. Specify the object types that this device can detect for proximity detonation.



77. Configuring Damage Systems

This chapter describes configuration files for damage systems in entity-level scenarios.

Introduction to Damage Systems 77-2
Calculating Damage from Munitions 77-2
Configuring Damage-by-Power 77-3
Configuring the Power of a Detonation 77-4
Configuring Detonation Power in the SOE 77-6
The Damage File for Damage-By-Power 77-8
Adding a New Power Type 77-10
Configuring Damage in the SOE 77-11
Configuring Damage Power Entries 77-11
Configuring Impact Surfaces 77-12
Configuring the Angle of Incidence 77-13
Specifying Kill Data 77-13
Viewing Inter-Object Damage Probability Results 77-14
Configuring Damage for Dynamic Terrain 77-16
Standard Terrain Damage 77-16
High Fidelity Terrain Damage 77-17
Configuring Detonation Results 77-18

77.1. Introduction to Damage Systems

Like weapon systems, damage systems are based on high-level template_system files and lower level system definition files that reference the templates. The template files specify the damage actuators and metadata. The system definition files specify the damage to entity surfaces by the different power types that can damage the entity.

You can create and edit damage systems by editing configuration files in a text editor or in the Simulation Object Editor. The preferred method is to use the SOE. It provides a point-and-click interface, handles all syntax issues, and includes error checking to ensure that all components required by a system are present.

Like other systems, damage systems have a General tab and Attributes tab for identification parameters and other default parameters. The damage done to the entity is configured on the Damage tab.

77.1.1. Calculating Damage from Munitions

VR-Forces has the following ways to calculate damage:

- Damage-by-power. Each munition is configured for its power in the explosive, kinetic, fragmentation, and armor-piercing domains as appropriate. (You can add new domains without writing code.) Damage systems (*.sysdef* files) contain entries for the types of munition power that can affect simulation objects that use the system. They configure the amount of damage that occurs when different levels of power are applied. The damage-by-power model is used for all entity-level simulation objects in VR-Forces 4.5 and later and is the preferred method for calculating damage.
- Damage-by-ammo-type. Each munition that can affect a simulation object is configured individually in the damage system used by that object and refers to a damage file that is specific to that type of munition. This model was used in VR-Forces 4.4.2 and earlier. It is supported in VR-Forces 4.5 and later. For details about damage-by-ammo-type, please see "Configuring Damage-by-Ammo-Type," on page D-22.

The damage-by-power method of configuration makes it much easier to add new munitions than damage-by-ammo-type. However, if you have specific requirements for the effects of a munition on different simulation objects, you can use the damage-by-ammo-type method in conjunction with damage-by-power. If VR-Forces finds an `ammo-entry` block in the damage system file, it uses that entry and its related damage file for that ammo type instead of the damage-by-power damage file that would otherwise be used.

77.2. Configuring Damage-by-Power

In the damage-by-power method of calculating damage, each munition has a `detonation-power-entry` block in *./data/simulationModelSets/base/detonationParams.mtl*. The following is the entry for the 7.62 mm bullet:

```
(detonation-power-entry
  (munition
    ;; 7.62 mm
    (munition-type  2 8 222 2 2 -1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (kinetic
      (direct 2.24) ;; Kilojoules
    )
  )
)
```

The `munition` block identifies the munition type using the DIS/RPR FOM object enumeration. (The default configuration does not use the **warhead** or **guidance-mode** parameters.)

The `power-list` block is a list of domains and the configuration for a determinant that produces a power value based on an input range. In this example, the 7.62 mm shell has power in the kinetic domain.

The default configurations support the following domains:

- Explosive. Munitions that affect simulation objects within some range of the detonation. Explosive munitions represent power in kiloPascals (kPa) of incident pressure.
- Kinetic. Munitions that directly strike a simulation object. Kinetic munitions represent power in kilojoules (kJ) of energy.
- Armor-piercing. Munitions that strike an armored object and penetrate it. Armor piercing munitions represent power in millimeters of penetration.
- Fragmentation. Explosives that cause damage by fragmenting, such as grenades.



You can add other domains without writing code. For details, please see "Adding a New Power Type," on page 77-10.

The power values used in the default configuration are based on publicly available information about the power of the various types of munitions and the values used try to be consistent in their relative power.

77.2.1. Configuring the Power of a Detonation

You can configure detonation power by editing *detonationParams.mtl* or in the Simulation Object Editor. This section explains how to edit the file directly and what the parameters mean. For details about editing detonation power in the SOE, please see "Configuring Detonation Power in the SOE," on page 77-6. To determine the power of a detonation at a specific location a power determinant must be defined. The determinant maps a range value to a power value. Range can have different possible meanings, depending on how the **range-from** parameter is defined for the determinant. If **range-from** is undefined, `impact-point` is assumed unless the determinant does not support this setting.

- If **range-from** is `impact-point`, the determinant determines power based on the range from the impact point of the detonation to some other location in the vicinity. This is used for the explosive and fragmentation domains, but could be used for any power that has some area of effect (for example, perhaps shrapnel or incendiary damage). This is equivalent to collateral damage as defined in the damage-by-ammo-type damage system.

  Example (from *detonationParams.mtl*):

```
(detonation-power-entry
  (munition
     ;; 85 mm HEAT
    (munition-type  2 2 222 2 8 1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (range-list
        (range 3.5
          (power 70.0)
        )
        (range 4.6
          (power 40.0)
        )
        (range 5.7
          (power 27.5)
        )
        ...
        (range 35.0
          (power 2.4)
        )
      )
    )
    (kinetic
      (range-from "shooter-to-impact")
      (range-coefficients -2.6600E-004   0.0000   5319.0)
    )
    (armor-piercing
      (range-from "shooter-to-impact")
      (range-coefficients -9.0000E-006   0.0000   180.0)
    )
  )
)
```

- If **range-from** is `shooter-to-impact` (as in the previous example), the determinant determines power based on the range from the entity that fired the round to the point of impact. Any points outside of the point of impact are not affected by the detonation. This is used for the kinetic and armor-piercing domains, but could be used for any power that is directed at a particular target and has no area of effect. This is equivalent to direct damage as defined in the damage-by-ammo-type damage system.
- If **range-from** is `unused`, the determinant determines power without using the input range. This is used by the direct determinant, which ignores the **range-from** setting completely. (For more information, please see "Determinant Types," on page 77-5.)
- If **range-from** is `shooter-and-impact-point`, the determinant determines power based on both the range from the shooter and from the impact point. This option is not implemented by VR-Forces.

Determinant Types

VR-Forces supports the following types of determinant.

- The **range-coefficients** determinant calculates the power based on a series of coefficients. Configuring coefficients works exactly the same way as the hit probability coefficients, except that the output value can be any number 0 or greater, not just a probability for 0.0 to 1.0. For example:

```
(kinetic
  (range-from "shooter-to-impact")
  (range-coefficients -1.3000E-003   0.0000   26005.0)
)
```

- The **range-list** determinant determines the power based on a step function. This is configured exactly the same way as the hit probability **range-list**, except that the output value can be any number 0 or greater, not just a probability for 0.0 to 1.0. For example:

```
(explosive
  (range-from "impact-point")
  (range-list
    (range 4.5
      (power 70.0)
    )
    (range 6.0
      (power 40.0)
    )
    ...
    (range 47.0
      (power 2.4)
    )
  )
)
```

- The **direct** determinant always produces the same power at the point of impact, regardless of the range from the shooter. It produces no power at any range from the point of impact. As a result, the **range-from** setting is ignored, because the range input is unused. For example:

```
(kinetic
  (direct 272.873) ;; Kilojoules
)
```

- The **equivalent-tnt-mass** determinant allows you to define an amount of TNT, in kilograms, that produces an equivalent explosion. VR-Forces calculates power at a specified range in kPa. This determinant is used only for the explosive damage category. The following example shows its use for M107 HE munition:

```
(detonation-power-entry
  (munition
    ;; M107 HE (155 mm)
    (munition-type  2 9 225 2 14 2 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (equivalent-tnt-mass 5.74)
    )
  )
)
```

77.2.2. Configuring Detonation Power in the SOE

You configure the detonation power of each munition type on its Detonation tab. When you add or edit the detonation power for a munition, VR-Forces creates a copy of the *detonationParams.mtl* in the SMS. It is merged with the version in *base.sms* when a scenario is loaded.

To configure the detonation power of a munition:

1. Select the Munitions tab of the Object List.
2. Select the munition that you want to configure.
3. Select the Detonation tab.
4. In the Power List, select the power type that you want to configure.
5. If the munition uses a range-list determinant or a coefficient-list determinant, edit the list by editing the table or by directly editing the graph, as described in "Editing Systems in the Simulation Object Editor," on page 68-36.

   If the munition uses the direct determinant, the range is ignored. Specify the direct power in the Direct text box.

   If the munition uses the equivalent TNT mass determinant, specify the explosive power in the Equivalent TNT text box.

Adding a Damage Power Type and Determinant

You can add and remove power types from the power list. When you add a power, you specify a determinant (range list, range coefficient, direct, equivalent TNT mass).

To add a power and determinant:

1. Select the Munitions tab of the Object List.
2. Select the munition that you want to configure.
3. Select the Detonation tab.
4. On the Power List line, click the Add button ( ). The Power and Determinant dialog box opens.
5. In the Categories window, select a power type. The Determinants window lists the options for this type of power.
6. In the Determinants window, select the determinant you want to use.
7. Click OK. The power is added to the list.

Changing the Determinant for a Power Type

You can change the determinant for a power type. If you change the determinant the values for the old determinant are removed and you must specify the correct data for the new determinant type.

To change the determinant for a detonation power type:

1. Select the Munitions tab of the Object List.
2. Select the munition that you want to configure.
3. Select the Detonation tab.
4. On the Power List line, click the Edit button ( ). The Choose Determinant dialog box opens.
5. Select the determinant you want to use.
6. Click OK. The graph and data on the tab are removed.
7. Specify the data for the new determinant.

77.2.3. The Damage File for Damage-By-Power

Entity damage is configured through the system definition files and the damage files they reference. A system definition file configures power in the `damage-by-munition-power` block.

The system definition files provided with VR-Forces have a `damage-model` block. However none of the files provided use this block. It is available for customers who want to configure munitions using the damage-by-ammo-type method. It also enables backwards compatibility with old SMSs. For information about using the damage-model block, please see "Configuring Damage-by-Ammo-Type," on page D-22.

Configuring Damage-by-Power in a System Definition File

A system definition file can define any number of `damage-by-power-entry` items, each for a different power-type. This is the damage domain of the power. To add a new type of power, add new settings in *detonationParams.mtl* and in the system definition files for simulation objects that need to be damaged by this effect.

Configuring the Damage File for Damage-By-Power

Damage system files configure damage based on the side of the bounding volume being affected. The damage actuator delivered with VR-Forces uses surfaces of front, rear, left-side, right-side, bottom, and top (other components could model the entity's geometry differently, and therefore use different surface strings). A surface entry has a list of angle-of-incidence entries. An **angle-of-incidence** entry has an angle (in radians).

The `determinant` block specifies the damage. The determinant can be a `step-function` or a list of coefficients. Each determinant can specify probabilities for catastrophic-kill, mobility-kill, firepower-kill or some combination of them.

The *ground-heavy-armor.sysdef* file has a parameter called **round-down** that indicates what probabilities should be used when the input power value is between two steps. For power we want to round down to the next lower power (and lesser probability). Therefore **round-down** is set to 1 (true) for all damage files. If this parameter is not specified, the step function rounds up.

The following code, which uses the `step-function` determinant, is from *./data/simulationModelSets/EntityLevel/vrfSim/systems/damage/ground-heavy-armor.sysdef*:

```
(damage-by-munition-power
  (damage-by-power-entry
    (key "explosive")
    (value
      (front
        (angle-of-incidence
          (angle  1.570800)
          (determinant
            (step-function (round-down 1)
              (power 7.5
                (catastrophic-kill  0.1)
              )
              (power 12.5
                (catastrophic-kill  0.5)
              )
              (power 20.0
                (catastrophic-kill  0.6)
              )
              (power 45.0
                (catastrophic-kill  0.75)
              )
              (power 70.0
                (catastrophic-kill  0.99)
              )
            )
          )
        )
      )
      ...
    )
  )
)
```

The following code, which uses the `coefficients` determinant, is from *./data/simulationModelSets/EntityLevel/vrfSim/systems/damage/missile-default.sysdef*:

```
(string-DtDamageProbabilityTable-map
  (damage-by-munition-power
    (damage-by-power-entry
      (key "explosive")
      (value
        (front
          (angle-of-incidence
            (angle  1.5708)
            (determinant
              (coefficients
                (catastrophic-kill 0.02 -0.4)
                (firepower-kill 0.022 -0.44)
                (mobility-kill 0.025 -0.5)
              )
            )
          )
        )
    ...
)
```

77.2.4. Adding a New Power Type

To add a new type of power, you add it to the relevant munitions in *detonation-Params.mtl* and then add it to the appropriate damage system. For example, suppose you want to model the effects of the heat generated by a munition. You could add thermal power. For the sake of this example, suppose that an 85mm HEAT round has thermal power. To add thermal power, you would have to do the following:

1. Add the new power to the `detonation-power-entry`, as follows:

```
(detonation-power-entry
  (munition
    ;; 85 mm HEAT
    (munition-type  2 2 222 2 8 1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (range-list
        (range 3.5
          (power 70.0)
        )
        ...
      )
    )
    (kinetic
      (range-from "shooter-to-impact")
      (range-coefficients -2.6600E-004   0.0000   5319.0)
    )
    (armor-piercing
      (range-from "shooter-to-impact")
      (range-coefficients -9.0000E-006   0.0000   180.0)
    )
    (thermal
      (range-from "impact-point")
      (range-list
        (range 3.5
          (power 70.0)
        )
        ...
      )
    )
  )
)
```

You would have to determine the unit of heat that you wanted to use to measure its effect and come up with realistic power levels at different distances.

2. In the damage systems for the simulation objects that can be affected by thermal power, add a `damage-by-power` entry. Suppose that you want lifeforms to be damaged by thermal power. You would update the *lifeform-default.sysdef* file as follows:

```
(damage-by-power-entry
  (key "thermal")
  (value
    (front-side
      ...
    )
    (left-side
      ...
    )
    (right-side
      ...
    )
    ...
  )
  ...
)
```

77.3. Configuring Damage in the SOE

The Simulation Object Editor provides an easy GUI for specifying damage for each of the power types that apply to an entity. You can configure damage for each surface on an entity and for angles of incidence.



Most of the damage systems provided with VR-Forces do not configure damage for different angles of incidence.

77.3.1. Configuring Damage Power Entries

The Damage Power Entries window lists the types of damage power that VR-Forces supports. A damage system does not necessarily support all types of damage power. If a damage power type is not supported, it is labeled [Unused]. It does not have entries in the Kill Data table. You can disable use of a damage power type and enable unused damage power types.

➤ To disable a damage power type, click the Remove button ().

➤ To enable an unused damage power type, add an entry to its Kill Data table.

77.3.2. Configuring Impact Surfaces

A damage system can vary the amount of damage sustained based on the surface that is struck or can use the same kill data for all surfaces.

To configure the same damage for all surfaces of an entity:

1. In the Damage Power Entries list, select the damage power for which you want to configure impact surfaces.
2. In the Impact Surface options list, select All Surfaces Identical (Figure 77-1).

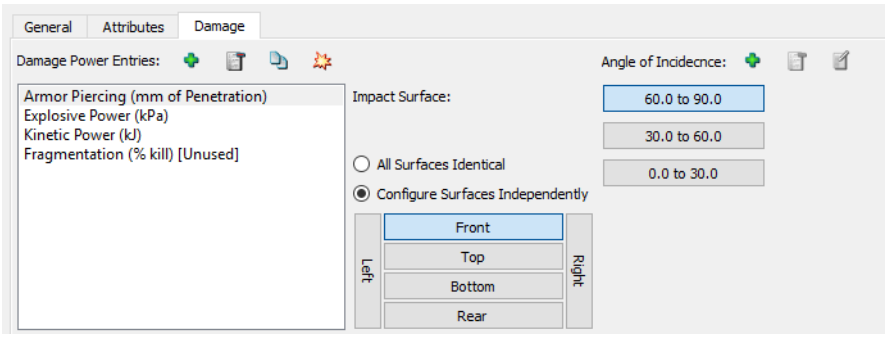


Figure 77-1. Damage power entries and impact configuration

3. Optionally, specify the angles of incidence to use, as described in "Configuring the Angle of Incidence," on page 77-13.
4. Specify kill data for each angle of incidence, as described in "Specifying Kill Data," on page 77-13.

If you configure damage differently for each surface, you must specify the angle of incidence and kill data for each surface.

To specify damage for each surface:

1. In the Damage Power Entries list, select the damage power for which you want to configure impact surfaces.
2. In the Impact Surface options list, select Configure Surfaces Independently (Figure 77-1).
3. For each surface that you want to configure:
   a. Click the box that represents the surface you want to configure.
   b. Optionally, specify the angles of incidence to use for this surface, as described in "Configuring the Angle of Incidence," on page 77-13.
   c. Specify kill data for each angle of incidence, as described in "Specifying Kill Data," on page 77-13.

77.3.3. Configuring the Angle of Incidence

You can specify different damage results based on the angle at which a munition strikes an entity. If a damage system treats all surfaces identically, you configure one set of angles of incidence. If it treats different surfaces differently, you can configure angles of incidence for each surface. The default for most damage systems is that there is no difference based on the angle. You can specify multiple angles as ranges. The total range of all angles must be 0 degrees to 90 degrees.

To add an angle of incidence:

1. In the Damage Power Entries list, select the damage power for which you want to configure angles of incidence.
2. If you are configuring each impact surface independently, click the surface for which you want to configure the angle of incidence.
3. Click the Add button (+) next to the Angle of Incidence label (Figure 77-1). The New Angle dialog box opens.
4. Type an angle, in degrees.
5. Click OK. The new angle is added to the list.

Editing an Angle of Incidence

To edit an angle of incidence:

1. In the Damage Power Entries list, select the damage power for which you want to configure angles of incidence.
2. If you are configuring each impact surface independently, click the surface for which you want to configure the angle of incidence.
3. Click the angle that you want to edit.
4. Click the Edit button (✎) next to the Angle of Incidence label. The Edit Angle dialog box opens.
5. Type the new angle.
6. Click OK.

77.3.4. Specifying Kill Data

For each of the damage power types used by a damage system, you can specify kill data. Kill data can be specified using a range list or a coefficient list. You can change the type of list that is used for a damage system. You can specify kill data for catastrophic kill, firepower kill, and mobility kill. (Many of the systems provided with VR-Forces only configure catastrophic kill.)

If you are using a range list, edit the ranges and range values as described in "Editing a Range List Probability Table," on page 68-40.

If you are using a coefficient list, edit the probability table as described in "Editing a Coefficient List Probability Table," on page 68-41.

77.4. Viewing Inter-Object Damage Probability Results

Sometimes when VR-Forces users create a new weapon system or damage system, the systems do not work as expected. The system might not fire on a target or might not damage it when it does. This is usually due to a mismatch in systems or a misunderstanding of how damage probabilities are calculated. The Damage Probabilities viewer shows you what you can expect when a specific entity fires on another specific entity.

To view damage probability results:

1. Choose **View** → **Damage Probability Results**. The Damage Probabilites viewer opens (Figure 77-2).

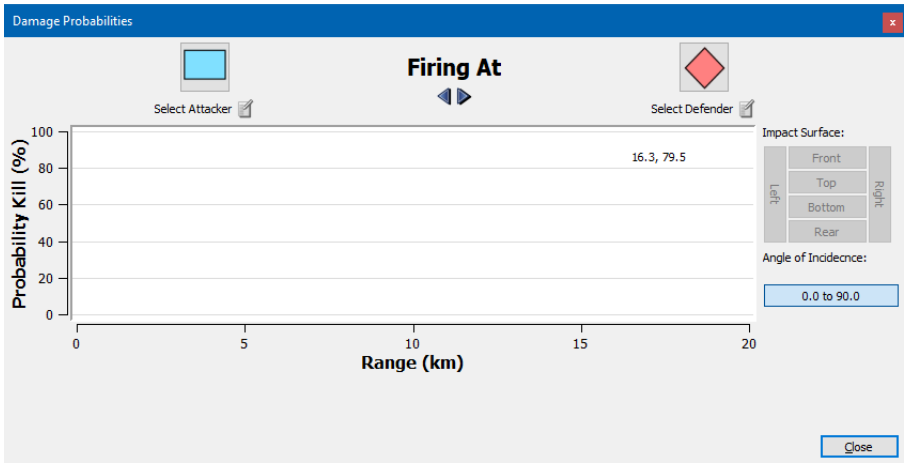


Figure 77-2. Damage Probabilities viewer

2. Click the Select Attacker button. The Select Simulation Object dialog box opens. It is an object list similar to the object list in the SOE or the Simulation Objects Panel.
3. Select the entity that you want to test as the attacker.
4. Click OK. The entity name is displayed under the button.
5. Click the Select Defender button. The Select Simulation Object dialog box opens.
6. Select the entity that you want the attacker to fire on.
7. Click OK. The entity name is displayed under the button. The viewer redisplays to show the probability graph for each weapon that the attacker has (Figure 77-3).

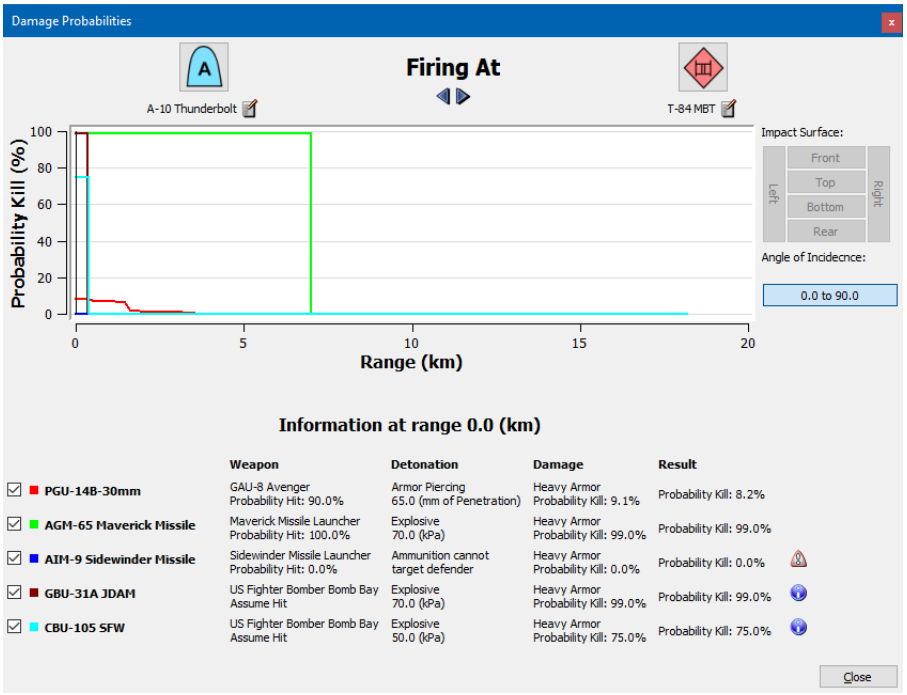


Figure 77-3. Damage Probabilities viewer with entities selected

8. Click in the graph at the range for which you want to see the probability of kill result. The Information table title (Information at range x.x) shows the range you have selected. The Result column in the Information table shows the probability that each weapon system on the attacker has of damaging the defender.
9. If the defender has multiple impact surfaces configured, select the surface whose probability you want to view.
10. If the defender has multiple angles of incidence configured, select the angle of incidence whose probability you want to view.
11. Optionally, clear the check boxes next to individual weapon systems to filter the number of graphs in the window.
12. Optionally, to reverse the relationship between attacker and defender, click the Firing At arrow buttons.
13. Optionally, to see the configuration page for the attacker or defender, click the Edit button (✎) next to the entity name.
14. Optionally to see the configuration page for the weapon, munition, or damage system, click the label in the appropriate column.
15. Optionally, click an Alert button (⚠) or Information button (ℹ) to display explanatory information, such as the fact that a weapon system cannot target the selected defender.

77.5. Configuring Damage for Dynamic Terrain

Dynamic terrain damage is configured using the damage-by-power approach. It is configured in *./data/simulationModelSets/base/vrfSim/systems/other/dynamic-terrain.sysdef* in the settings for the dynamic-terrain-munition-damage controller. Terrain can be damaged using standard terrain damage or high fidelity terrain damage.

77.5.1. Standard Terrain Damage

When you use standard terrain damage, the controller publishes a series of concentric damage areas of decreasing damage from the impact point. The type of terrain damage (that is, the switch in the terrain to be updated) is defined by **standard-terrain-damage-type** in *dynamic-terrain.sysdef*. The detonation power domain that causes this damage is defined by **standard-power-type**. The `standard-terrain-damage-table` block specifies the concentric areas. When a detonation is received, for each entry in this table a new area is created based on the range that produces the specified power. Then the switch in the terrain model is set to the specified damage setting.

The following code is from *./data/simulationModelSets/base/vrfSim/systems/other/dynamic-terrain.sysdef*:

```
(standard-terrain-damage-table
  (damage-by-power-entry
    (power 15)
    (damage "slight")
  )
  (damage-by-power-entry
    (power 40)
    (damage "moderate")
  )
  (damage-by-power-entry
    (power 75)
    (damage "destroyed")
  )
)
```

This table generates three dynamic terrain areas. The smallest area is defined by the area on the terrain that experiences an explosive power of at least 75. It is switched to a value of “destroyed”. The next larger area is the area that experiences power between 40 and 75. It is switched to a value of “moderate”. Finally the outer area is the area that experiences power between 15 and 40. It is switched to a value of “slight”.

77.5.2. High Fidelity Terrain Damage

You can configure high fidelity damage areas along with the standard damage. To determine where high fidelity damage areas should be created, VR-Forces queries the terrain to find any dynamic terrain features that match the given **dynamic-terrain-type** setting. This can be used to generate additional or more specific types of damage to select locations in the terrain that support a higher fidelity of damage. Just as with standard damage, a **power-type** is specified that corresponds to the detonation power domain that is causing this damage. High fidelity terrain damage is configured in the `high-fidelity-terrain-damage-table` block. It has the same format as the `standard-terrain-damage-table`.

The high fidelity damage lets you specify the damage-algorithm used. VR-Forces only supports the range-based algorithm, but you can develop your own algorithms in a plug-in. If you add additional damage algorithms, they might require configuration options that are not supported in the `high-fidelity-damage-table` provided by MAK. If necessary, you can specify additional parameters using the **additional-algorithm-config** parameter, which accepts a list of key/value string pairs.

High fidelity damage can produce a secondary detonation. This can be specified by setting **detonation-on-destroy** to True and defining the **detonation-munition** parameter. The **delay** parameter defines a delay in seconds between when power was first experienced and when the damage should be done. This allows the secondary detonation to create a particle effect that can mask the underlying terrain switch.

The following code is from *./data/simulationModelSets/base/vrfSim/systems/other/dynamic-terrain.sysdef*:

```
(high-fidelity-terrain-damage-table
  (entry
    (dynamic-terrain-type "damage_house")
    (power-type "explosive")
    (damage-algorithm "range-based")
    (delay 2.0)
    (detonate-on-destroy True)
    (detonation-munition 2 0 0 4 2 0 0)
    (terrain-damage-table
      (damage-by-power-entry
        (power 15) ;; explosive power is in kPa
        (damage "slight")
      )
      (damage-by-power-entry
        (power 40)
        (damage "moderate")
      )
      (damage-by-power-entry
        (power 70)
        (damage "destroyed")
      )
    )
```

```
      (additional-algorithm-config
      ;; Additional key/value pairs (string values) can be added here for
      ;; custom algorithm configuration
      ;; For example:
      ;; (my-custom-algorithm-config "config value string")
      )
    )
  )
```

77.6. Configuring Detonation Results

The SISO Enumerations Document specifies a list of detonation results, such as Entity Impact, Ground Impact, Detonation, and Air Burst. When a munition detonates, a detonation-result message is sent on the network. Applications can use this information to determine how to visualize the detonation or for other purposes.

By default, VR-Forces uses a small subset of the detonation results specified in the SISO Enumerations Document. You can customize the detonation result messages sent by VR-Forces by editing the detonation-results-table in *./data/simulationModelSets/base/detonationParams.mtl*.

For each munition type, you can configure the detonation results for entity impacts, surface impacts, and mid-air detonations. You can specify the detonation results using the strings listed in the comments at the top of the detonation-results-table or you can use any of the integer values specified in the SISO Enumerations Document for the different detonation result types. The *DtDetonationManager* converts the strings or integers into the appropriate detonation result messages to send over the network. (For details about the *DtDetonationManager*, please see *VR-Forces Developers Guide* class documentation. If editing *detonationParams.mtl* does not meet your needs, you can use the API to override *DtDetonationManager*.

The following code shows the entry in the detonation-results-table for an anti-ship missile:

```
(detonation-result-entry
  (munition
    ;; Anti-Ship Missile (incl. Exocet)
    (munition-type  2 6 -1 1 -1 -1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (entity-impact-result-list
    (default-entity
      (entity-type -1 -1 -1 -1 -1 -1 -1)
      (detonation-result "entity-impact")
    )
  )
  (surface-impact-result-list
    (default
      (surface-type "default")
      (detonation-result "ground-impact")
    )
    (water
      (surface-type "water")
      (detonation-result "detonation")
    )
    (undefined
      (surface-type "undefined")
      (detonation-result "detonation")
    )
    (ground
      (surface-type "ground")
      (detonation-result "ground-impact")
    )
  )
  (mid-air-detonation-result "detonation")
)
```



78. Configuring Movement Systems

This chapter describes configuration files that affect simulation object movement.

Configuring Formations 78-2
Formation Files 78-2
Creating a User-Defined Formation 78-4
Configuring Obstruction Avoidance 78-4
Configuring Object Types 78-5
Configuring Vector Feature Types 78-5
Configuring Embarkation 78-6

78.1. Configuring Formations



Although you can create and edit formation files by hand as described in this section, the recommended way to create and edit formations is to use the Simulation Object Editor.

The object parameter database lists the formations available to a unit in the formations element in its *.entity* file. Each formation is identified by its name and the path to the formation file, for example:

```
<formations paramName="formation-list" autoLayout="0">
  <formation formationName="Line" formationFile=
    "@(formation-dir)/platoonLine.frm"/>
  <formation formationName="Column" formationFile=
    "@(formation-dir)/platoonColumn.frm"/>
  <formation formationName="Wedge" formationFile=
    "@(formation-dir)/platoonWedge.frm"/>
  <formation formationName="Vee" formationFile=
    "@(formation-dir)/platoonVee.frm"/>
</formations>
```

78.1.1. Formation Files

A formation file specifies the name of the formation and has an entry for each simulation object in the default formation definition. Each entry specifies a promotion-id, a leader-promotion-id, and a position-offset, where:

- The promotion-id specifies the order in which simulation objects are promoted to unit leader if the leader is incapacitated.
- The leader-promotion-id specifies the unit member from whom the entry's position will be offset.
- The position-offset specifies how many meters behind, to the right, and above the entry will follow its leader. The position-offset is specified in terms of the body coordinate system of its leader.

The following is a portion of the formation file for a column formation:

```
(column-formation
   (entry-0
      (promotion-id  0)
      (leader-promotion-id  -1)
      (position-offset  0.000000 0.000000 0.000000)
   )
   (entry-1
      (promotion-id  1)
      (leader-promotion-id  0)
      (position-offset  -25.000000 0.000000 0.000000)
   )
   (entry-2
      (promotion-id  2)
      (leader-promotion-id  1)
      (position-offset  -25.000000 0.000000 0.000000)
   )
   (entry-3
      (promotion-id  3)
      (leader-promotion-id  2)
      (position-offset  -25.000000 0.000000 0.000000)
   )
)
```

In a column, each simulation object follows the simulation object in front of it by 25 meters. Therefore, the **leader-promotion-id** for each entry is the **promotion-id** of the simulation object in front of it, and each entry has a **position-offset** of -25 meters.

The number of entries in a formation file is finite. However, it is possible to place more simulation objects into a unit than there are entries in a formation file. If this happens, VR-Forces increments the **promotion-id** of the last entry in the formation file and assigns the new simulation object the same **position-offset** as the last entry in the formation file. For a simple formation like a column, this works well. If you are using a more complex formation, this mechanism might create unintended relationships among simulation objects.

If you want to change the layout of the formations provided with VR-Forces, you can edit the formation files.

78.1.2. Creating a User-Defined Formation

You can create formations other than those provided by VR-Forces. It is recommended that you create formations in the Simulation Object Editor. If you use the Simulation Object Editor, you do not have to edit any formation files or *.entity* files by hand.

To create a formation:

1. Using one of the formation files supplied with VR-Forces as your template, place as many entries as you want in the formation file and specify their offsets.
2. Save the new formation under a new name.
3. Add an entry to the formation list block in the *.entity* file for each unit that you want to be able to use the formation. Be sure to use the same structure and naming conventions as the default entries. For example:

```
<formations paramName="formation-list" autoLayout="0">
  <formation formationName="Line" formationFile="@(
    formation-dir)/DI-companyLine.frm"/>
  <formation formationName="Column" formationFile="@(
    formation-dir)/DI-companyColumn.frm.frm"/>
  <formation formationName="Wedge" formationFile="@(
    formation-dir)/DI-companyWedge.frm"/>
  <formation formationName="Vee" formationFile="@(
    formation-dir)/DI-companyVee.frm"/>
</formations>
```

We recommend that you configure your formations so that simulation objects follow the leader, rather than the next closest member of the formation (as is done in column formation.) This ensures that formations have the best chance of maintaining their formation as they maneuver.

78.2. Configuring Obstruction Avoidance

You can configure ground entities to avoid specific entity types, vector features, or both.



The more vector feature types and entity types specified for avoidance, the more memory and processing power required.

Obstruction avoidance is influenced by the stopping-distance-factor and lateral-clearance parameters. If entities display erratic behavior, try raising the values, lowering the speed of the entities, or both.

78.2.1. Configuring Object Types

To specify the types of objects that an entity type should avoid, add or remove the appropriate object type from the obstruction sensor entry in the movement system definition file. For example, the following object types are configured for ground entities:

```
(object-type
  (object-type  1 (18 -1 -1 2 -1 -1 -1)) ;; vrfObstacles object type
  (object-type  1 (1 -1 -1 -1 -1 -1 -1)) ;; DtPlatform entity kind
  (object-type  1 (3 -1 -1 -1 -1 -1 -1)) ;; DtLifeForm entity kind
)
```

78.2.2. Configuring Vector Feature Types

Configuring avoidance of vector features relies on the attributes on the feature data as well as the named queries defined in *./appData/settings/feature-config.txt*. To configure specific feature data for obstruction avoidance, modify the feature-types-to-avoid type list in the obstruction sensor in the appropriate system definition file. The types in this file are named feature queries, and you can add as many as you need.

Example of *human.sysdef*:

```
(obstruction-sensor
  ...
  (feature-types-to-avoid
    (type "MAK_OBSTACLE")
  )
)
```

Example of *air-cushion-default.sysdef*. This entity specifically calls out building, infrastructure, and vegetation, instead of obstacle, because obstacle also includes waterway.

```
(obstruction-sensor
  ...
  (feature-types-to-avoid
    (type "MAK_BUILDING")
    (type "MAK_INFRASTRUCTURE")
    (type "MAK_VEGETATION")
  )
)
```

78.3. Configuring Embarkation

Entities and units that are configured to allow entities to embark on them have an occupancy-director-controller component. Its parameters configure embarkation, as follows:

- The locations at which entities can embark on an entity are configured as ingress-points.
- The locations from which entities can disembark are configured as egress-points.
- The number of entities that can embark on an entity is defined by the embarkation-slots parameter.

The embarkation-slots parameter has one or more embarkation-slot-entry parameters that identify a location on the entity where an embarked entity can be placed. The embarkation-slot-entry parameters also specify the orientation of the embarked entity and the types of entities that can occupy the slot.

The ingress-points and egress-points parameters each have one or more load-entry parameters. The load-entry parameters specify one or more load-entry-point parameters. If there is one load-entry-point, an embarked entity enters the parent entity at that point and is immediately placed in an available slot. If a load-entry has multiple load entry points, they specify the points in a route that the embarking entity follows to reach its slot. This configuration is for large parent entities, such as aircraft carriers, on which entities land and then taxi to a slot.

You configure embarkation in the Simulation Object Editor. For details, please see Chapter 72, *Configuring Embarkation*.



79. Configuring Emitters and Radios

This chapter describes configuration files for emitters and radios. For information about configuring the display of emitters, please see Chapter 91, *Configuring Emitter Volumes*.

Configuring Emitters 79-2
Configuring Radios 79-4
Publishing Radio Transmitters 79-4

79.1. Configuring Emitters

All fixed-wing aircraft have emitters as part of their radar sensor. Emitters have search modes. Search modes have a list of beams. You can configure entities with emitters that have more than one beam. You can also configure multiple emitters. The basic radar sensor with an emitter and two search modes (*./data/simulationModelSets/<model_set>/vrfSim/systems/sensors/active-radar-sensor.sysdef*) is as follows:

```
(active-radar-sensor-system
  (systems )
  (sensors
    (radar-sensor
      (component-descriptor-type "radar-signature-sensor-descriptor")
      (component-type  "radar-signature-sensor")
      . . .
      (sensor-domain  "radar")
      (sensor-offset $sensor-position)
      (sensor-positional-error  0.000000)
      (detection-level-determinator
        (determinator-type  "signature-detection-level-determinator")
        (detection-level-to-set-hostility  3)
        (combat-identification-level-table-file
          (filename  "$(detection-dir)\std-radar-detection-table.csv")
        )
      )
      (detection-period  0.000000)
      (publish-emitter-system True)
      (emitter-system
        (system-name-enum  5570)
        (system-function-enum  5)
        (radar-mode-list
          (search-mode
            (index  0)
            (radar-mode  "Search")
            (beam-list
              (beam-0
              (index  0)
              (type  "track")
              (frequency  4999999488.000000)
              (frequency-range  10000.000000)
              (power  70.000000)
              (pulse-repetition-frequency  0.000000)
              (pulse-width  0.000000)
              (azimuth-center  0.000000)
              (azimuth-sweep  0.523599)
              (elevation-center  0.349066)
              (elevation-sweep  0.523599)
              (sweep-sync  0.000000)
              (beam-function-enum  1)
              (beam-param-index  0)
            )
          )
        )
```

```
          (track-mode
            (index  1)
            (radar-mode  "Track")
            (beam-list
              (beam-0
                 ...
              )
            )
          )
        )
      )
    )
  )
```

To configure multiple emitters for an entity, add additional radar-sensor blocks. Each block must have a unique name. For example:

```
(radar-sensor-1
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type  "radar-signature-sensor")
  . . .
  (emitter-system
    ...
    )
  )
)
(radar-sensor-2
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type  "radar-signature-sensor")
  . . .
  (emitter-system
    ...
    )
  )
)
```

To add multiple modes on an emitter, add additional mode blocks to the radar mode list. The names of radar modes (for example, track-mode and search-mode) and the value for the radar-mode parameter are arbitrary. You can name them whatever you want.

To add multiple beams on a radar mode, add additional `beam-list` blocks and change the name of each block. For example:

```
(search-mode
  (index  0)
  (radar-mode  "Search")
  (beam-list
    (beam-0
      ...
    )
    (beam-1
      ...
    )
  )
)
```

79.2. Configuring Radios

All radios are *DtVrfRadio* classes or subclasses. The radio is responsible for managing incoming and outgoing radio messages on a single simulation object. Each radio is associated with a specific communication model, and uses this communication model to send and receive messages.

Radios are configured in platform (OPE) files. The radios block in the OPE file lists all the radios on the simulation object and the parameters for each radio. Parameters include the type of radio descriptor and type of radio class, which are strings used to create the C++ classes from the factories. The comm-model-name parameter identifies which communication model a radio is associated with. Every radio must be associated with a communication model.

A typical radio entry is as follows:

```
(main-radio
  (radio-descriptor-type "radio-descriptor")
  (radio-type  "default-radio")
  (comm-model-name  "default-radio-model")
  (publish-transmitter True)
  (transmitter-params
    (radio-type  7 1 225 2 1 20)
    (initial-frequency  3000000.000000)
    (power  30.000000)
  )
)
```

79.2.1. Publishing Radio Transmitters

The external communication model requires that any simulation objects sending or receiving messages though the communications effects server publish a DIS or HLA radio transmitter object. By default, the internal communication model does not publish radio transmitters.

Publishing radio transmitters is controlled by two parameters:

- The publish-transmitters parameter in *./data/simulationModelSets/<model_set>/commModelParams.mtl*. By default, this parameter is False for the simple communication model and True for the external communication model.
- The publish-transmitter parameter in an entity's OPE file. This parameter is set to true for all object types.

To publish radio transmitters, both of these parameters must be True. If you do not change the OPE files and set publish-transmitters to True, all simulation objects will publish radio transmitters. If you set publish-transmitters to True and do not want all simulation objects to publish radio transmitters, you can set publish-transmitter to False for the object types that you do not want to publish them.



80. Joystick and Keyboard Controls

This chapter explains how to configure joysticks and the keyboard to control entity movement.

Using a Joystick to Control Entities 80-2
Switching Between Systems 80-3
Disabling Joystick Control 80-4
Using the Keyboard to Control an Entity 80-4
Configuring Joysticks and Keyboard Control 80-4
Creating a Joystick Configuration 80-5
Mapping the Joystick Stick 80-7
Mapping Joystick Buttons and Keyboard Keys 80-8
Configuring “Switch Between” Options 80-11
Using Multiple Joysticks 80-14
Specifying a Favorite Joystick Configuration 80-15
Installing a Gamepad on Linux 80-15

80.1. Using a Joystick to Control Entities

VR-Forces supports use of joysticks, other types of game controllers, and keyboard keys to control ground, fixed-wing, and rotary-wing entities. You can attach multiple joysticks, which allows you to control multiple entities, movement and weapons fire on one entity, or some combination. You can also use a joystick and keyboard mappings together to control an entity.

Joystick control is managed in the front-end. Joysticks should be attached to the computer from which you are running the front-end. Control messages are sent to the back-end, which can be on any computer in the simulation.

- Red Hat 7 does not support joysticks by default, but it can be enabled in the operating system. For details, go to this web site: https://access.redhat.com/solutions/64114
- You cannot enable or disable joystick support for all entities at once. You can only enable joysticks for individual entities.
- Embarked entities controlled by joystick cannot disembark. For example, if you use a joystick to control a fixed-wing taking off from a carrier, the aircraft will take off, but it will remain in an embarked state.
- For information about how to configure joysticks, please see "Configuring Joysticks and Keyboard Control," on page 80-4.

To enable joystick control:

1. Select the entity for which you want to enable joysticks.
2. Choose **Objects** → **Joystick** → **Take Control**. The Take Control dialog box opens.

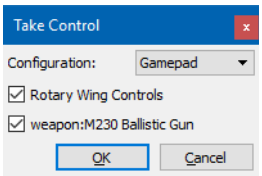


Figure 80-1. Take Control dialog box

3. Select a joystick configuration from the Configuration list. (If you have specified a joystick configuration as a favorite, you do not have a choice of the configuration to use.)
4. Select the entity functions you want to control, usually movement and one or more weapons. (You can have more than one joystick attached and can also use keyboard controls, if they are configured.)
5. Click OK.

80.1.1. Switching Between Systems

Systems that are controlled by joysticks can be organized into functional groups so that you can configure them similarly and switch between the systems using a "switch between" key or button. VR-Forces includes a switch group for weapons. If an entity has multiple weapon systems, you can manage them in the following ways:

- Configure all weapons to use the same keys or joystick controls for the same function. For example, to fire any weapon, you could press **`f`**. Use the switch between key to switch between weapons for a given entity. You only need to learn one set of controls to manage all your weapons.
- Configure all weapons to use different keys or joystick controls for the same functions. For example, to fire a 120mm gun, press **`f`**; to fire a 7.62mm gun, press **`g`**. You would not use the switch between option.

If an entity has multiple weapons and you use switch options to change which one you control, there is no visual cue in the GUI to show you which weapon you are currently controlling. If there are more than two weapons on an entity, the switch option cycles through them.

You can create your own switch groups. For details, please see "Configuring "Switch Between" Options," on page 80-11.

Configuring Multiple Identical Systems

If an entity has more than one instance of the same system, you must change the name of the function group if you want to control them separately. For example, suppose you want to mount two chain guns on an Apache helicopter. If they both have the same function group name (`weapon:M230 Ballistic Gun`), then you could not switch between them. You would need to rename one of them. This is done in the Simulation Object Editor when you add the additional system. (For complete details about using the Simulation Object Editor, please see Chapter 67, *The Simulation Object Editor and SMSs* through Chapter 72, *Configuring Embarkation*.)

To rename a system function group:

1. On the Start menu, choose **MAK Technologies** → **Simulation Object Editor**. The Simulation Object Editor opens.
2. Load *EntityLevel.sms*.
3. Select the entity whose system you want to edit.
4. In the systems list, for example the Weapons list, select the system you want to edit.
5. Click the Properties button for the system. A Properties dialog box opens.
6. Edit the function group name. For example, change `weapon:M230 Ballistic Gun` to `weapon:M230 Ballistic Gun 1`.
7. Click OK.
8. Save the SMS.

80.1.2. Disabling Joystick Control

To disable joystick control:

1. Select the entity for which you want to disable joysticks.
2. Choose **Objects** → **Joystick** → **Release Control**.

80.2. Using the Keyboard to Control an Entity

The joystick configuration process allows you to configure keyboard control of entity movement and weapons. You can use keyboard control along with joysticks. For example, you could control entity movement from the keyboard and weapons from a joystick, or the opposite.

To enable or disable keyboard control, follow the procedures for enabling and disabling joystick control. Similarly, to configure keyboard control, follow the joystick configuration process. For more information, please see "Using a Joystick to Control Entities," on page 80-2 and "Configuring Joysticks and Keyboard Control," on page 80-4.

80.3. Configuring Joysticks and Keyboard Control

You can control entity movement and weapon fire using a joystick, the keyboard, or both. You can attach multiple joysticks and can use a mix of joystick and keyboard control at the same time. You configure joystick and keyboard mappings on the Application Settings dialog box, Joystick Configuration page.

The controls available for configuration on the Joystick Configuration page are taken from the joystick controllers in the movement, gun, and other system definitions. For example, the *ground-tracked.sysdef* file has the following entry:

```
(joystick
  (component-descriptor-type "joy-controller-descriptor")
  (component-type  "ground-move-joystick-controller")
...
  (joystick-controls
    (steering
      (function-name "steering")
      (function-group $automotive-driving)
      (description "Controls the steering of this entity")
      (min-value -1.0)
      (max-value 1.0)
    )
    (throttle
      (function-name "throttle")
      (function-group $automotive-driving)
      (description "Controls the throttle of this entity")
      (min-value 0.0)
      (max-value 1.0)
    )
    (brake
      (function-name "brake")
      (function-group $automotive-driving)
      (description "Controls the braking of this entity")
```

```
            (min-value 1.0)
            (max-value 1.0)
            )
          (change-gear
            (function-name "change-gear")
            (function-group $automotive-driving)
            (description "Puts the entity in forward, reverse or pivot gear")
            (min-value 0)
            (max-value 2)
            (cycle True) ;; Each click cycles from min to max
          )
        )
    )
```

Based on this system definition file, the Joystick Configuration page (Figure 80-2) has an Automotive Driving function with controls for brake, change-gear, steering, and throttle.

The min-value and max-value parameters determine how the edit buttons on the Joystick Configuration page work, as follows:

- If min-value and max-value are both set to 1 (as in the brake control in the example system definition file), the control is a key mapping button ( ). You can set it to Yes or No.
- If max-value is an integer greater than 1 (as in the change gear control in the example system definition file), the control is a key mapping button. You can configure keyboard buttons to increase or decrease the control by 1, or to set it to a specific value.
- If min-value does not equal max-value and max-value is less than or equal to 1 (as in the throttle control in the example system definition file), there is a joystick button ( ) and a key mapping button on the Joystick Configuration page. The keyboard button lets you specify a percentage increase or decrease of the control value. (For example, increase throttle by 10%.)

80.3.1. Creating a Joystick Configuration

A joystick configuration contains the joystick mappings and, optionally, keyboard mappings that you can use to control an entity. VR-Forces includes default configurations for a gamepad and a joystick.

To create a joystick configuration:

1. Create or open a scenario.
2. Choose **Settings** → **Application**. The Application Settings dialog box opens.
3. Select the Joystick Configuration page (Figure 80-2).

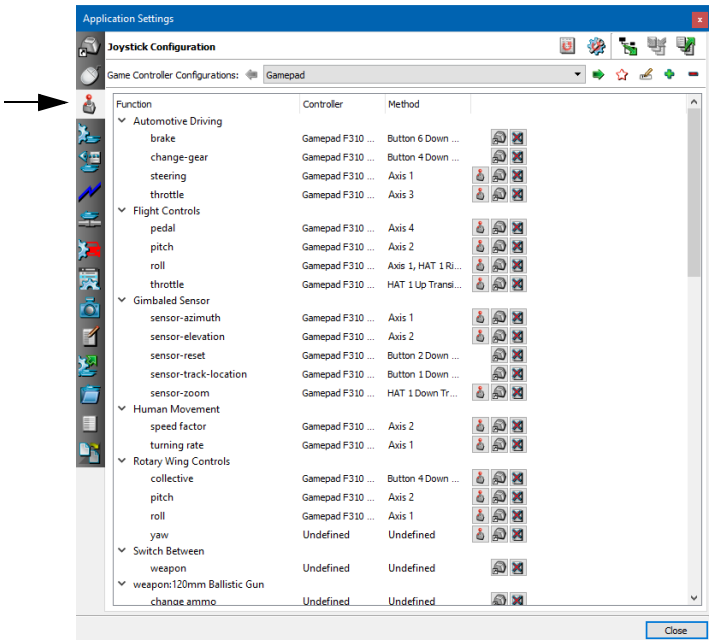


Figure 80-2. Joystick Configuration page

4. Click the Add button (+). The New Configuration dialog box opens.
5. Type a name for the configuration.

! The name of the joystick configuration must match the name that the joystick uses to identify itself to the computer.

---

6. Click OK. The window redisplays to show the functions that this device can control (Figure 80-3).

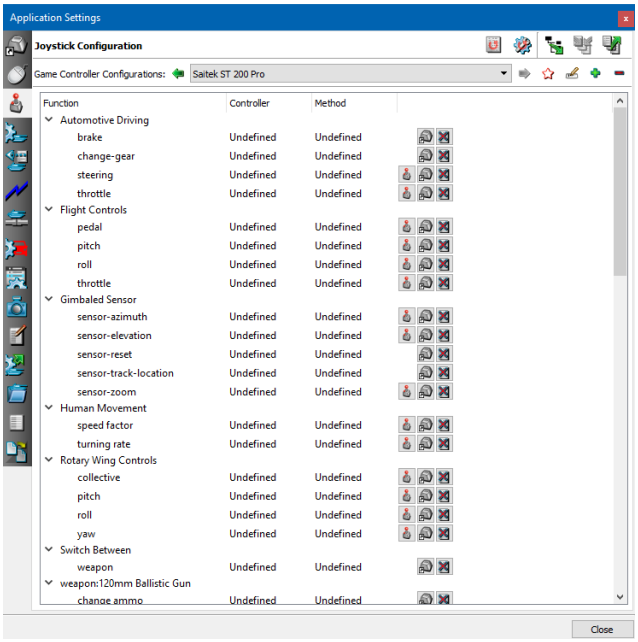


Figure 80-3. Joystick configuration

7. Edit the joystick and keyboard mappings as described in "Mapping the Joystick Stick," on page 80-7 and "Mapping Joystick Buttons and Keyboard Keys," on page 80-8.

80.3.2. Mapping the Joystick Stick

To control an entity with a joystick, you map the joystick controls to specific entity movement components and weapons.

To map joystick controls:

1. Create a joystick configuration, as described in "Creating a Joystick Configuration," on page 80-5.
2. On the Joystick Configuration page (Figure 80-3), click the Joystick button ( ) for the function you want to map. A joystick configuration dialog box opens (Figure 80-4).

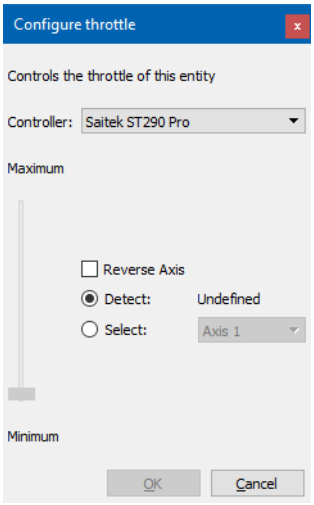


Figure 80-4. Joystick configuration dialog box

3. In the Controller list, select the controller you want to configure.
4. Configure each axis as follows:
   a. To let VR-Forces select the axis, select the Detect option and move the joystick in the axis you want to use for this control.
   b. To choose the axis, select the Select option and select an Axis from the list.
   c. Move the joystick in the axis you are configuring. If the Maximum/Minimum slider moves in the direction that you expect, you are finished. If the slider moves opposite to what you expect, select the Reverse Axis check box. It should now move as expected.
5. Click OK.
6. Repeat this process for each function you want to control with the joystick.

80.3.3. Mapping Joystick Buttons and Keyboard Keys

You can map joystick buttons and keyboard keys to entity movement control and weapon control. When entity control is enabled, the entity control key mappings override any other key mappings, such as observer control mappings.

To map joystick buttons and keyboard keys to entity control:

1. Create a joystick configuration, as described in "Creating a Joystick Configuration," on page 80-5.
2. On the Joystick Configuration page ("Joystick Configuration page," on page 80-6), click the Key Mapping button ( ) for the control you want to map. A configuration dialog box opens. It may call for a range of values (Figure 80-5), or just be a toggle (Figure 80-6).

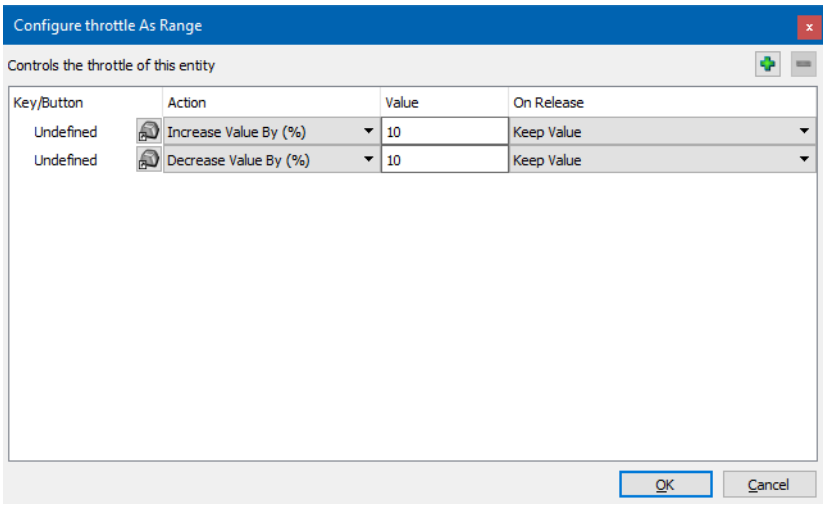


Figure 80-5. Key mapping configuration dialog box

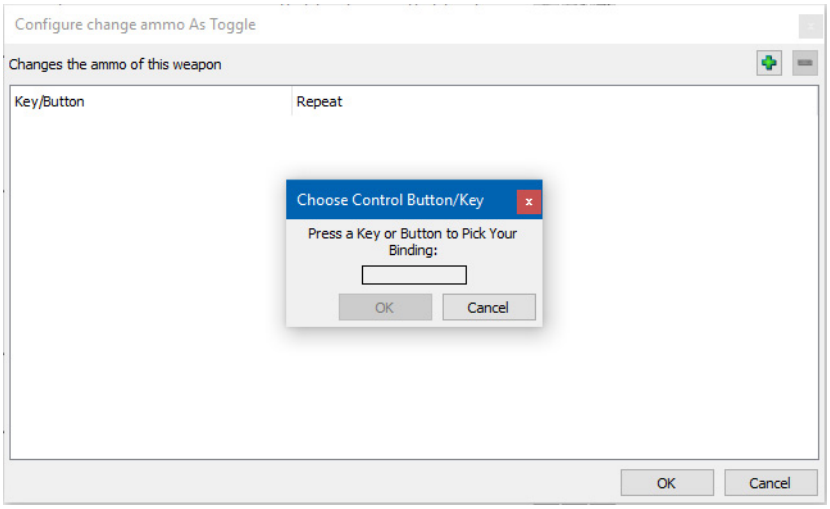
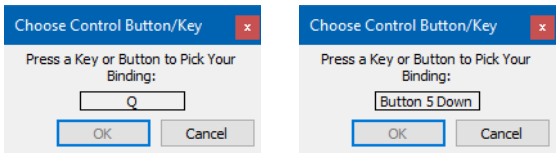


Figure 80-6. Key mapping configuration as toggle dialog box

3. If the mapping requires a range, for each row in the dialog box, do the following:

   a. Click the Key Mapping button (). The Choose Control Button/Key dialog box opens (Figure 80-6).

   b. Press the joystick button or keyboard key that you want to use to activate this control. It is displayed in the dialog box (Figure 80-7). (If a key binding is not available, the OK button does not become activated.)



Key Joystick button

Figure 80-7. Choose Control Button/Key dialog box with selection

c. Click OK.
d. In the Action column, select an option from the list.
e. In the Value column, enter an appropriate value for the Action option.
f. In the On Release column, select an option from the list. (This is what happens when you release the key.) A complete dialog box looks similar to Figure 80-8.

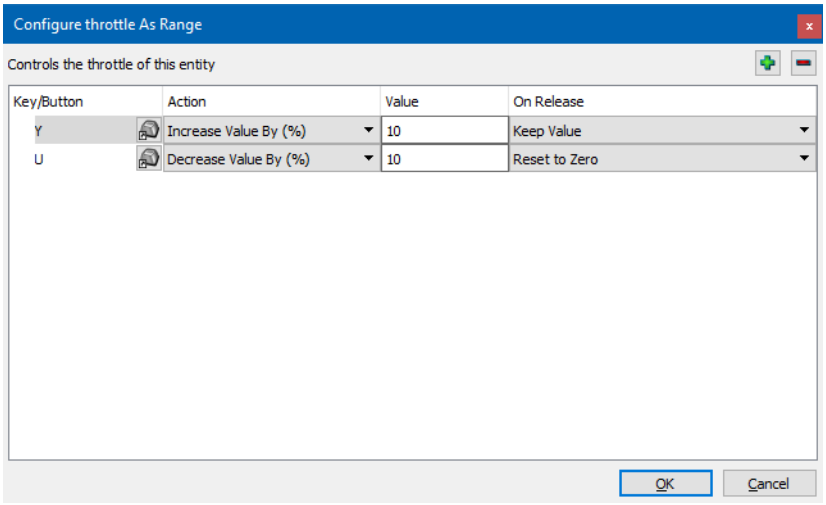


Figure 80-8. Key mapping configuration dialog box

4. If the keyboard mapping is a toggle, The Choose Control Button/Key dialog box opens automatically. Do the following:
   a. Press the key or joystick button that you want to use.
   b. Click OK.
   c. In the Repeat column, select Yes or No. Yes means that the action repeats if you hold down the key. No means it happens one time for each key press.
5. Click OK.
6. Repeat this process for each function you want to configure.

80.3.4. Configuring "Switch Between" Options

Some entities may have multiple systems of a certain type, such as weapons. You might want to be able to control all similar systems with the same set of keys and joystick controls. VR-Forces lets you do this with switch groups. You can specify a key or button to press to switch between the different systems in a switch group. For example, you might press a button to switch between a tank's main battle gun and a machine gun.

The controls for which you can specify a switch between option are listed in the Switch Between section of the Joystick Configuration page. Specifying the key or button to use is described in "Mapping Joystick Buttons and Keyboard Keys," on page 80-8. This section explains how to set up switch options.

To set up a switch option, when you define a function group in a system definition file's meta data, you use the syntax `switch_option:system`. The `switch_option` is a keyword. If an entity has multiple systems that use that keyword, you can switch between them using the switch between key or button. VR-Forces systems use the `weapon` keyword for switching between weapon systems. The following code, from *mm-gun.template_sysdef* shows how this works.

The controllers specify two function groups for joystick control – `$slew-group` and `$ballistic-gun-group`. These groups specify four functions that the joystick can control – slew, elevation, fire, and change ammo:

```
(weapon-120mm-gun
  (controllers
    (turret-joystick
      ...
      (joystick-controls
        (slew
          (function-name "slew")
          (function-group $slew-group)
          (description "Controls the slew of this weapon")
          (min-value -1.0)
          (max-value 1.0)
        )
      )
    )
...
    (main-gun-control-joy
      (joystick-controls
        (elevation
          (function-name "elevation")
          (function-group $ballistic-gun-group)
          (description "Controls the elevation of this weapon")
          (min-value -1.0)
          (max-value 1.0)
        )
        (fire
          (function-name "fire")
          (function-group $ballistic-gun-group)
          (description "Fires this weapon")
          (min-value 1.0)
          (max-value 1.0)
        )
        (change-ammo
```

```
            (function-name "change ammo")
            (function-group $ballistic-gun-group)
            (description "Changes the ammo of this weapon")
            (min-value 1.0)
            (max-value 1.0)
          )
        )
```

The meta-data section of the sysdef file specifies a value for these groups. It is specified as the keyword `weapon` followed by the name of the system – `weapon:120mm Ballistic Gun`:

```
(meta-data
  (system-name "Ballistic Gun (Elevating/Rotating)")
  (system-description "Template for a system that will allow for the
 Simulation Object to be equipped with a rotating, elevating mounted
 ballistic weapon.")
  (allowed-state-repository-types "ground-vehicle-param"
    "surface-entity-param" "rotary-wing-entity-param")
  (system-categories "weapon")
  (parameter-data-list
    ...)
    (string-parameter-data
      (parameter-name  "slew-group")
      (variable-type  "DtRwString")
      (display-label  "Slew Joystick Group Name")
      (display-units  "")
      (source-units  "")
      (list-items "")
      (list-display 0)
      (allow-edit True)
      (display-value True)
      (roll-up-rule "none")
      (default-value $default-slew-group
    )
    (string-parameter-data
      (parameter-name  "ballistic-gun-group")
      (variable-type  "DtRwString")
      (display-label  "Ballistic Gun Joystick Group Name")
      (display-units  "")
      (source-units  "")
       (list-items "")
      (list-display 0)
      (allow-edit True)
      (display-value True)
      (roll-up-rule "none")
      (default-value $default-ballistic-gun-group)
    )
...
```

Other weapon systems are configured similarly. On the Joystick Configuration page (Figure 80-9), the Switch Between group lists weapon as an option. The weapons configured for switching all show the `switch_option:sysdef` syntax.

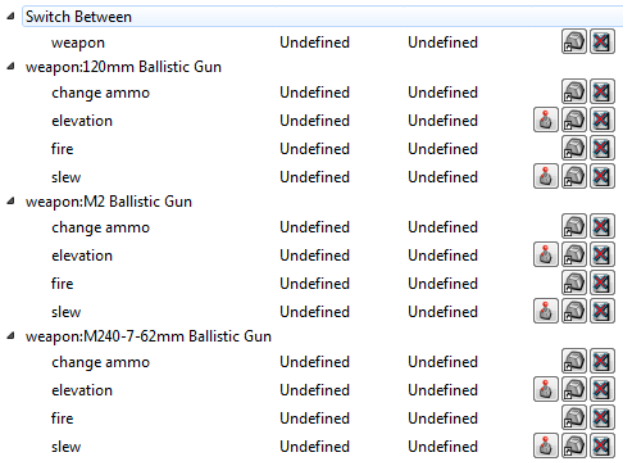


Figure 80-9. Switch between option

For details about using the switch between option, please see "Switching Between Systems," on page 80-3.

80.3.5. Using Multiple Joysticks

You can attach multiple joysticks or game controllers to your computer. You must create a configuration for each joystick that you want to use. When you take control of an entity, you select the configuration you want to use and the functions that you want to control with the selected configuration.

In Figure 80-10, the selected entity (a helicopter) has two functions that can be controlled by joystick or game controller.

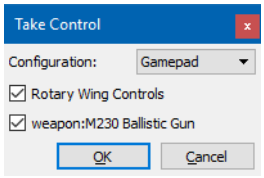


Figure 80-10. Take Control dialog box

You could select one configuration to control both, or you could control the two functions with different controllers, as follows:

1. Select the entity.
2. Choose **Objects** → **Joystick** → **Take Control**. The Take Control dialog box opens.
3. Clear the Rotary Wing Controls check box. This leaves the Gamepad configuration controlling the M230 ballistic gun.
4. Click OK.
5. Select the entity again and choose **Objects** → **Joystick** → **Take Control**. The Take Control dialog box opens (Figure 80-11). Only the Rotary Wing Controls function is available, because the ballistic gun is already under the control of the Gamepad configuration.

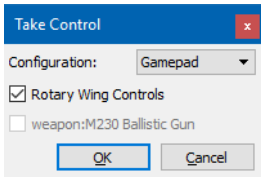


Figure 80-11. Take Control dialog box, one function available

6. In the Configuration list, select a different configuration (Figure 80-12).

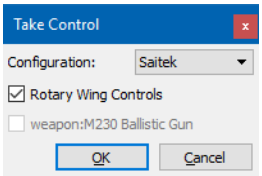


Figure 80-12. Take Control dialog box, second configuration selected

7. Click OK. The rotary wing controls are now controlled by the Saitek configuration and the ballistic gun by the Gamepad configuration.

80.3.6. Specifying a Favorite Joystick Configuration

Ordinarily, when you ask to take control of an entity with a joystick, you must specify the joystick configuration to use. You can specify a configuration as the favorite. In that case the configuration is automatically applied. However, you cannot choose a different configuration.

To specify a favorite joystick configuration:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Joystick Configuration page (Figure 80-2).
3. In the Game Controller Configurations list, select the configuration that you want to specify as the favorite.
4. Click the Star button (☆). A star is added to the configuration name in the list.

80.4. Installing a Gamepad on Linux

You can use Logitech Gamepads on Linux. You configure them like you would a joystick.

To install a Gamepad driver on Linux:

1. Download the xboxdrv driver for Linux.
2. Install the driver:

   ```
   yum install xboxdrv
   ```

3. Test to see if your controller is recognized:

   ```
   xboxdrv -L
   ```

   Your device should be listed win an ID next to it.

4. To test that the device is working:

   ```
   xboxdrv -i id
   ```

5. To get the device to work with VR-Forces, issue the following command:

   ```
   xboxdrv --detach-kernel-driver --silent
   ```



81. Attaching Components to Remote Entities

This chapter explains how to configure components on remote entities.

Attaching Components to Remote Entities 81-2

81.1. Attaching Components to Remote Entities

You can attach VR-Forces sensors, actuators, and controllers to remote (non-VR-Forces) entities. This feature allows you to add to the capabilities of the remote entity and better integrate it with the VR-Forces simulation. For example, you could add a radio to an entity or you could enable spot reporting. When you attach a component to a remote entity, you can write a plan for that entity that uses the attached components. For details, please see "Writing Plans for Remote Entities," on page 37-41.

VR-Forces includes an example scenario, remoteAttachment, that uses remote attachment. Its use requires that you run an external application, the f18 example that comes with VR-Link. For details about how to run the scenario, please see the readme file in *./data/scenarios/remoteAttachment*.

When you add a component to a remote entity, the application that is simulating the entity is responsible for updating its position and health status. VR-Forces simulates the attached components. You can specify that VR-Forces simulate the components on one back-end or across all back-ends.

Simulation of components on remote entities has the following limitations:

- Only individual platforms and lifeforms can attach components. Units and environmental process objects are not supported.
- VR-Forces components cannot control remote entities. That is, they cannot change their position or state. Therefore, you should not attach components such as movement systems to remote entities, because they will have no effect on the entities. Similarly, components that expect to be attached to a locally simulated entity such as an articulated part, may not work properly when attached to remote entities.

Components are attached to remote entities on the basis of entries in a component attachment table. The component attachment table provided by VR-Forces is *./data/simulationModelSets/<model_set>/extra/componentAttachmentTable.mtl*.

The component attachment table associates entities with attachment key-names. The following is a sample component attachment table:

```
(remote-configurations
  (attachment-entry
    ;; CIS MIG 29
    (attach-by object-type (obj-type 1 (1 2 222 1 2 1 -1)))
    (key-name "mig29-config")
  )
  (attachment-entry
    ;; US F-18
    (attach-by object-type (obj-type 1 (1 2 225 1 9 -1 -1)))
    (key-name "f18-config")
  )
)
```

The entries in this table associate the key-names mig29-config and f18-config with appropriate entity type enumerations. You can also associate a key-name with marking text, as follows:

```
(attachment-entry
  (attach-by marking-text "HMMWV 1)
  (key-name "hmmwv-attachments")
)
```

The key-names in the component attachment table have corresponding entries in *vrfSim.opd* in the remote-configurations block, as follows:

```
(remote-configurations
  (remote-configuration
    (key-name "f18-config")
    (configuration-path "$(remote-config-dir)/f18.mtl")
  )
  (remote-configuration
    (key-name "mig29-config")
    (configuration-path "$(remote-config-dir)/mig29.mtl")
  )
)
```

The key-names are associated with specific configuration files, which by default are located in *./data/simulationModelSets/<model_set>/vrfSim/remoteConfigurations*. The configuration files contain the components that are to be attached to the remote entity. For examples, please see the files in the *remoteConfigurations* directory.



XIII. Configuring the Graphical User Interface

The chapters in this section describe the display engine, windows, and channels, explain features of the graphical user interface (GUI), and how to configure menus and toolbars.



82. Display Engine and Window Management

This chapter explains how to add and configure windows and channels and how to save display engine configurations.

The Display Engine 82-2
- Window Types 82-2

Managing Display Engine Configurations 82-2
- Adding a Window 82-2
- Adding a Channel to a Window 82-4
- Removing a Window 82-5
- Removing a Channel 82-5
- Saving a Display Engine Configuration 82-5
- Loading a Display Engine Configuration 82-5

Changing a Window's Attributes 82-6

Changing a Channel's Attributes 82-7
- Setting the Clipping Planes 82-9
- Specifying the Projection Resize Policy Attribute 82-11
- Changing a Channel's Frustum (Field of View) 82-14
- Changing the Viewport 82-15
- Configuring Water Visibility 82-16
- Channel Keywords 82-17

Configuring Multichannel Displays 82-18
- Changing the Camera's Position and Orientation Offset 82-20

Stereoscopic Displays 82-21
- Configuring Anaglyphic Stereo 82-22
- Configuring Polarized Stereo 82-23

82.1. The Display Engine

The VR-Forces front-end has a component called a display engine. The display engine renders 3D and 2D data and can display data in one or more windows, which can each have zero or more channels. The windows and channels being rendered at any particular time make up a display engine configuration. You can save display engine configurations and can load them to quickly recreate a particular configuration.

82.1.1. Window Types

VR-Forces supports the following types of windows:

- Full screen. A full screen window uses the entire area of a monitor. It does not have any menus, panels, or other GUI controls.
- Resizable. A resizable window can be moved, resized, maximized, and minimized. It does not have any GUI controls. An inset view is an example of a resizable window. (For information about inset views, please see "Inset Views," on page 53-2.)
- Embedded. An embedded window is part of an application. The main display window in the VR-Forces front-end is an embedded window. The application provides a variety of controls that affect the view in the window. You could, if you wanted to, change VR-Forces's window to a different type, in which case it would no longer be embedded in the main application window.

A window can have zero or more channels. Each channel defines a viewable area on the screen. You must have at least one channel to see a rendered image in the window. Each channel can have its own observer.

82.2. Managing Display Engine Configurations

You can add windows to the display engine and you can add channels to the windows. The windows and channels make up a display engine configuration. You can save a display engine configuration and load a saved configuration.

82.2.1. Adding a Window

You can add as many windows as you want. New windows are named for the window type, for example Stealth Inset 1, Stealth Inset 2, Plan View Inset 1, and so on. Each window is created with one channel. You can add new windows from the File menu or in the Display Engine Configuration Editor.

➤ To add a window from the File menu, choose **File** → **New Stealth Window** (or **New Plan View Window**). A new window opens (Figure 82-1).

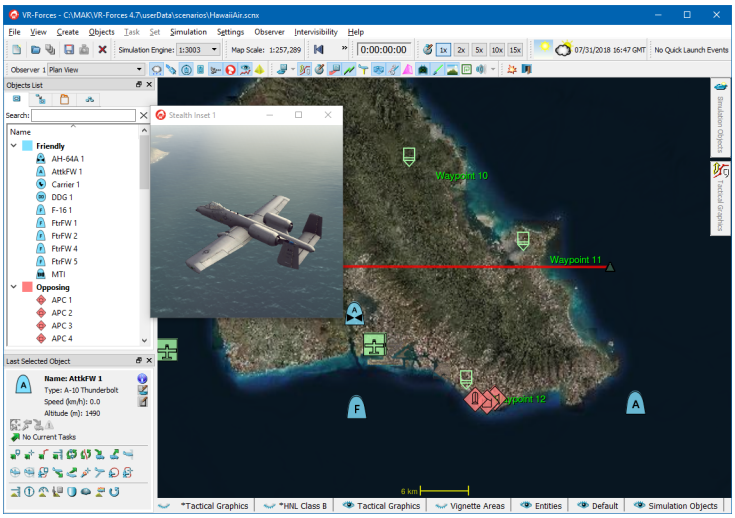


Figure 82-1. New window

To add a window from the Display Engine Configuration Editor:

1. Choose **View** → **Display Engine Configuration Editor Panel**. The Display Engine Configuration Editor Panel is added to the VR-Forces window (Figure 82-2).

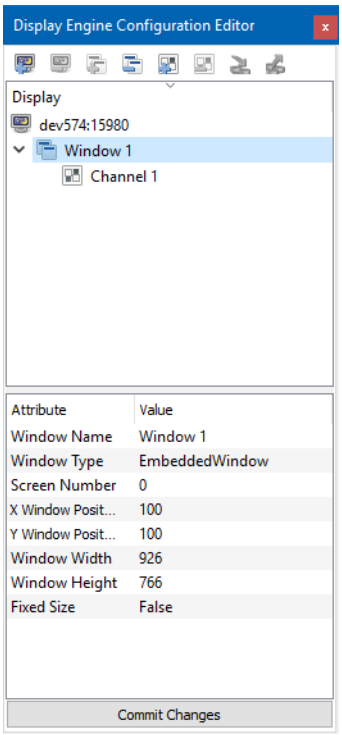


Figure 82-2. Display Engine Configuration Editor Panel

2. Select the display engine at the top of the Display tree.
3. Click the Add a Window button (), or right-click the display engine name and choose **Add a Window** on the context-sensitive menu. A new window opens and is listed in the Display Engine Configuration Editor Panel (Figure 82-3).

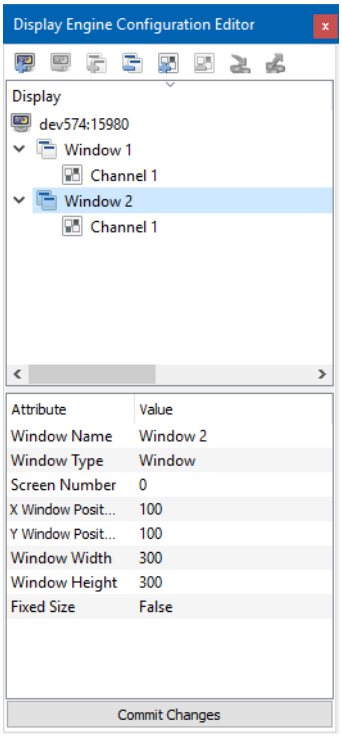


Figure 82-3. New window

82.2.2. Adding a Channel to a Window

Adding a channel to a window lets you configure multiple views within that window.

To add a channel to a window:

1. In the Display Engine Configuration Editor Panel, select the window to which you want to add a channel.
2. Click the Add a Channel button (), or right-click the window name and choose **Add a Channel** from the menu.

To see a difference between the channels, change the attributes of the new channel. For details about editing channel attributes, please see "Changing a Channel's Attributes," on page 82-7.

82.2.3. Removing a Window

To remove a window:

1. In the Display Engine Configuration Editor Panel, select the window that you want to remove.
2. Click the Remove Window button (), or right-click the window name and choose **Remove Window** from the menu.

82.2.4. Removing a Channel

To remove a channel:

1. In the Display Engine Configuration Editor Panel, select the channel that you want to remove.
2. Click the Remove Channel button (), or right-click the window name and choose **Remove Channel** from the menu.

82.2.5. Saving a Display Engine Configuration

You can save a display engine configuration so that you can easily replicate the configuration at a later time.

To save a display engine configuration:

1. Click the Save Display Engine Configuration button (), or right-click the name of the display engine and choose **Save Display Engine Configuration** on the menu. The Save Display Engine Configuration dialog box opens.
2. Type a name for the display engine configuration.
3. Click Save.

82.2.6. Loading a Display Engine Configuration

To load a display engine configuration:

1. In the Display Engine Configuration Editor Panel, select the display engine at the top of the window.
2. Click the Load Display Engine Configuration button (), or right-click the name of the display engine and choose **Load Display Engine Configuration** on the menu. The Load Display Engine Configuration dialog box opens.
3. Select the display engine configuration you want to load.
4. Click Open.

82.3. Changing a Window's Attributes

You can change the following attributes of a window:

- Window Name.
- Window Type (for details about window types, please see "Window Types," on page 82-2).
- Screen Number.
- Position (X, Y coordinates of upper left corner, in screen pixels).
- Width and height, in pixels.
- Fixed size.

To change a window's attributes:

1. Select the window in the Display Engine Configuration Editor Panel. A list of attributes is displayed at the bottom of the editor (Figure 82-2).
2. Click the value of the attribute that you want to change.
3. To change a numeric value, type or select a value.

   To change the Window Type or Fixed Size, select a value from the list.

   To change the name, type a new name.
4. Click Commit Changes. The window is updated.



You can also change a window's position by dragging it to a new location. You can change its size by resizing the window directly. The new values are shown in the attributes list the next time you select the window in the Display Engine Configuration Editor Panel.

82.4. Changing a Channel's Attributes

You can change the following attributes of a channel:

- Channel Name.
- Observer Name.
- Sensor. Use the sensor configured for the current observer, or override the observer and use one of the listed sensors.
- Projection Units – field of view, in angles, or database, in meters.
- Projection Resize Policy – fixed, vertical, or horizontal.
- Frustum (Left, Right, Top, and Bottom) (field of view).
- Z Near and Z Far (clipping planes).
- Near Far Clip Policy.
- Reduce Z Fighting Ratio.
- Dynamic Near Clip Attached Policy.
- Attached Near Clip Altitude.
- Fixed Near Clip When Attached.
- Dynamic Water Visibility Enabled.
- Dynamic Water Visibility Maximum.
- Render Order. Set the order in which channels are rendered in a particular window. For example, to display Channel 1 over Channel 2, set Channel 1 to render after Channel 2.
- LOD Scale Factor.
- Camera Position Offset (X, Y, Z).
- Camera Orientation Offset (Psi (heading), Theta (pitch), Phi (roll)).
- Viewport (Left, Right, Bottom, Top).
- Keywords – keywords to associate with the channel, for example, "showCockpits". (For a list of channel keywords, please see "Channel Keywords," on page 82-17.)

To change a channel's attributes:

1. Select the channel in the Display Engine Configuration Editor Panel. A list of attributes is displayed at the bottom of the editor (Figure 82-4).

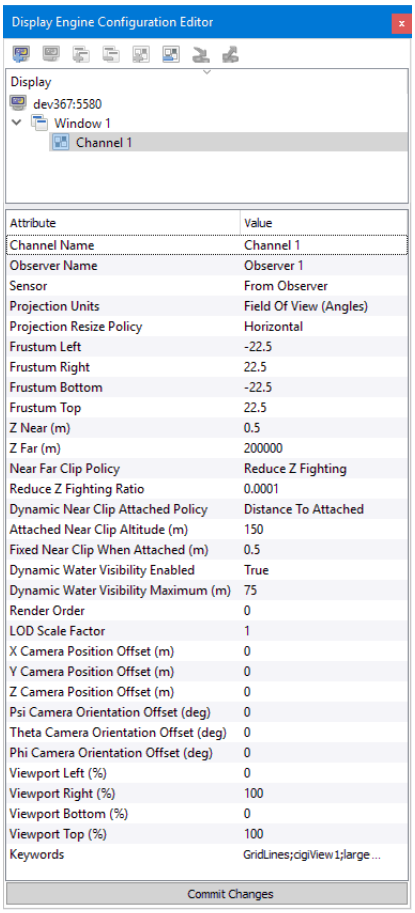


Figure 82-4. Channel attributes

2. Click the value of the attribute that you want to change.
3. To change a numeric value, type or select a new value.

   To change the Observer Name, Sensor, Project Units, Projection Resize Policy, or Parent Channel, select a value from the list.

   To change the Channel Name, type a new name. (For details about some of these attributes, please see the sections that follow this one.)
4. Click Commit Changes. The window is updated.

82.4.1. Setting the Clipping Planes

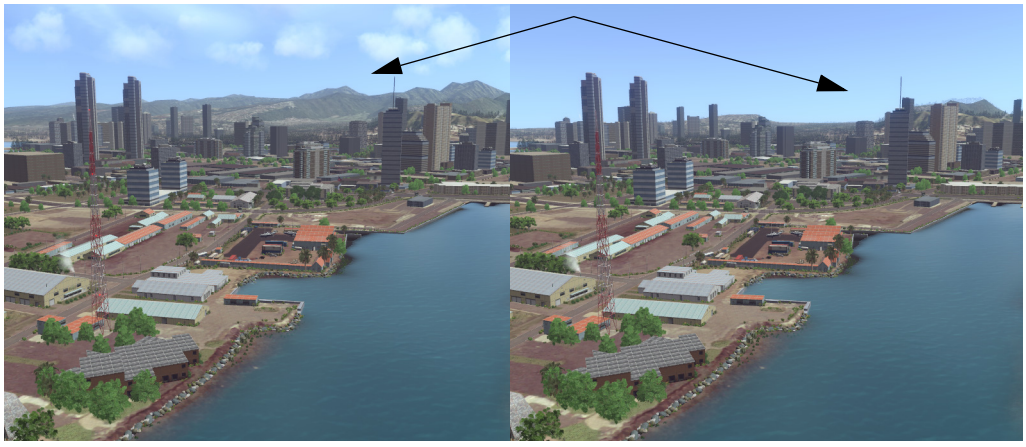
To improve performance, the graphics engine does not render anything that falls outside of a specified range of distances from the observer. This range is determined by the near and far clipping planes. Clipping planes are set on a per-channel basis.

Figure 82-5 illustrates the effect of changing the near clipping plane (the Z Near attribute). Figure 82-6 illustrates the effect of changing the far clipping plane (the Z Far attributes). (To see the effect of changing the near and far clipping planes, the Near Far Clip Policy must be set to Fixed.)



Z Near = 1 Z Near = 500

Figure 82-5. Near clipping



Z Far = 200000 Z Far = 5000

Figure 82-6. Far clipping

The ratio between the far clipping plane and the near clipping plane determines the precision with which the renderer can determine differences in depth. If this ratio becomes too small, then terrain and objects that are distant may not sort correctly, resulting in what is referred to as Z-fighting. VR-Forces provides different policies for managing the way near and far clipping planes are determined.

VR-Forces sets the clipping planes using one of the following policies (the Near Far Clip Policy attribute), plus their modifiers:

- Fixed. Clipping planes are set explicitly using the Z Near and Z Far attributes.
- Reduce Near Clip. Clipping planes are adjusted dynamically based on the observer's altitude. The near clipping plane is increased slightly as the observer rises to 20,000 meters, but stays under 1.5 meters. When the observer rises above 20,000 meters, the near and far clipping planes gradually increase. The Reduce Near Clip policy can be modified by the Dynamic Near Clip Attached Policy.
- Reduce Z Fighting. Clipping planes are set based on a ratio of the near clipping plane to the far clipping plane (the Reduce Z Fighting Ratio attribute). The near clipping plane can get as large as needed to maintain the ratio. This policy is the default clipping plane policy. The Reduce Z Fighting policy can be modified by the Dynamic Near Clip Attached Policy.
- Clamp Near Clip When Magnified. It is the same as Reduce Z Fighting, but adds an intersector to clamp the near clipping plane to prevent the near clip from going past objects or terrain in the scene when using view magnification. The intersector is only used when view magnification is 2x or more (this value can be changed through the API by extending the channel and overriding the call that calculates the clamp). This is an expensive operation, so only enable it if you really need it.

The Reduce Z Fighting clipping plane policy greatly reduces Z-fighting when you are using dynamic ocean. However, if the observer is attached to a simulation object and its altitude is moderately high, it may be closer to the entity than the near clipping plane and the entity may be clipped away. You can adjust the ratio to provide the best results. If you are not using dynamic ocean or you are frequently attaching to entities, you may want to use one of the other clipping plane policies.

If you set the Near Far Clip Policy to Reduce Z Fighting or Reduce Near Clip and the observer is attached to a simulation object, VR-Forces can dynamically change the near clipping plane based on the Dynamic Near Clip Attached Policy, as follows:

- Ignore Attached. Do not take entity attachment into account when setting the near clipping plane.
- Fixed Near Clip. Set the near clipping plane to the value of the Fixed Near Clip When Attached attribute if the observer altitude is >= the value of the Attached Near Clip Altitude attribute.
- Distance To Attached. Use the distance between the observer and the attached entity to determine the near clipping plane if the observer altitude is >= the value of the Attached Near Clip Altitude attribute. When this policy is selected, VR-Forces compares the near clipping value without this modifier to the value with it and uses the smaller of the two values. This is the default policy.



If the settings described in this section do not reduce Z-fighting, you can try using reverse Z buffering. To enable reverse Z buffering, start VR-Forces with the `--useReverseZBuffer` command-line option. The Oculus Rift HMD does not work with reverse Z buffering. Reverse Z buffering is not supported in 2D observer modes.

82.4.2. Specifying the Projection Resize Policy Attribute

The Projection Resize Policy determines how a channel behaves when you resize it. The options are:

- Fixed. Keep the aspect ratio the same regardless of how the window is resized.
- Horizontal. Change the horizontal field of view (FOV) to so that the ratio of the FOV angles matches the aspect ratio of the viewport.
- Vertical. Change the vertical FOV so that the ratio of the FOV angles matches the aspect ratio of the viewport.

Figure 82-7 illustrates the effect of different resize policies

You can change the projection resize policy for individual channels by editing the Projection Resize Policy attribute. You can also change the projection resize policy for all channels.

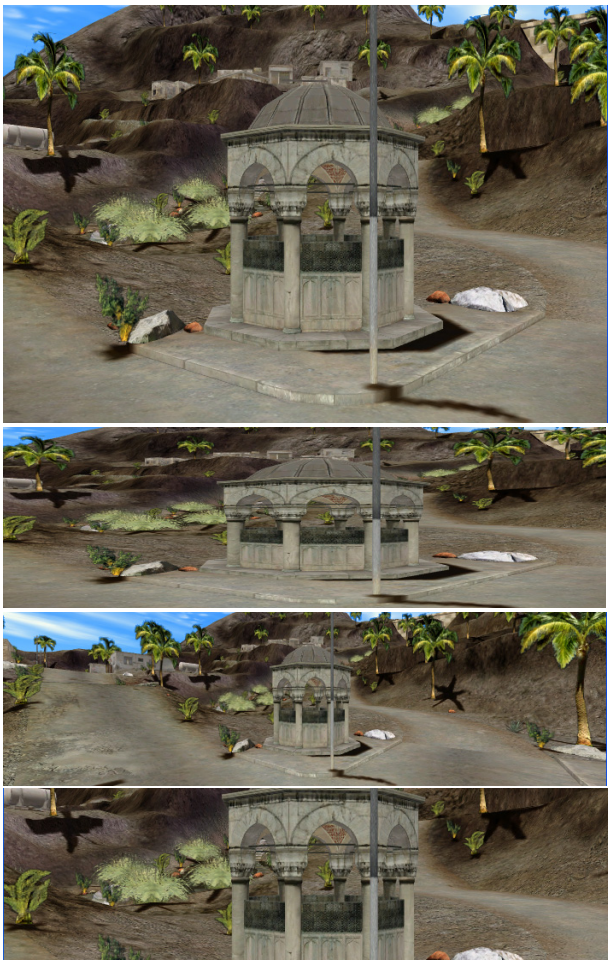


Figure 82-7. Effect of different projection resize policies

To change the projection resize policy for all channels:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 82-8).

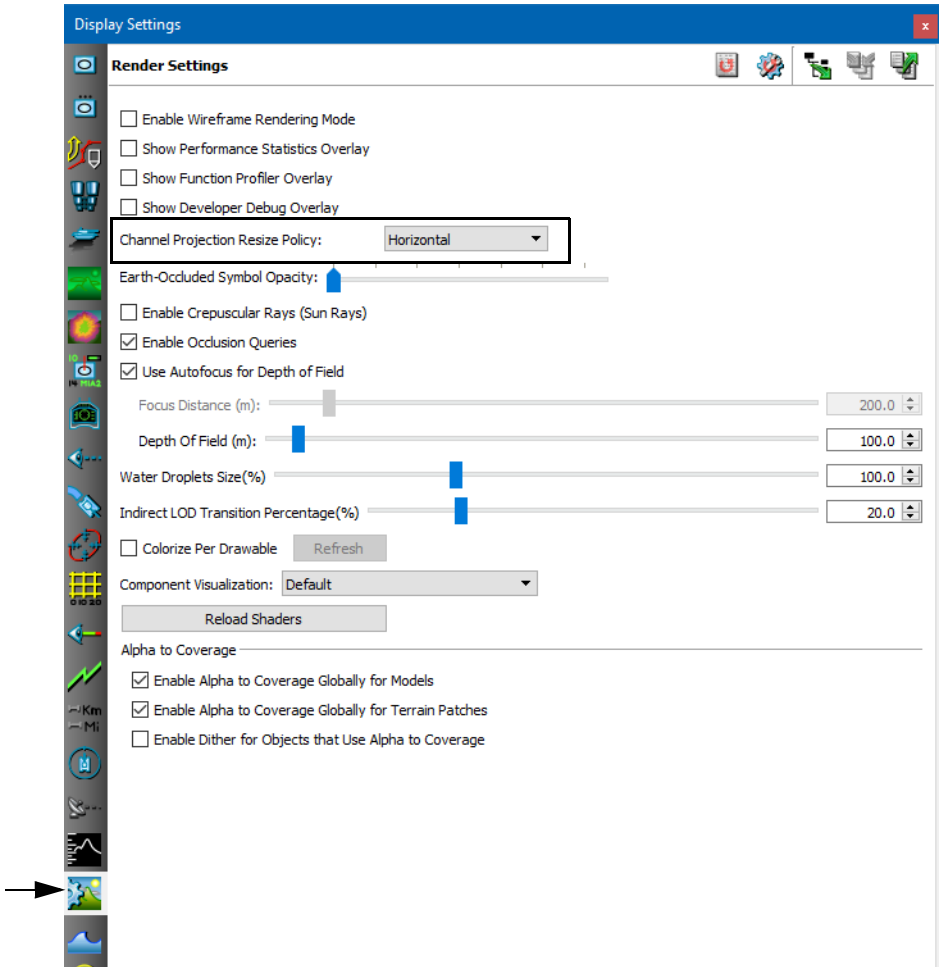


Figure 82-8. Display Settings, Render Settings page

3. Select an option on the Channel Projection Resize Policy list.

82.4.3. Changing a Channel's Frustum (Field of View)

The field of view setting affects perspective in a manner similar to a camera lens.

A wide field of view creates an effect like that of a wide-angle lens. Objects appear smaller and farther away from the observer, since the observer coverage spans a wider area. Depth distances between objects become exaggerated.

A narrow field of view creates an effect like that of a telephoto lens. Objects appear larger and closer to the observer, and the overall scene depth appears flattened. The distances between objects appears compressed. Since objects appear closer but the distance from the object to the camera has not changed, you may want to scale the distance used to determine the level of detail. You can do this by modifying the LOD Scale Factor attribute.

Figure 82-9 illustrates different field of view settings taken from the same observer location. The view on the left has frustum values -22.5, 22.5, -22.5, 22.5. The view on the right has the frustum values -10, 10, -10, 10.

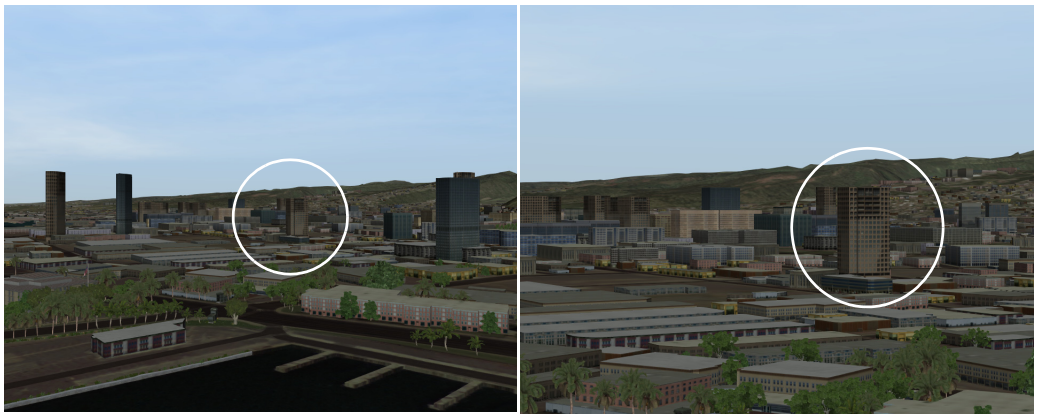


Figure 82-9. Field of view

➤ To change a channel's field of view, edit the frustum values in its attributes list.

82.4.4. Changing the Viewport

When you change the viewport of a channel, you change the portion of the window in which the channel is displayed. This does not change the extents of the scene. A narrow viewport compresses the scene width. A short viewport compresses the height. Figure 82-10 shows a window that has three channels, each with a different viewport. (The arrows show which portion of the window corresponds to each channel.) Notice that each channel has the same field of view.

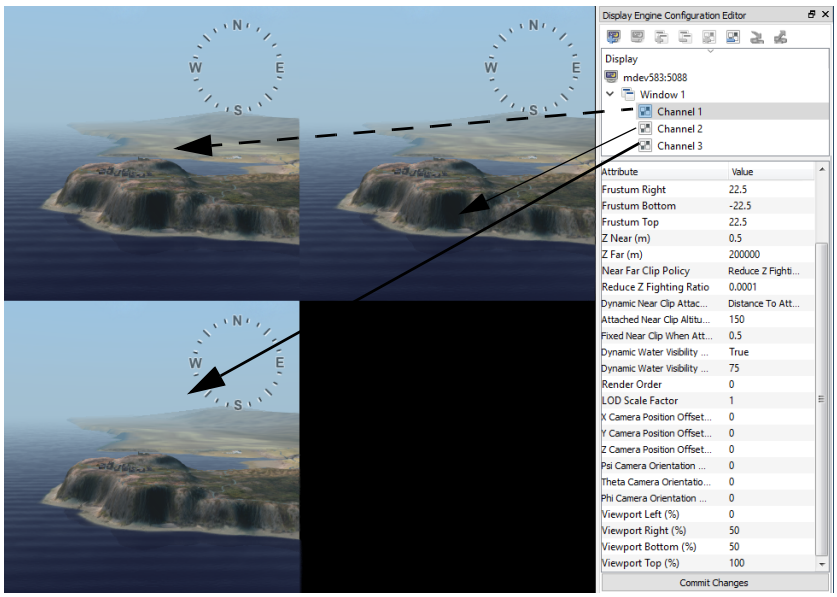


Figure 82-10. Window with three channels and three viewports

Viewports are specified as a percentage of the window from left to right and top to bottom. Figure 82-11 shows the percentage allocation for the viewports in Figure 82-10.

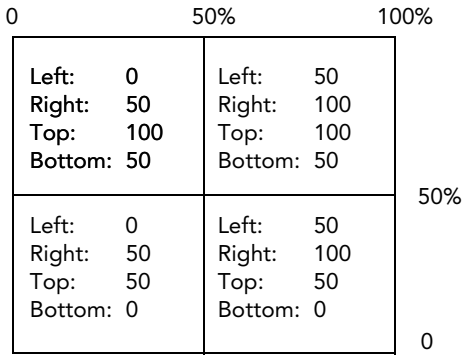


Figure 82-11. Viewport configuration

When you change viewports, the window might not completely refresh. To see the changes, resize the window.

82.4.5. Configuring Water Visibility

The water visibility attributes (along with the surface transparency setting on the Environment Conditions dialog box, Weather page) are designed to reduce Z-fighting. Z-fighting occurs when the altitude of the ocean surface and the depth of the ocean floor are very close to each other, usually around coastlines that have bathymetry data. The likelihood of Z-fighting decreases as the difference between the ocean surface and ocean floor increases, but it increases as the observer's altitude increases (because the ocean surface and ocean floor are proportionally closer to each other). Therefore there is a dynamic relationship as these factors change.

Adding transparency to the water reduces Z-fighting. The Dynamic Water Visibility Enabled attribute increases transparency automatically as the observer's altitude increases. Dynamic Water Visibility Maximum specifies the maximum depth to which transparency is added (Figure 82-12).

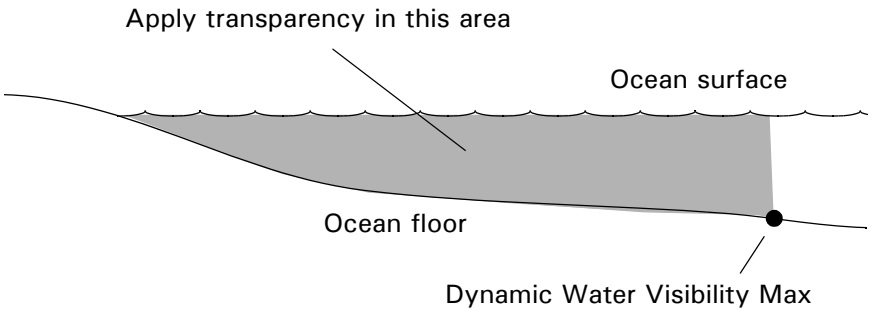


Figure 82-12. Dynamic water visibility

Dynamic water visibility is enabled by default. When Dynamic Water Visibility Enabled is True, VR-Forces increases the water visibility depth from the value Surface Transparency setting (at 0 meters altitude), to the maximum value specified at the current Ocean LOD Elevation. The default depth is 75 meters. This value provides reasonable performance given the default Ocean LOD Elevation (5000 meters), the altitude above which VR-Forces stops displaying dynamic ocean. If you change the Ocean LOD Elevation so that dynamic ocean is displayed when the observer is higher than 5000 meters, Z-fighting may increase and you may need to increase the Dynamic Water Visibility Maximum value. (For details about setting the Ocean LOD Elevation, please see "Setting the Ocean LOD Elevation," on page 57-17.)

There is no performance cost to enabling this attribute when you are not using dynamic ocean. If there are no underwater terrain polygons, these attributes have no effect.

82.4.6. Channel Keywords

VR-Forces supports the following channel keywords:

- largeCompass. If the compass is enabled, this channel shows the large compass. Default: Included in configuration of all channels.
- smallCompass. If the compass is enabled, this channel shows the small compass. Default: Not included in the configuration of any channels.
- GridLines. If grid lines are enabled, this channel shows the grid lines. Default: Included in configuration of all channels.
- showRemoteGraphics. If remote graphics are received this channel displays them. Default: Included for all channels except inset views.
- showCockpits. If GL Studio cockpits are enabled, this channel shows them. Default: Included in configuration of all channels.
- NoExternalInput. This keyword disables all mouse/keyboard/touch events for the channel. Default: Only included in the configuration of Sensor Gimbal View channels.
- cockpitOnly. If this keyword is true, the scene will not be rendered in the channel. Default: Included in the configuration of instrument panels only.

82.5. Configuring Multichannel Displays

People running simulations often want to set up a multichannel configuration using multiple monitors (on one or more computers) to display exercises. Typical configurations are a horizontal alignment, which widens the view, and the angled alignment, which surrounds the observer to simulate an out-the-window view (Figure 82-13).

You could set up these display configurations by having one computer with several monitors attached, or you could have multiple computers, each running a VR-Forces front-end. You might also want to have a 2D view on one monitor and a 3D view on another.

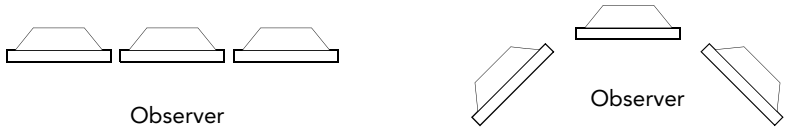


Figure 82-13. Multi-channel displays (top-down view)

By default, all windows and channels show the same view of the terrain (Figure 82-15, default window views). Therefore, if you want a setup in which each channel shows a different view of the terrain, you must configure the channels to calculate an offset from the master view.

The distribution of the view across multiple channels is controlled by the frustum. Figure 82-14 illustrates the default frustum for Channel 1. It has a 60^o field of view, from -30^o to $+30^o$.

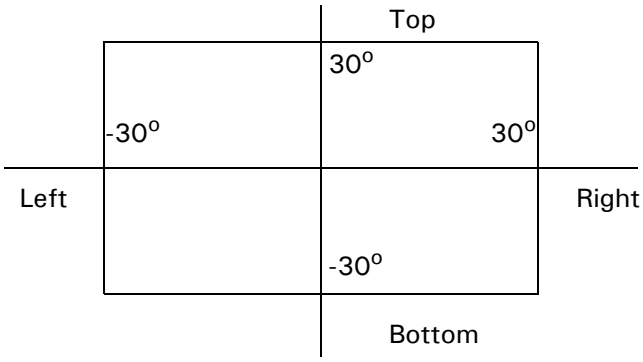


Figure 82-14. Default frustum for Channel 1

If you wanted to split the view into, for example, three horizontal channels, you would edit the left and right frustum values for each channel, as shown in Figure 82-15. The field of view is the same: -30^o to $+30^o$. However, now each channel shows a 20^o segment of the total field of view.

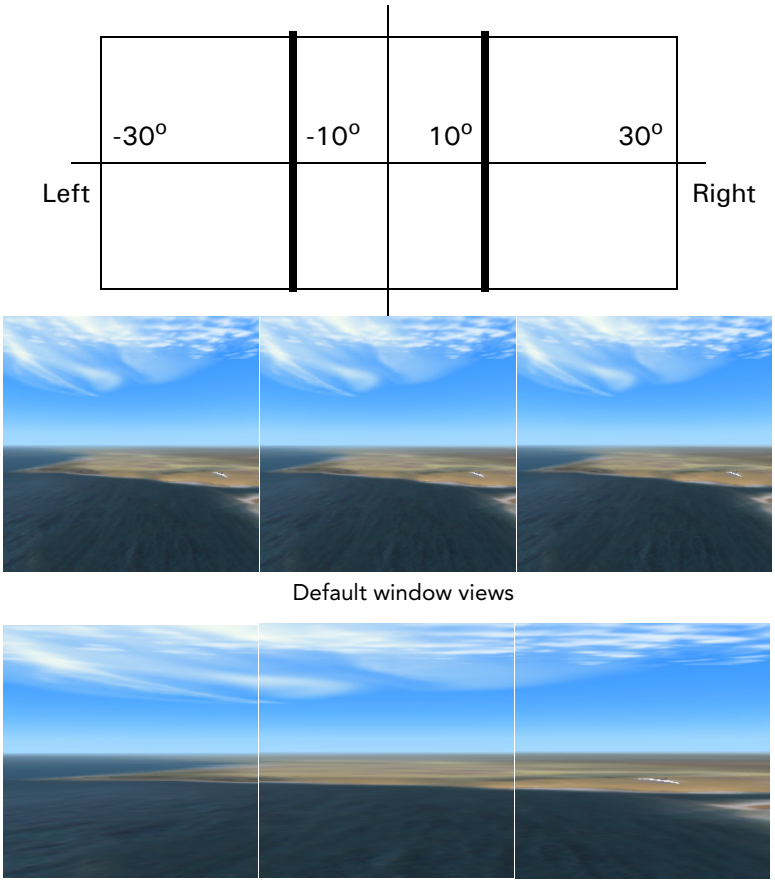


Figure 82-15. Frustum values for three channels

Once you configure the windows and channels you need to create the views you want, save the display engine configuration on each computer that you have configured. The next time you run the scenario, load the appropriate display engine configurations on each computer.

For a multi-monitor configuration to work correctly, use fixed size windows whose size matches the aspect ratio specified by the horizontal and vertical fileds of view and set the projection resize policy to fixed. (For information about the projection resize policy, please see "Specifying the Projection Resize Policy Attribute," on page 82-11.)

82.5.1. Changing the Camera's Position and Orientation Offset

Changing the frustum affects how much of the database is in the view. Changing the camera's position and orientation affects the point from which you look at the view. For example, imagine you are simulating the view from a vehicle, as in vehicle 1 in Figure 82-16. (The camera location is represented by the binoculars.)

If you wanted to show the view as experienced by a gunner riding in the back of the vehicle, or in a turret, you would change the X, Y, and Z camera position offsets in the channel's attribute list (vehicle 2).

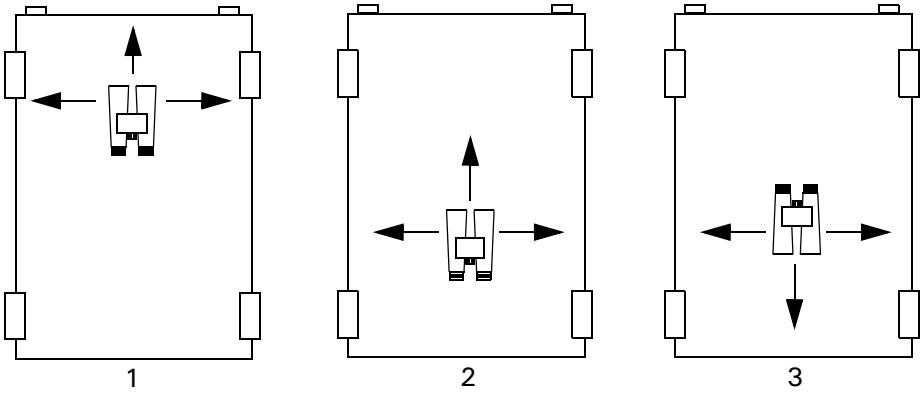


Figure 82-16. Position and orientation offsets

If you wanted the gunner to look out the rear of the vehicle, you would change its orientation, with the Psi (heading), Theta (pitch), and Phi (roll) Camera Orientation Offsets (vehicle 3).

82.6. Stereoscopic Displays

Stereoscopic display produces realistic three dimensional imagery by showing each of the viewer's eyes a different image, rendered from that eye's point of view. A stereoscopic display requires a system that can render two channels, one for each eye, and a method of displaying the correct channel to the correct eye of the viewer. Three common display methods are active stereo, polarized display, and anaglyphic stereo. OpenSceneGraph can support all three of these methods, and since VR-Forces uses OpenSceneGraph for most of its rendering, all three should be possible in VR-Forces.

- Active Stereo: In an active stereo setup, frames are alternated on the display for the left eye and the right eye, and a pair of shutter glasses blocks light to either the left or right eye in synchronization with the display. This method requires hardware support to synchronize the glasses to the display, along with a graphics card capable of quad buffering, and a display that can refresh at twice the desired frame rate. Active stereo should be possible with VR-Forces, but has not been tested.
- Anaglyphic Stereo: Anaglyphic stereo is the cheapest and simplest method to use. A typical setup is a display showing a red channel and a cyan channel simultaneously, along with a pair of glasses with one red lens and one cyan lens for the viewer. This will work with no special hardware aside from the inexpensive glasses. It is also the simplest method to set up in VR-Forces, requiring only that some OpenSceneGraph environment variables be set.
- Polarized Stereo: A polarized display shows the frames for both eyes simultaneously, but with their light polarized in different directions. The viewer wears glasses that have lenses polarized to match the two different frames, allowing each eye to see only the frame intended for it. Polarized stereoscopy requires some specialized equipment to display the images with the correct polarization. One simple method for producing this kind of display is to use two projectors with polarizing filters, projecting onto a rear projection screen. Monitors are available that can polarize the two frames. It is easy to configure a polarized display with VR-Forces.

82.6.1. Configuring Anaglyphic Stereo

You can configure OpenSceneGraph for anaglyphic stereo rendering through environment variables. To use VR-Forces in anaglyphic mode, run it with the environment variables described in Table 82-1.

Table 82-1: Environment variables for anaglyphic stereo

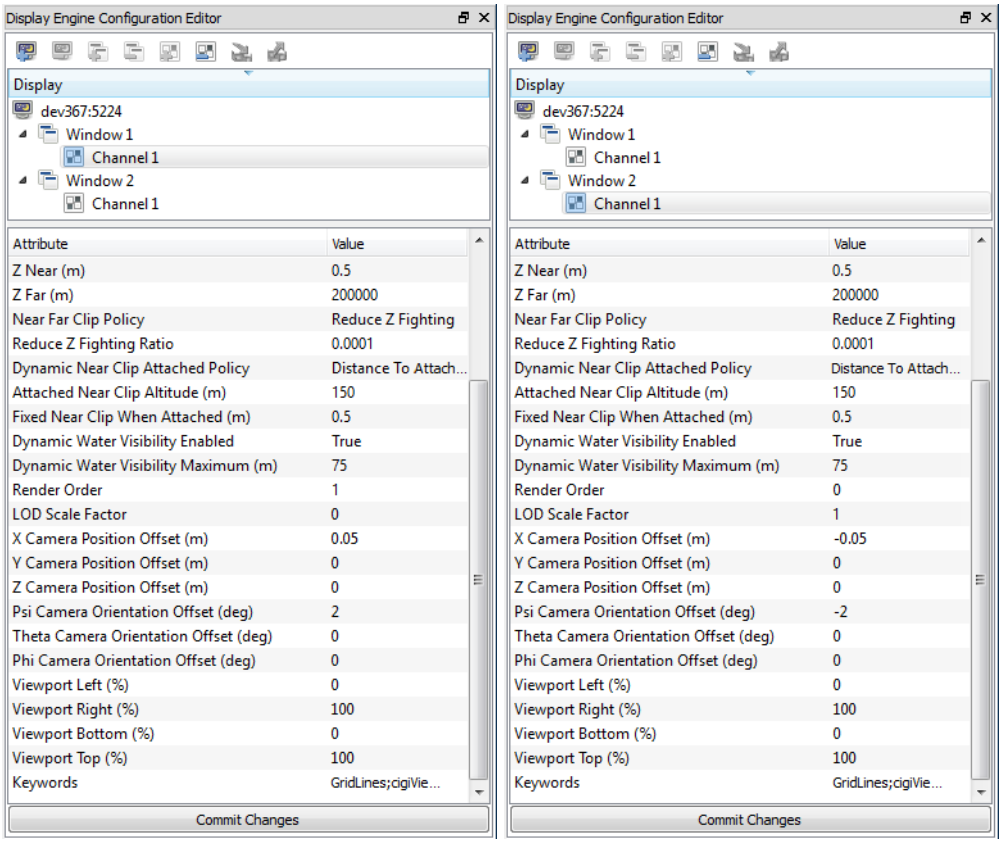
Environment Variable	Description	Default
OSG_STEREO	Turn stereo on.	ON
OSG_STEREO_MODE	Use anaglyphic stereo when in stereo.	ANAGLYPHIC
OSG_SCREEN_DISTANCE	Specifies the distance, in meters, that the viewer is from the screen.	0.50
OSG_SCREEN_HEIGHT	Specifies the height of the image, in meters, on the screen.	0.26
OSG_SCREEN_WIDTH	Specifies the width of the image, in meters, on the screen.	0.325
OSG_EYE_SEPARATION	Specifies the eye separation (interocular distance).	0.06

For more information, pleases see the OpenSceneGraph documentation at http://www.openscenegraph.org/index.php/documentation/guides/user-guides/62-stereo-settings.

82.6.2. Configuring Polarized Stereo

You can configure polarized stereo by taking advantage of VR-Forces's ability to produce multichannel displays. You configure one channel per eye. Each channel renders with a camera that is at a slight offset from the other channel's camera to represent the spacing between the viewer's eyes.

Varying the eye spacing and focal length can produce different effects, and the best values may depend on what the viewer is looking at. Figure 82-17 shows an example configuration.



Left channel Right channel

Figure 82-17. Polarized stereo channel configurations

To compute the correct channel configuration values:

1. Pick a focal length, `L`. Objects at this distance will appear to be in the plane of the display. Objects that are further away will appear behind the display, and objects closer will appear in front of it.
2. Pick an eye spacing, `E`, to represent the distance between the viewer's eyes. Human eyes are typically about 0.063m apart[1].
3. Set the X Camera Position values to `-E/2` meters for the left channel and `E/2` meters for the right channel.
4. Set the Psi Camera Orientation values to `-arctan(E/2)/L` degrees for the left channel and `arctan(E/2)/L` degrees for the right channel.

1. "Variation and extrema of human interpupillary distance", N. A. Dodgson, Proc SPIE 5291. Select publications link and scroll to 2004 at http://www.cl.cam.ac.uk/~nad10/pubs/EI5291A-05.pdf.



83. GUI Controls, Layouts, and Behaviors

This chapter provides details about using the various panels and toolbars in the front-end.

Dockable Panels and Toolbars 83-2
- Docking and Undocking a Panel 83-2
- Docking and Undocking a Toolbar 83-3
- Displaying and Hiding Panels and Toolbars 83-3

Saving Window Layouts 83-3
- Creating a Window Layout 83-4
- Choosing the Window Layout 83-5
- Setting a Window Layout as the Default Layout 83-5
- Reverting a Window Layout 83-6
- Updating a Window Layout 83-6
- Renaming a Window Layout 83-6
- Deleting a Window Layout 83-7
- Restoring a Factory Layout 83-7

Using Tear-Off Menus 83-8

Viewing the Window in Full-Screen Mode 83-8
- Hiding the Mouse in Full Screen Mode 83-9

Using Context-Sensitive Menus 83-10

Common VR-Forces Buttons 83-10

Adding Text to the Title Bar 83-11

Specifying Display Units 83-12

83.1. Dockable Panels and Toolbars

VR-Forces has floating, dockable panels for many of its important functions (Figure 83-1). It also has toolbars that let you quickly access the commands available on the menus.

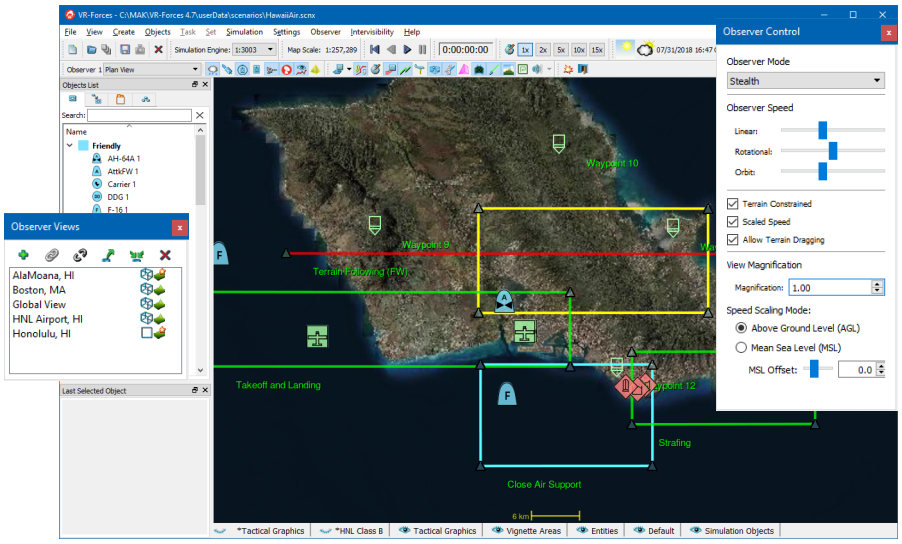


Figure 83-1. VR-Forces window and undocked panels

83.1.1. Docking and Undocking a Panel

To undock a panel:

1. Place the cursor over the panel's title.
2. Press and hold the left mouse button.
3. Drag the panel away from the window.

➤ To dock a floating panel, double-click its title bar.

83.1.2. Docking and Undocking a Toolbar

To undock a toolbar:

1. Place the cursor over the handle.



Figure 83-2. Toolbar handle

2. Press and hold the left mouse button.
3. Drag the toolbar away from the window.

➤ To dock a toolbar, drag it back to the toolbar strip. The other toolbars will move to create a space to drop in the toolbar (Figure 83-3).

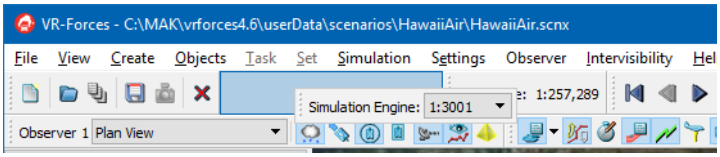


Figure 83-3. Docking a toolbar

83.1.3. Displaying and Hiding Panels and Toolbars

You can hide any of the panels or toolbars.

➤ To hide or view a panel or toolbar, select it on the View menu.

83.2. Saving Window Layouts

You can save window layouts to named layouts and can easily switch between them while VR-Forces is running. A layout stores the window size and the visibility and location of all toolbars and panels. You can specify that a layout be the default layout used at startup.

Layouts get saved to *./appData/settings/vrfGui*. Each layout is saved to a separate file using the naming convention layout_*layout_name*.uisx. For backup, you can save layouts to *./factory/settings/vrfGui*.

VR-Forces includes two layouts to get you started. The Scenario Creation layout has a set of toolbars and panels that users typically need when creating a scenario. Scenario Viewer removes the toolbars and panels to maximize the amount of the window that displays the scenario.

When you shut down VR-Forces, it saves the window layout. If you do not specify a default window layout, when VR-Forces starts up, it uses the saved window layout.

83.2.1. Creating a Window Layout

To create a window Layout:

1. Arrange the toolbars, panels, and window size in the layout you want.
2. Choose **View** → **Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 83-4).

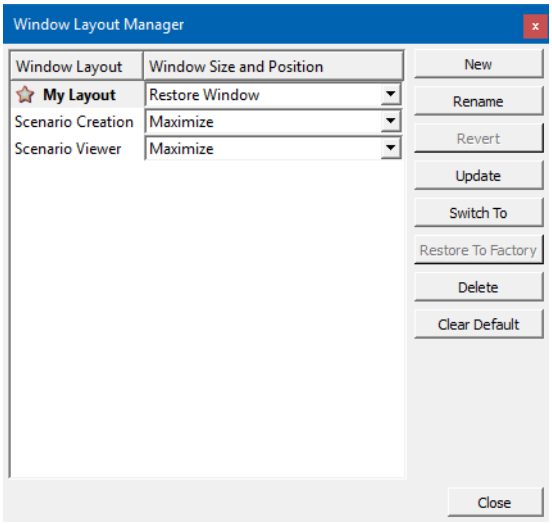


Figure 83-4. Layout Manager

3. Click New. The Layout Name dialog box opens.
4. Type a name in the Layout Name box.
5. Click OK.
6. Select one of the window size options:
   - Maximize. Maximizes the window when this window layout is selected, regardless of the window size when the layout is created.
   - Restore Window. Save the current window size as part of the window layout. This is the default option.
   - Full Screen Mode. Put the window into full screen mode when this layout is selected, regardless of the window size when the layout is created.
7. Click OK. The layout is added to the list of layouts.

83.2.2. Choosing the Window Layout

You can easily choose a window layout at any time. You can change layouts from the View menu or from the Layout Manager.

You can map keys to layouts to select them from the keyboard. Each layout can be mapped to a key (Figure 83-5). For details about mapping keys, please see Chapter 85, *Creating and Editing Key Mappings*.

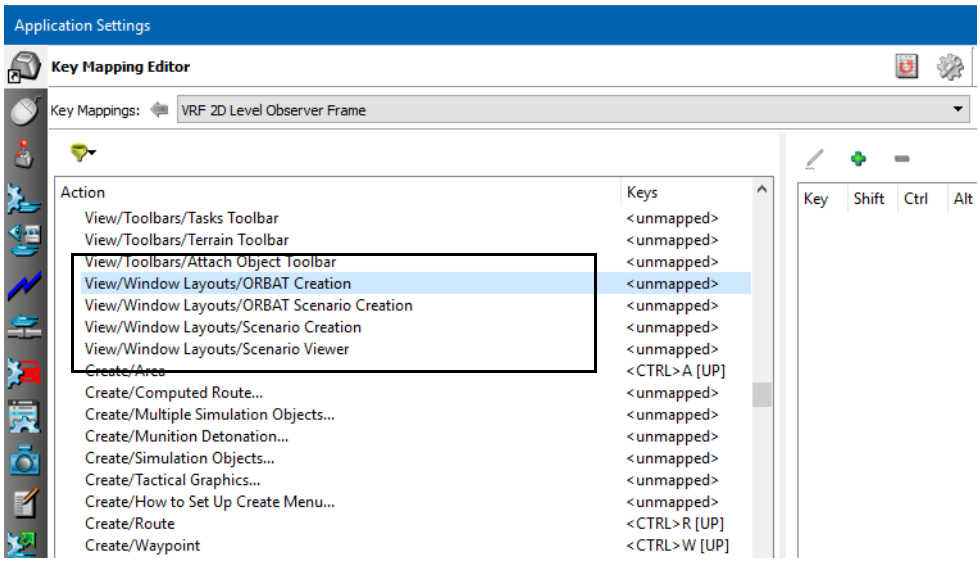


Figure 83-5. Layout action in Key Mapping Editor

- To choose a window layout, choose **View** → **Window Layouts** → *layout*, where *layout* is one of the listed named layouts.
- To change the window layout in the Window Layout Manager, in the list of layouts, double-click the layout you want to use or select the layout you want and click Switch To.

83.2.3. Setting a Window Layout as the Default Layout

When VR-Forces starts, if there is a default layout, it uses that layout. Otherwise, it uses the window layout it saved the last time you shut down VR-Forces.

To set a window layout to be the default layout:

1. Choose **View** → **Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 83-4).
2. In the list of layouts, select the layout you want to be the default.
3. Click Make Default. A star is displayed next to the name of the layout to indicate it is the default layout.

Clearing the Default Layout

If you have specified a default layout, you can specify that a different layout be the default or clear it so that there is no default layout.

To clear the default layout:

1. Choose **View** → **Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 83-4).
2. To specify a different default layout, select the desired layout and click Make Default.
3. To just clear the current default layout, in the list of layouts, select the default layout and click Clear Default.

83.2.4. Reverting a Window Layout

If you use a named window layout and then rearrange the window, you can revert to the saved layout.

➤ To revert a window layout, choose **View** → **Revert Window Layout**. (You can also revert a layout on the Window Layout Manager dialog box.)

83.2.5. Updating a Window Layout

If you load a window layout and then make changes to the layout, such as resizing the window or closing a panel, VR-Forces does not keep track of the changes. If you switch to a different layout or shut down, the changes are lost. If you want to save changes to a layout, you must explicitly update it.

To update a window Layout:

1. Choose **View** → **Window Layout Manager**. The Window Layout Manager opens (Figure 83-4).
2. Select the layout you are using.
3. Click Update.

83.2.6. Renaming a Window Layout

You can change the name of a window layout.

To rename a window layout:

1. Choose **View** → **Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 83-4).
2. Select the layout you want to rename.
3. Click Rename. The Layout Name dialog box opens.
4. Type a new name for the layout.
5. Click OK.

83.2.7. Deleting a Window Layout

You can delete window layouts that you have created.

To delete a window layout:

1. Choose **View** → **Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 83-4).
2. Select the layout you want to delete.
3. Click Delete.

83.2.8. Restoring a Factory Layout

VR-Forces includes two layouts. They are in *./appData/settings/vrfGui*, which makes them available to the Layout Manager. They are also in *./factory/settings/vrfGui* so that they can be restored. All layouts created after you install VR-Forces are saved to *./appData/settings/vrfGui*. If you want to back up your layouts, you can copy them to any backup location you want. If you want to be able to restore a layout in the Layout Manager, copy it to *./factory/settings/vrfGui*.

To restore a layout from the factory, it must be present in *./appData/settings/vrfGui* so that you can select it in the Layout Manager. If you delete a layout and then want to recover it, if you have saved a copy of it to *./factory/settings/vrfGui* or anyplace else, you can copy it to *./appData/settings/vrfGui*.

To restore a window layout to the factory layout:

1. Choose **View** → **Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 83-4).
2. Select the layout you want to restore.
3. Click Restore From Factory.

83.3. Using Tear-Off Menus

You can "tear-off" the Create menu and leave it displayed on the desktop.

To tear off the Create menu:

1. Click the Create menu.
2. Move the cursor over the grab bar (dashed line) at the top of the menu to highlight it (Figure 83-6).
3. Right-click. The menu is displayed as a window that you can leave open and reposition.

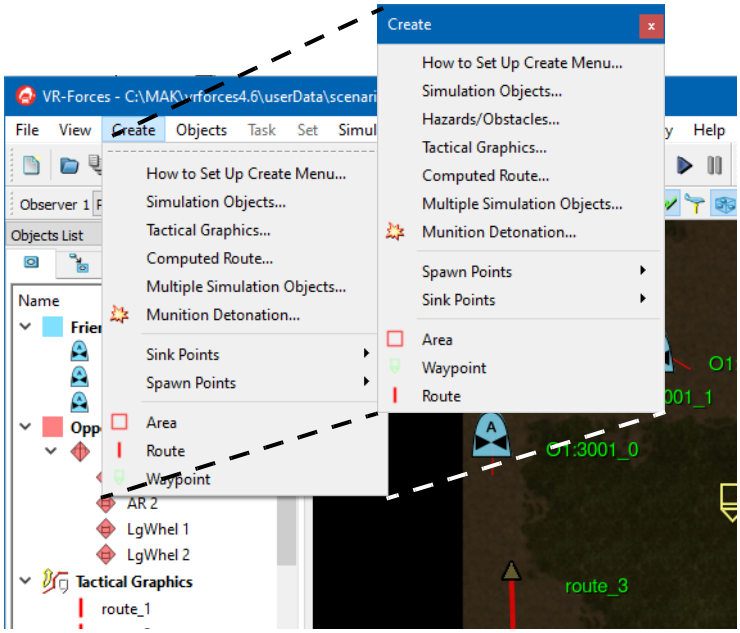


Figure 83-6. Tearing off a menu

83.4. Viewing the Window in Full-Screen Mode

You can display VR-Forces so that it covers the entire screen and does not show the title bar or border. This mode is intended for use in demonstrations. When menus and panels are hidden, you can still use all keyboard controls, and you can still click simulation objects in the window to attach the observer to them.

You can configure VR-Forces to start in full screen mode with the `--fullScreen` command-line option. If you are viewing a simulation that takes place primarily over the ocean, you may get better performance using the `--pseudoFullScreen` command-line option.

➤ To view VR-Forces in full-screen mode, choose **View** → **Show Full Screen**, or press **CTRL+ENTER**.

➤ To exit full screen mode, press **CTRL+ENTER**.

83.4.1. Hiding the Mouse in Full Screen Mode

Since full screen mode is often used only for display purposes, with no user interaction, the mouse is automatically hidden in this mode. As soon as you move the mouse, it is displayed. If you then stop using it, it is hidden when the threshold is reached.

To hide the mouse in full screen mode:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Mouse Settings page.

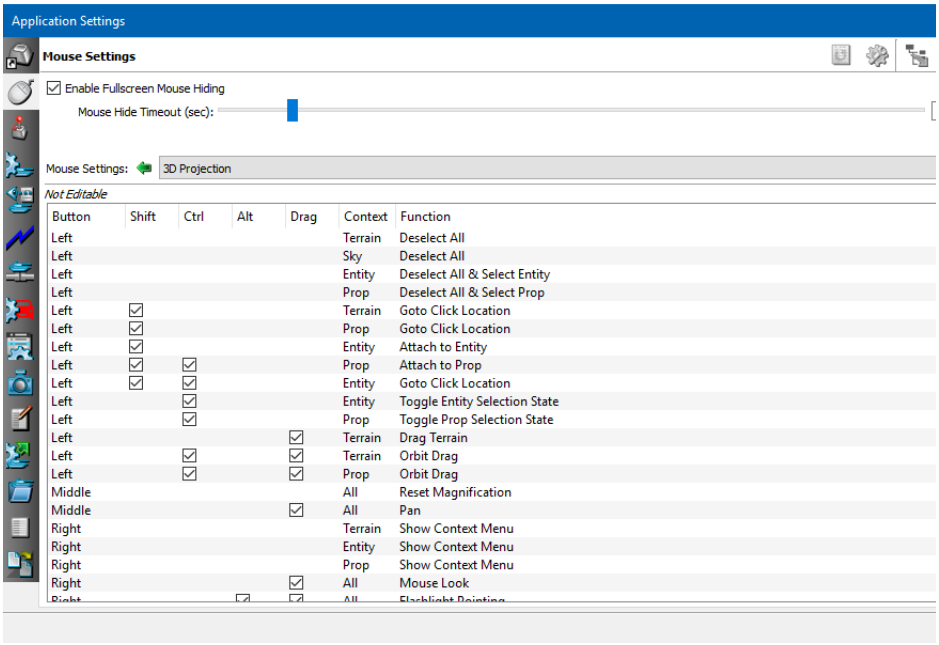


Figure 83-7. Mouse Settings page

3. Select the Enable Fullscreen Mouse Hiding check box.
4. Optionally, change the Mouse Hide Timeout value.

83.5. Using Context-Sensitive Menus

VR-Forces has context sensitive menus (also called shortcut menus, popup menus, or right-click menus) that you can display by clicking the right mouse button.

If you right-click a simulation object, tactical graphic, or other scene object, the menu displays the actions that you can take.

If you right-click the menu bar, you can select the commands available on the View menu.

83.6. Common VR-Forces Buttons

VR-Forces dialog boxes have a common set of buttons, as follows:

- Add an item (definition, model, and so on).
- Remove the selected item.
- Copy the selected item and add the copied version to the list.
- Select a filter of some type.
- Rename the selected item.
- Add from a list.
- Move up in the list.
- Move down in the list.
- Select the previous item.
- Select the next item.
- Add an attribute.
- Remove local attribute definition and use the parent's definition.
- Search.

83.7. Adding Text to the Title Bar

VR-Forces displays the name of the scenario in the title bar. You can display additional text of your choice if you want to. The text is independent of the scenario. It is an application-wide setting.

To add text to the title bar:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 83-8).

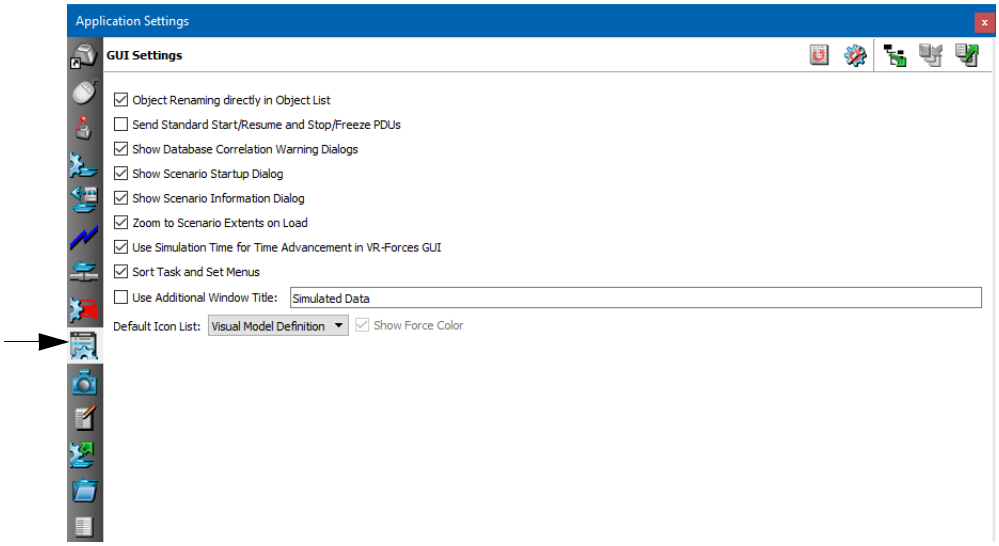


Figure 83-8. GUI Settings page

3. Select the Use Additional Window Title check box.
4. Type the text you want to display in the text box.

83.8. Specifying Display Units

You can specify the units used to display coordinate system, distance, altitude, and other measurements. Display units are configured by measurement type. This means that you can use different measurement units for different types of data. You can also specify the number of decimal places for measurement values. When you change a measurement display unit, the changes take effect immediately. They are saved and applied the next time you run VR-Forces.



The coordinate system information for the observer location in the Observer Information Panel and grid line coordinates, units, and precision are not affected by the settings described in this section.

The types of data and the display objects that they affect are as follows:

- Coordinate system
- Distance
- Altitude
- Velocity (speed)
- Acceleration
- Rotational velocity
- Orientation (heading).

Table 83-1 describes the coordinate systems that you can specify.

Table 83-1: Coordinate systems supported

Coordinate system	Description
Latitude/Longitude (degrees:minutes:seconds)	Location is displayed as two position fields and one altitude field. Displays coordinates in the latitude and longitude using the geodetic WGS84 coordinate system. Each angle is displayed in degrees:minutes:seconds with seconds displaying base 10 fractional seconds.
Latitude/Longitude (decimal radians)	Location is displayed as two position fields and one altitude field. Position coordinates in the latitude and longitude using the geodetic WGS84 coordinate system. Each angle is displayed in decimal radians.
Geocentric	Location is displayed as three position fields. Position coordinates are displayed in the geocentric coordinate system. The position fields are always in meters.
UTM	Location is displayed as two position fields and one altitude field. Position coordinates are displayed in the Universal Transverse Mercator system. The first position field displays the zone and the x location in meters. The second position field displays the y position in meters.

Table 83-1: Coordinate systems supported

Coordinate system	Description
MGRS	Location is displayed as two position fields and one altitude field. Position coordinates are displayed in the Military Grid Reference System. The first position field displays the zone. The second position field displays the grid location. The precision controls the number of digits used in the grid display.
Database	Location is displayed as three position fields. Location is displayed using VR-Forces's current internal cartesian database system. The position fields are displayed using the current distance units.

To specify measurement units:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Display Units Settings page (Figure 83-9).

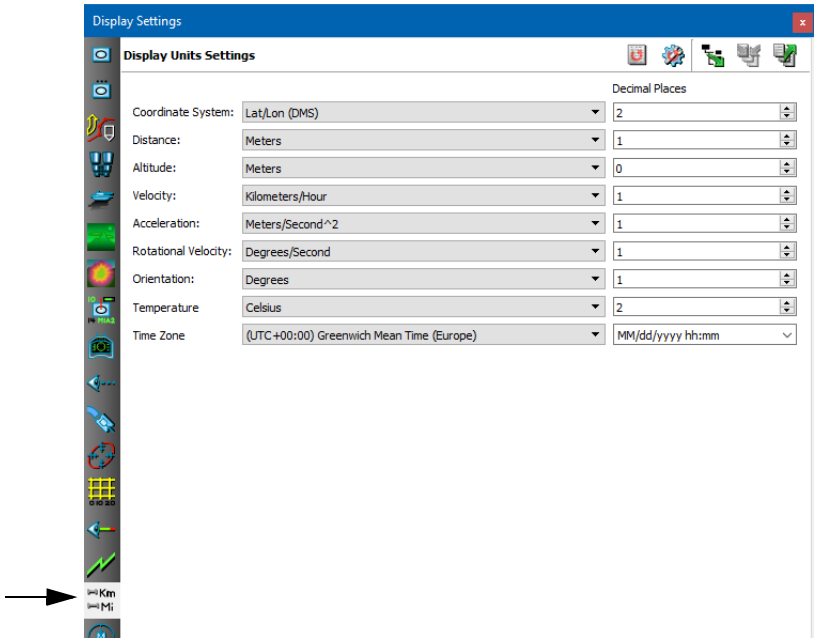


Figure 83-9. Display Units Settings page

3. Select the measurement unit you want for each option from its list.
4. Type a precision for each measurement unit in the corresponding box in the Decimal Places column.



84. Configuring Toolbars, Dialog Boxes, and Menus

This chapter explains how to specify which menus, menu options, toolbars, and dialog box pages get displayed in the graphical user interface (GUI).

Introduction 84-2
Configuring Menus and Menu Commands 84-2
Alphabetizing the Task and Set Menus 84-3
Creating New Menus 84-4
Configuring Dialog Box Pages 84-5
Configuring Context-Sensitive Menus 84-7
Editing the Task Menu 84-8
Editing the Set Menu 84-8
Configuring Toolbars 84-9
Creating Toolbars 84-9
Specifying an Icon for a Toolbar Item 84-10
Locking Main GUI Elements 84-10

84.1. Introduction

VR-Forces displays menus, menu commands, toolbars, dialog box pages, and shortcut menus based on the information in an XML configuration file. The default file is *./appData/settings/vrfGui/default_guiConfiguration.gui*. You can edit this file or create alternative GUI configuration files. VR-Forces includes an example of a simple GUI configuration file, *./appData/settings/vrfGui/default_guiSimpleScenarioPlayer.gui*, that shows how you might create a highly modified GUI. You can specify an alternative GUI configuration file with the `--gui` command-line option when you start VR-Forces.

The elements in the configuration file are fairly self-explanatory. This chapter provides a brief description of the file.



For those GUI elements that can be rearranged, such as toolbars, the *.gui* file provides the initial organization of GUI elements. If a user rearranges toolbars, the new layout is saved in the settings file and overrides the *.gui* file.

84.2. Configuring Menus and Menu Commands

Menus are configured in the `<mainmenuconfiguration>` element. Each menu is configured in a `<mainmenu>` element. Menu commands are configured in `<menuitem>` elements. The following example configures a menu bar with one menu that has two items, which are separated by a separator line.

```
<mainmenuconfiguration name="main_menu_configuration">
  <mainmenu menuname="DtFileMenu">
    <menuitem itemname="DtNewScenarioMenuItem"/>
    <separator separatorname="1"/>
    <menuitem itemname="DtExitItem"/>
  </mainmenu>
</mainmenuconfiguration>
```

The values for the `itemname` attributes must be valid IDs registered in VR-Forces code. *default_guiConfiguration.gui* has all of the valid IDs for menus and menu commands.

To remove a menu or menu command from the GUI, delete the applicable element from *default_guiConfiguration.gui* or create an alternative file that does not have the elements. To add menus or menu items, you must know the ID.

84.2.1. Alphabetizing the Task and Set Menus

You cannot edit the Task and Set menus in *default_guiConfiguration.gui*. By default, VR-Forces alphabetizes the commands on the Task and Set menus and their submenus. This alphabetization overrides the menu ordering specified for scripted tasks and sets in their metadata. It also overrides menu placement specified for tasks and sets through the API.

If you want to, you can disable alphabetization and the commands will be listed according to script metadata and C++ code.

To enable or disable alphabetization of Task and Set menu commands:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 84-1).

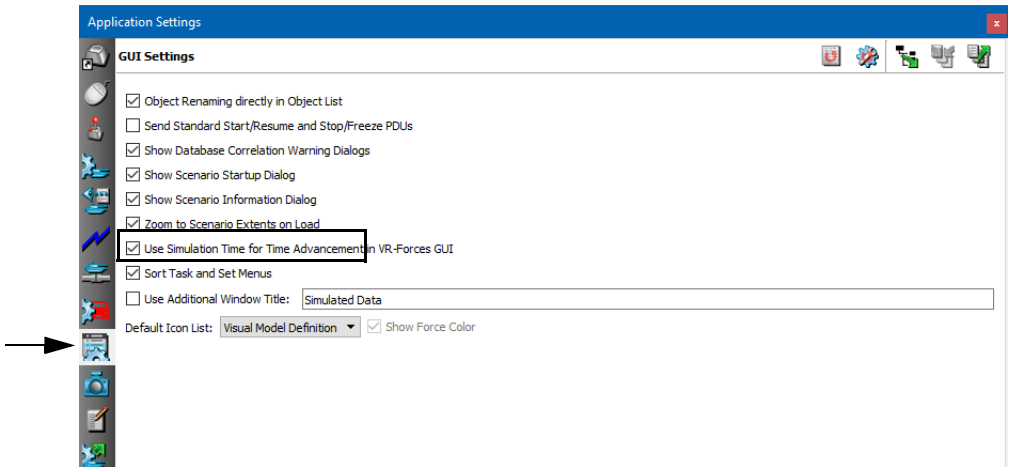


Figure 84-1. GUI Settings page

3. Select or clear the Sort Task and Set Menus check box.

84.2.2. Creating New Menus

You can create new menus and submenus. To do so use the same syntax as the existing menu structure in *./appData/settings/vrfGui/default_guiConfiguration.gui*. For top level menus, add a `mainmenu` element. For submenus, use a `menu` element. For the menu name, specify the string you want to appear in the GUI. The following example creates a menu called My File Menu at the top level. It has a submenu called Save Scenario Commands.

```
<mainmenuconfiguration name="main_menu_configuration">
  <mainmenu menuname="My File Menu">
    <menuitem itemname="DtNewScenarioMenuItem"/>
    <separator separatorname="1"/>
    <menu menuname="Save Scenario Commands">
      <menuitem itemname="DtSaveScenarioAsMenuItem"/>
      <menuitem itemname="DtSaveScenarioMenuItem"/>
    </menu>
    <menuitem itemname="DtExitItem"/>
  </mainmenu>
</mainmenuconfiguration>
```

84.3. Configuring Dialog Box Pages

Dialog boxes such as Display Settings and Terrain Settings have multiple pages for configuring different features. The GUI configuration file lets you disable or enable the pages on dialog boxes. The dialog boxes are configured as <pagecollection> elements inside the <pagecollections> element. The following example configures the Application Settings dialog box:

```
<pagecollections name="page_collections">
  <pagecollection pagecollectionname="DtApplicationSettings"
  title="Application Settings" location="8" visible="True" index="-1"
  show="DtKeyMappingEditorPage" closable="True" floatable="True"
  movable="True">
    <pageitem pageitemname="DtKeyMappingEditorPage"/>
    <pageitem pageitemname="DtMouseMappingEditorPage"/>
    <pageitem pageitemname="DtJoystickConfigurationPage"/>
. . .
    <pageitem pageitemname="DtScriptedTaskOptionsPage"/>
    <pageitem pageitemname="DtEntityBehaviorOptionsPage"/>
  </pagecollection>
</pagecollections>
```

In addition to settings dialog boxes, you can configure the pages on information dialog boxes, and the specific data on the pages. These are configured in <informationpagecollection> elements. The following example is part of the Information dialog box. The page is broken up into sections. The first section displays summary data. The second section has the individual pages and specifies the information on each page.

```
<informationpagecollections name="information_page_collections">
  <pagecollection pagecollectionname="DtEntityInformation" title="Entity
  Information" location="9" visible="True" index="-1"
  show="DtTaskStatePage" closable="True" floatable="True" movable="True">
    <section sectionname="DtEntityCompactData"
  maximumHeightIsMinimumHeight="true">
      <informationitem informationitemname="DtEntityIcon" row="0"
  col="0" rowspan="2" width="50"/>
      <informationitem informationitemname="DtStateMarkingText" row="0"
  col="1"/>
      <pageitem pageitemname="DtInformationIconBar" row="1" col="1"
  width="20">
        <informationitem informationitemname="DtEmbarkedOnItem"/>
      </pageitem>
      <informationitem informationitemname="DtCreatePaletteName" row="0"
  col="2"/>
      <pageitem pageitemname="DtSensorInformationPage" row="1" col="2"/>
      <informationitem informationitemname="DtEntitySpeed" row="0"
  col="3" width="128"/>
      <informationitem informationitemname="DtEntityAltitude" row="1"
  col="3" width="128"/>
      <pageitem pageitemname="DtTaskStatePage" row="0" col="4"
  width="128"/>
      <informationitem
  informationitemname="DtObjectConsoleInformationStatusItem" row="1"
  col="4"/>
    </section>
    <section sectionname="DtEntityExpandedData" collapsible="True"
  tabbed="small" collapsed="True" title="Detailed Information">
      <pageitem pageitemname="DtTaskStatePage"/>
```

```
        <pageitem pageitemname="DtEntityInformationPage">
          <informationitem informationitemname="DtEntityLocation"/>
          <informationitem informationitemname="DtEntitySpeed"/>
          <informationitem informationitemname="DtEntityAltitude"/>
          <informationitem informationitemname="DtEntityVelocity"/>
          <informationitem informationitemname="DtEntityAcceleration"/>
          ...
          <informationitem informationitemname="DtNavigationAreaPorts"/>
          <informationitem
    informationitemname="DtEntityStateDRAlgorithm"/>
        </pageitem>
        <pageitem pageitemname="DtEntityAttributesPage"/>
        <pageitem pageitemname="DtResourceInformationPage"/>
        ...
      </section>
      <section sectionname="DtObjectConsoleSection" title="Object Console"
    collapsible="true" collapsed="true"
    maximumHeightIsMinimumHeight="true">
        <pageitem pageitemname="DtObjectConsoleInformationWidget"/>
      </section>
    </pagecollection>
  ...
   </informationpagecollections>
```

Notice that by default, the DtTaskStatePage does not have any `<informationitem>` elements. The information on that page is generated dynamically. There are no static fields.

84.4. Configuring Context-Sensitive Menus

The Objects menu on the main menu bar and the context-sensitive, or popup, menus vary depending on which objects are selected. These menus are configured as a set of <rightclickmenu> elements. There are multiple versions of these elements to account for the different contexts in which a menu might be displayed, for example, one unit, multiple single entities, a mix of entities and control objects. The following example is part of the context-sensitive menu for multiple entities:

```
<rightclickmenu rightclickmenuname="Multiple_Entity">
  <popupmenu menuname="DtVrfRightClickContextMenu">
    <menu menuname="DtRightClickTaskMenu"/>
    <menu menuname="DtRightClickSetMenu"/>
    <menuitem itemname="DtTopLevelRightClickScriptedTasksSets"/>
    <menuitem itemname="DtVrfEntityPlanAction"/>
    <menu menuname="Manage Plan">
      <menuitem itemname="DtVrfEntityAbandonPlanAction"/>
      <menuitem itemname="DtVrfEntityRestartPlanAction"/>
      <menuitem itemname="DtVrfEntitySkipTaskAction"/>
    </menu>
      <menuitem itemname="DtVrfManageReactiveTasksAction"/>
      <menuitem itemname="DtVrfVrlinkObjectInformationItem"/>
      <separator separatorname="3"/>
      <menuitem itemname="DtVrfEditObjectAction"/>
      <menuitem itemname="DtVrfMoveObjectAction"/>
      <menuitem itemname="DtVrfCopyAction"/>
      <menuitem itemname="DtVrfDeleteAction"/>
      <separator separatorname="16"/>
    <menu menuname="Embarkation">
      <menuitem itemname="DtVrfEmbarkOnAction"/>
      <menuitem itemname="DtVrfDisembarkAction"/>
      <menuitem itemname="DtVrfDisembarkAllAction"/>
    </menu>
    <menu menuname="DtUnitDisplayEntityMenu">
      <menuitem itemname="DtVrfAggregateAsAction"/>
      <separator separatorname="AggSeparator1"/>
      <menuitem itemname="DtExpandAggregateMenuItem"/>
      <menuitem itemname="DtExpandAllAggregateMenuItem"/>
      <menuitem itemname="DtCollapseAggregateMenuItem"/>
      <menuitem itemname="DtCollapseAllAggregateMenuItem"/>
    </menu>
    <menu menuname="Decorations">
      <menuitem itemname="DtObjectSettingsAction"/>
    </menu>
  </popupmenu>
</rightclickmenu>
```

Notice that the entries for the Task menu (DtRightClickTaskMenu) and the Set menu (DtRightClickSetMenu) do not have any <menuitem> elements. That is because these menus are themselves context-sensitive and get built dynamically.

84.4.1. Editing the Task Menu

The Task menu is a context sensitive menu built based on the platform type, controllers, scripts, and systems for a simulation object. Therefore, there is no all inclusive configuration file that you can edit to add and remove tasks. However, the file *./data/simulationModelSets/<sms>/gui/initialTaskMenuConfiguration.xml* is an SMS-specific set of tasks that get initialized when you load a scenario. If you remove a task from the list in this file, it will not be included on the Task menu. Removing a task in this way affects all simulation objects. If you edit this file, be sure to change the value for the `count` element to match your changes.

84.4.2. Editing the Set Menu

The Set menu is a context-sensitive menu built dynamically based on the controllers and scripts that define set data requests. While some set data requests are specific to particular platforms or object types, some are generically available to any platform that has a `set-data` controller in its platform file. This can result in some set data requests being available for objects for which they do not have much meaning. For example, you can set the ordered speed for a lamp post or a building.

You can cause VR-Forces to ignore set data requests when it builds the Set menu by adding an **ignore-sets** parameter to a `set-data` controller in an OPE file. The following code shows the set-data controller in *AirBase.ope* in the AggregateLevel SMS:

```
(set-data
  (component-descriptor-type "set-controller-descriptor")
  (component-type  "cultural-feature-set-controller")
  (min-tick-period  -1.000000)
  (min-tick-period-variance  -1.000000)
  (tick-period-uses-real-time  False)
  (process-state-repository-name  "")
  (process-state-repository-type  "")
  (is-enabled  True)
  (debug-detail  False)
  (create-component  True)
  (ignore-sets "set-speed" "set-capability" "set-engagement-rules"
    "set-engagement-zones" "set-scan-sector" "set-current-speed"
    "set-altitude")
)
```

This method of editing the Set menu only applies to set data requests that are registered for in code, not scripted sets. The strings that you can add to **ignore-sets** are in *./include/vrftasks/radioMessageTypes.h*.

You can remove some set data requests from the menu for all simulation objects by editing *./data/simulationModelSets/<sms>/gui/initialSetMenuConfiguration.xml*. Be sure to change the value for the `count` element to match your changes.

84.5. Configuring Toolbars

You can configure use of the various toolbars provided with VR-Forces. For each displayed toolbar, you can configure which row it is in and which command icons are used. The following example shows that the Scenario toolbar is in the top row:

```
<toolbarconfiguration name="toolbar_configuration">
  <toolbarrow toolbarname="Top_Row">
    <toolbar toolbarname="DtScenarioToolbar" visible="True"
      floating="False">
      <item itemname="DtNewScenarioMenuItem"/>
      <separator separatorname="DtScenarioToolbar_1"/>
      <item itemname="DtLoadScenarioMenuItem"/>
      <item itemname="DtLoadBatchScenarioMenuItem"/>
      <separator separatorname="DtScenarioToolbar_2"/>
      <item itemname="DtSaveScenarioMenuItem"/>
      <separator separatorname="DtScenarioToolbar_3"/>
      <item itemname="DtCloseScenarioMenuItem"/>
    </toolbar>
  </toolbarrow>
</toolbarconfiguration>
```

84.5.1. Creating Toolbars

Since toolbars are just buttons that reference existing menu options, you can create your own toolbars as easily as you add, delete, or edit the toolbars provided with VR-Forces. Simply add a `toolbar` element in one of the `toolbarrow` elements.

If you want to add tasks or sets to the Task or Set toolbars, or create a new toolbar, specify the `itemname` using the task or set ID as listed in *./include/vrfGuiConstants/vrfMenuIds.h*. If you want to add a scripted task to a toolbar, the `itemname` is its script ID plus "ToolbarItem". For example, if you add the Place IED scripted task to the Task toolbar, it would look like this:

```
<toolbar toolbarname="DtTaskToolbar" visible="True" floating="False">
  <item itemname="DtVrfTaskMoveToActionToolbarItem"/>
  <item itemname="DtVrfTaskMoveToLocationActionToolbarItem"/>
  <item itemname="DtVrfTaskMoveAlongRouteActionToolbarItem"/>
  <item itemname="DtVrfTaskMoveToAltitudeActionToolbarItem"/>
  <item itemname="DtVrfTaskPatrolObjectsActionToolbarItem"/>
  <item itemname="DtVrfTaskPatrolRouteActionToolbarItem"/>
  <item itemname="DtVrfTaskEmbarkActionToolbarItem"/>
  <item itemname="DtVrfTaskDisembarkActionToolbarItem"/>
  <item itemname="DtVrfTaskFlyAltitudeActionToolbarItem"/>
  <item itemname="DtVrfTaskFlyHeadingActionToolbarItem"/>
  <item itemname="DtVrfTaskFlyHeadingAndAltitudeActionToolbarItem"/>
  <item itemname="DtVrfTaskFireAtTargetActionToolbarItem"/>
  <item itemname="Place_IEDToolbarItem"/>
</toolbar>
```

84.5.2. Specifying an Icon for a Toolbar Item

If you add an item to a toolbar, you can associate an icon with the toolbar button. To associate an icon with a scripted task, specify it in the Scripts dialog box. For details, please see "Specifying a Menu Icon," on page 31-12. For tasks or sets, create an SVG file and put it in *./data/Icons*. The file should use the menu ID of the task or set (from *./include/vrfGuiConstants/vrfMenuIds.h*) with the DtVrf text and Action text stripped from it. For example, the icon for DtVrfTaskMoveToAltitudeAction is *TaskMoveToAltitude.svg*. The icon for DtVrfSetHeadingAction is *SetHeading.svg*.

84.6. Locking Main GUI Elements

VR-Forces plug-ins can modify menus, dialog boxes, and other GUI features. You can lock GUI settings for first level elements (elements under the root element). This feature is used in *./appData/settings/vrfGui/default_guiSimpleScenarioPlayer.gui*. The syntax would be as follows:

```
<mainmenuconfiguration name="main_menu_configuration" locked="True">
  ...
</mainmenuconfiguration>
```

Given this configuration, none of the menus on the main menu could be modified.



85. Creating and Editing Key Mappings

This chapter explains how to edit key mappings and create new key mappings.

Introduction 85-2
The Key Mapping Editor 85-2
Binary Key Mappings 85-3
Editing a Key Map 85-3
Adding a Key Mapping 85-3
Changing a Key Mapping 85-5
Deleting a Key Mapping 85-5
Using Combined Key Mappings 85-6
Filtering the Function List 85-7
Creating a Key Map 85-7
Deleting a Key Map 85-7

85.1. Introduction

When you move the observer, you are actually executing functions that control the observer's movement. For example, the Move Observer Forward function moves the observer forward in observer coordinates. The Move Level Observer Forward function moves the observer forward in level observer coordinates.

The various movement functions are mapped to the keys on the keyboard and to mouse buttons and the mouse wheel.

VR-Forces comes with key mappings that we think will meet the needs of most VR-Forces users. However, you can edit the default keyboard mappings and create new mappings of your own with the Key Mapping Editor.

85.2. The Key Mapping Editor

The Key Mapping Editor (Figure 85-1) lists all of the functions that you can control from the keyboard and the keys to which they are mapped. Some functions are mapped to more than one key or key combination. A key can be mapped to more than one function (combined mappings). You can:

- Edit a function's key mappings.
- Add new mappings.
- Delete mappings.

Changes are saved automatically.

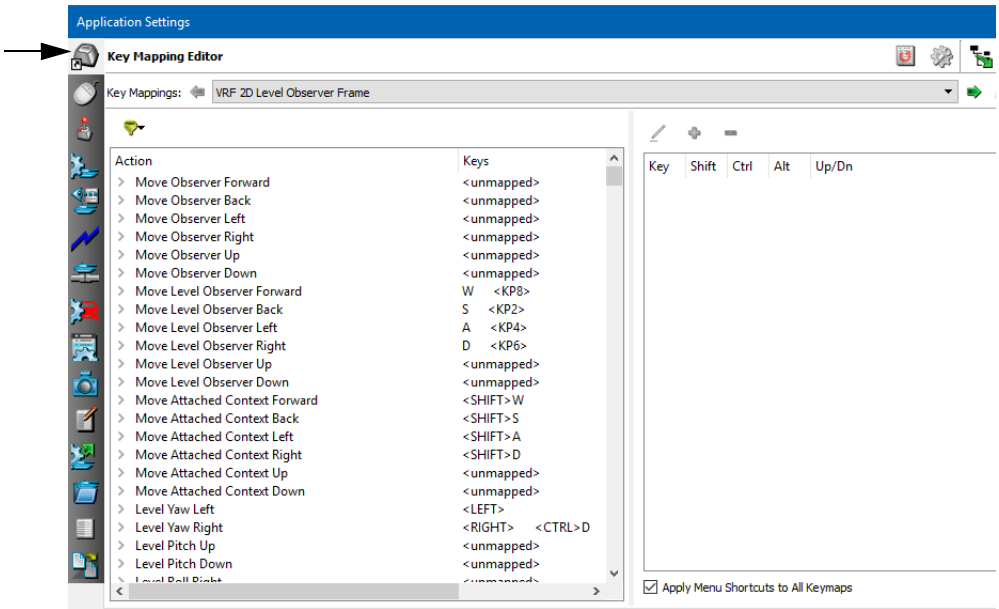


Figure 85-1. Key Mapping Editor

85.2.1. Binary Key Mappings

Many navigation functions require you to press and hold a key. For example, you press and hold **w** to move the observer forward. (By contrast, some keyboard actions are toggles or unary functions, such as pressing period to attach to the next simulation object.) For these actions, VR-Forces needs to keep track of when the key is pressed down (start the action) and when it is released (stop the action). To a human user, a function like Move Observer Forward is experienced as one action. But the computer experiences it as two actions. Therefore, key mappings must take this into account. The Key Mapping Editor shows the key mappings for these functions as paired functions that are subordinate (children) to the primary function (the parent), as shown in the Move Observer Up entry in Figure 85-1.

85.3. Editing a Key Map

The Key Mapping Editor lists all of the VR-Forces functions that you can control from the keyboard.

To edit a key map:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Key Mapping Editor page (Figure 85-1).
3. Select the function whose key mappings you want to edit.
4. Edit the key mappings as described in:
   - "Adding a Key Mapping," on page 85-3.
   - "Changing a Key Mapping," on page 85-5.
   - "Deleting a Key Mapping," on page 85-5.
5. Optionally, export the key map to a file.
6. Click OK.

85.3.1. Adding a Key Mapping

If a function is not mapped to a key, it is marked `<unmapped>`. You can add a mapping to unmapped functions and you can add additional key mappings to functions that are already mapped. If you add a key mapping to a binary function, the Key Mapping Editor automatically applies it to the child (UP and DN) functions. You can add key mappings to the child functions of binary mappings. When you add a key mapping to a child function, it is not applied to its paired function or the parent function.



In some cases, an unmapped function is unmapped because it is not meaningful in the observer frame for the key mapping. For example, the level observer movement functions are not meaningful in the observer frame key mappings.

To add a key mapping:

1. Optionally, filter the function list. For details, please see "Filtering the Function List," on page 85-7.
2. Select the function to which you want to add a key mapping.
3. Click the Add button (+) above the Key Map window (Figure 85-2). The Key Entry dialog box opens (Figure 85-3).

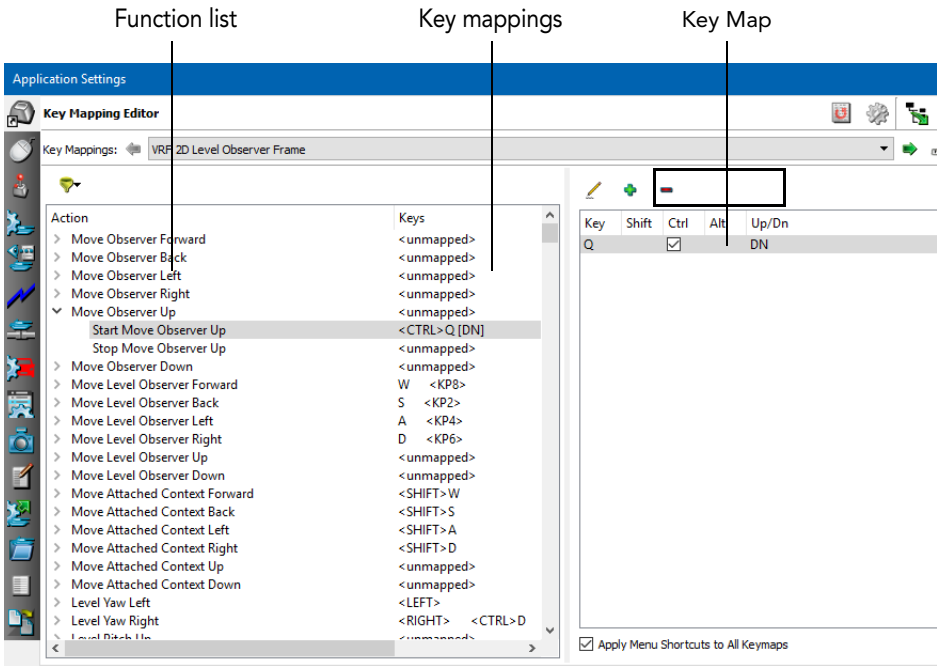


Figure 85-2. Key Mapping Editor

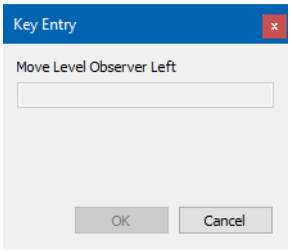


Figure 85-3. Key Entry dialog box

4. On the keyboard, press the key or key combination (such as **CTRL**+key or **SHIFT**+key) that you want to map to this function.

5. Click OK. (If you choose a key combination that is already in use, VR-Forces notifies you and asks how to proceed. For details about duplicate key mappings, please see "Using Combined Key Mappings," on page 85-6.) The mapping is added to the function and is listed in the Keys column of the function list and in the Key Map window.

85.3.2. Changing a Key Mapping

If a key mapping is assigned to a binary function, the key is automatically assigned to the child functions. You cannot edit the key mappings for the child functions. You can only change the parent's key mapping. However, if you add a key mapping to a child function, you can edit it.

To change a key mapping:

1. Select the function whose key mapping you want to change. Its key mappings get listed in the Key Map window (Figure 85-2).
2. In the Key Map window, select the key mapping that you want to change.
3. Click the Edit Key Mapping button (). The Key Entry dialog box opens (Figure 85-4). It shows the current key mapping.

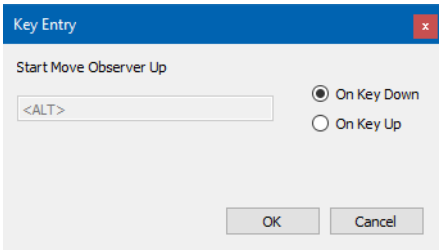


Figure 85-4. Key Entry dialog box

4. On the keyboard, press the key or key combination (such as **CTRL**+key or **SHIFT**+key) that you want to map to this function.
5. Click OK. The mapping is changed.

85.3.3. Deleting a Key Mapping

To delete a key mapping:

1. Select the function whose key mapping you want to delete. Its key mappings get listed in the Key Map window (Figure 85-2).
2. In the Key Map window, select the key mapping that you want to delete.
3. Click the Delete button () above the Key Map window. The mapping is deleted. The function list is updated.

85.4. Using Combined Key Mappings

You can assign a key combination to more than one function. This allows you to combine the behaviors of two or more functions into one key combination. For example, the **Z** key detaches the observer from the attached simulation object. The **Spacebar** resets the observer view to the default view. Suppose that whenever you detach from a simulation object, you want the observer to return to the default view. You could map the Reset function to **Z**. **Z** is now mapped to two functions. When you press **Z**, VR-Forces detaches from the attached simulation object and moves the observer to the default view.

To manage combined key mappings:

1. Add the appropriate key mapping to a function.
2. Click OK. The Key Already Mapped dialog box opens (Figure 85-5). It lists the functions that already use this key mapping.

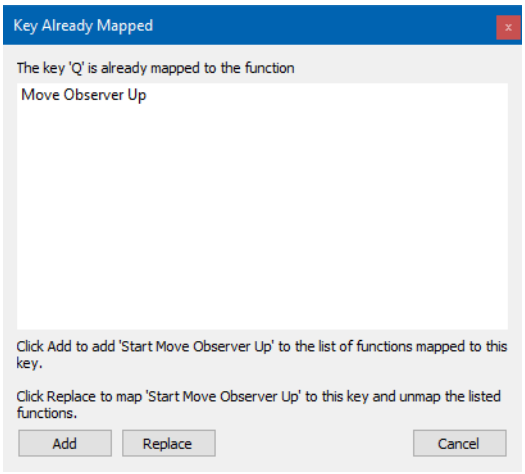


Figure 85-5. Key Already Mapped dialog box

3. If the functions listed are the ones with which you want to share this key mapping, click Add. The key mapping is added to the function.
4. If you do not want to share this key mapping with other functions and want to use it for the function that you are editing, click Replace. The key mapping is applied to the function you are editing and removed from the functions listed in the dialog box.
5. If you do not want to remove the key mapping from existing functions and do not want to share it with the other functions, click Cancel. Presumably you wanted a unique key mapping for the function you are editing and did not realize you chose one that was already in use. Choose a different key mapping for this function.

85.5. Filtering the Function List

You can filter the function list in the Key Mapping Editor. Functions are grouped thematically.

➤ To filter the function list, click the Filter button (▾) and select an option on the list.

85.6. Creating a Key Map

To create a key map:

1. Choose **Settings** → **Application**.The Application Settings dialog box opens.
2. Select the Key Mapping Editor page (Figure 85-1).
3. Click the Add button (+). The Create New Key Map dialog box opens.
4. Type a name for the new key map.
5. Click OK.
6. Add key mappings for the movement actions as described in "Adding a Key Mapping," on page 85-3.

85.7. Deleting a Key Map

To delete a key map:

1. Choose **Settings** → **Application**.The Application Settings dialog box opens.
2. Select the Key Mapping Editor page (Figure 85-1).
3. In the key mapping list, select the key mapping that you want to delete.
4. Click the Delete button (−).
5. Confirm that you want to delete the selected key map.



XIV. Configuring Visual Models in the Visual Models Editor

XIV.1. Managing Model and Element Definitions

When VR-Forces discovers a simulation object (entity or unit) on the network, it must decide how to represent it in the GUI. This is accomplished by mapping 3D models and 2D icons to object type enumerations.

This section explains how to use the Visual Model Editors dialog box to create and edit schemas, model definitions, and element definitions. This process can be rather complex. If you are customizing the visual characteristics of the simulation objects provided with VR-Forces or adding new object types that are significantly different from those provided with VR-Forces, you may need to create new model definitions or edit the existing ones. However, if you just need to add new object types that are similar to existing ones, it is strongly recommended that you do so in the Simulation Object Editor.

When you create a simulation object in the Simulation Object Editor, it adds new model definitions and object type mappings automatically. You will not need to do anything in the Visual Models Editor to configure the new simulation object.

Model definitions and element definitions (visual definition files) are SMS-specific. The visual definition files for the SMSs provided with VR-Forces are stored in subdirectories of *./appData/definitions*. If you create a new SMS, the visual definition files are stored as part of the SMS, in *./data/simulationModelSets/<sms>/gui/visuals*. You can copy the visual definition files for the provided SMSs into the SMS hierarchy if you want to make it easier to copy an SMS to another VR-Forces installation.

When you open the Visual Models Editor, it loads the visual model definitions for the SMS used by the loaded scenario. If no scenario is loaded, it defaults to *EntityLevel.sms*. For details about how to edit visual models in the Simulation Object Editor, please see "Editing a Simulation Object's Visual Model," on page 68-13

XIV.2. Introduction to Object Modeling

The objects that VR-Forces can display that are not part of the terrain, such as simulation objects, tactical graphics, detonations, and cockpit displays, are generically called scene elements. Scene elements are defined in element definitions. When VR-Forces runs a scenario, it publishes information about the objects it is simulating. It may also receive object information over the network from other simulation participants. The objects are identified using object type enumerations. VR-Forces maps these object types to element definitions. (For more information about object type enumerations and mapping them to element definitions, please see "Introduction to Object Type Mapping," on page 89-2.)

An element definition is a set of visual definitions that tell VR-Forces how to render (display) the object. Visual definitions specify the 3D model, 2D icon, or graphic, and the visual effects that the object supports, such as trailing effects, height-above-terrain lines, track histories, and so on. A scene element can have up to three different element definitions – one for each model set (3D, 3D Colorized, 2D Icon). VR-Forces chooses the correct element definition to use based on the model set being used by the observer. (For more information about model sets, please see "Model Sets," on page 85-5.)

Element definitions are created and edited in the Visual Model Editors dialog box. In addition to letting you manage element definitions, this dialog box lets you create and manage model definitions and schemas. Model definitions specify values for some basic object parameters. Most importantly for most VR-Forces users, model definitions specify the 3D model to use for element definitions in the 3D Models model set. What this means is that if you want to add a new model to VR-Forces, you will specify the model file, such as an OpenFlight file, in a model definition and then specify that model definition in the element definition for the simulation object you want to add.



The chapters that discuss how to configure visual models do so in the context of using the Visual Models Editor dialog box. We strongly recommend that whenever possible you use the Simulation Object Editor to add and configure visual models. The only time that you should have to use the Visual Models Editor dialog box is if you need to add a new schema or edit model attributes that are not supported by the Simulation Object Editor. For details about how to use the Simulation Object Editor, please see Section XII, *Creating and Editing Simulation Models* and the chapters in that section.

Schemas define the most basic scene elements, such as an entity, SpeedTree tree, or line. Most VR-Forces users will never have to create or edit a schema. However, if you create a new model definition, you must make sure it uses the correct schema and that it includes all required parameters.

Figure 85-1 illustrates these relationships using an M1A2 entity. VR-Forces receives the object type 1:1:225:1:1:3:-1 over the network (callout 1). In the Simulation Object Type Mappings dialog box, this object type is mapped to the M1A2 Abrams entity definition (2). (An entity definition is a specific type of element definition.) The M1A2 Abrams entity definition (in the Visual Model Editors dialog box) specifies all of the visual definitions for this entity (3). The Main Model visual definition has a modeldefinition parameter. The value for this parameter is TrackedM1A2DESERT. The TrackedM1A2DESERT model definition has a filename parameter that specifies *M1A2.medf* as the model to use for entities that use this model definition (4). The TrackedM1A2DESERT model definition has a filename parameter because it uses the Entity schema and the Entity schema requires a filename (5).

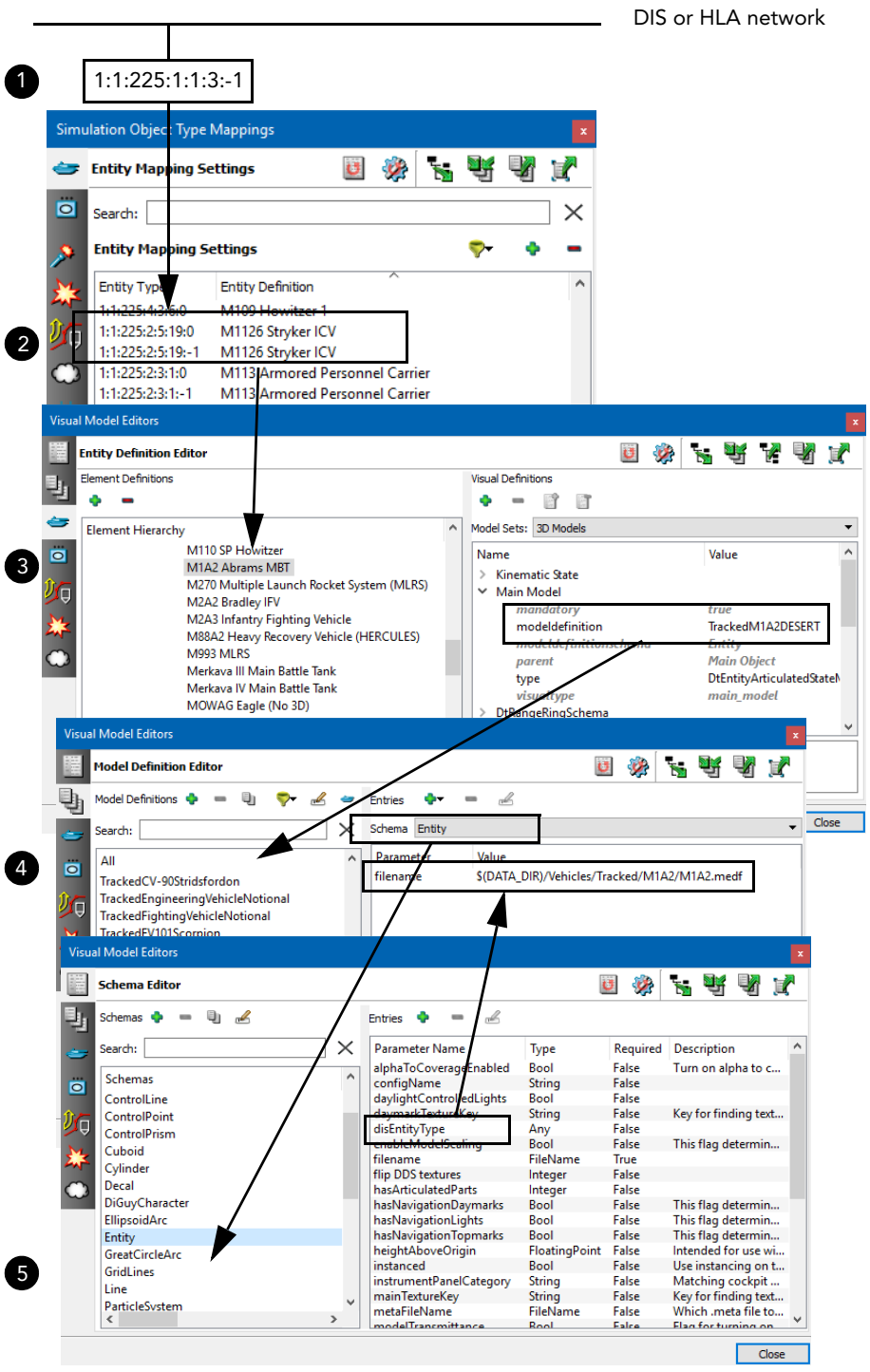


Figure 85-1. Element definitions and simulation object type mapping



When you add simulation objects and other scene elements in the Simulation Object Editor, it creates model definitions and handles all of the mappings described in the previous paragraphs and the illustration. Although this manual explains all of the details of the Visual Model Editors dialog box and Simulation Object Type Mappings dialog box, using the Simulation Object Editor to add simulation objects is the preferred method.

XIV.2.1. Model Formats for 3D Entities, Objects, and Effects

VR-Forces includes 3D models for vehicles, lifeforms, environmental objects, and effects. The models are in a compressed, encrypted format (extension *.medf*).



VR-Forces includes a utility, the MEDF Tool, that lets you encrypt files in MEDF format. You cannot use it to decrypt encrypted files. For more information, please see "Compressing Model Files," on page 90-35.

You can provide your own models if they are in OpenFlight format.

XIV.2.2. Model Sets

The model or icon that is used for an object is determined by its element definition. For convenience, element definitions are organized into model sets. For example, all of the element definitions that specify 3D models are in the 3D Models model set, and so on. As a result, when you choose the 3D Models model set for an observer mode, all of the entities use a 3D model. (The model set is an observer-specific setting.)

VR-Forces includes the following types of object models and icons:

- 3D Models. Realistic 3D models. This is the default model set for the Stealth observer mode.
- 3D Colorized Models. 3D colorized models are cartoon-like models that create an exaggerated notional view of the world. This is the default model set for XR observer mode.
- 2D Icons. 2D MIL-STD 2525B font-based icons. This is the default model set for the Plan View observer mode. You can also configure VR-Forces to use 2525a images or any arbitrary images for 2D icons.



The model sets are fixed. You cannot add model sets.

For information about changing model sets, please see "Changing the Model Set," on page 50-11.



86. Schemas

In Section XIV, "Configuring Visual Models in the Visual Models Editor" we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit model definitions and model schemas.

Creating and Editing Schemas 86-2
Creating Schemas 86-3
Deleting Schemas 86-4
Copying Schemas 86-4
Adding a Parameter to a Schema 86-5
Deleting a Parameter from a Schema 86-6
Editing a Schema Parameter 86-6

86.1. Creating and Editing Schemas

The elements that make up a scene have parameters that help define them, for example a model filename. The set of parameters for a particular scene element is a schema. VR-Forces uses schemas to validate element definitions. You can create, copy, and delete schemas. For any schema, you can add and delete parameters and edit parameter values.



Changes to schemas are saved automatically. For more information, please see "Managing VR-Forces Settings," on page 3-13.

Model definition schemas define the elements that can be displayed in a scene. The model definition schemas provided with VR-Forces are:

- 3DText
- Animation
- Channel Compass
- Cockpit Display
- ControlLine
- ControlPoint
- ControlPrism
- Cuboid
- Cylinder
- Decal
- DiGuyCharacter
- EllipsoidArc
- Entity
- GreatCircleArc
- GridLines
- Line
- ParticleSystem
- Ribbon
- RotorWash
- SpeedTree
- TacticalSmoke
- Terrain
- TidalStreamWake
- Tree
- Wake
- Watercraft
- WaterImpact
- WidgetTheme.

86.2. Creating Schemas

To create a schema:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 86-1).

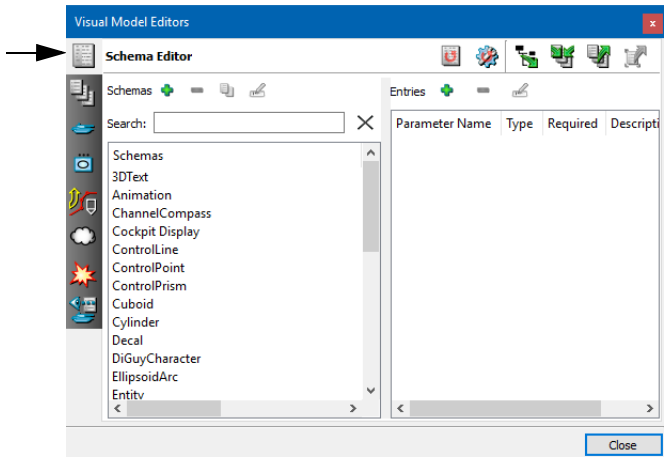


Figure 86-1. Schema Editor page

3. Click the Add button (+) next to the Schemas label. The Create Instance dialog box opens (Figure 86-2).

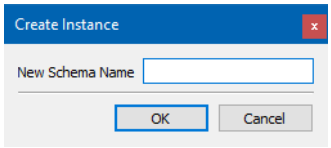


Figure 86-2. Create Instance dialog box

4. Type a name for the new schema.
5. Click OK.
6. Add parameters as described in "Adding a Parameter to a Schema," on page 86-5.

86.3. Deleting Schemas



If a schema is used by a model definition, you cannot delete it. The Delete Schema button will be unavailable.

To delete a schema:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 86-1).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema you want to delete.
5. Click the Delete button () next to the Schemas label. You are prompted to confirm the deletion.
6. Click Yes. The schema is removed from the list.

86.4. Copying Schemas

If you want a new schema that is a variant of an existing one, it can be easier to copy the existing schema and edit it than it is to create a new schema and add all of the parameters.

To copy a schema:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 86-1).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. Select the schema that you want to copy.
5. Click the Duplicate button (). The Duplicate Schema dialog box opens.
6. Type a name for the copied version of the schema.
7. Click OK.
8. The new schema is added to the list.
9. Edit the parameters as described in "Editing a Schema Parameter," on page 86-6.

86.5. Adding a Parameter to a Schema

To add a parameter to a schema:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page.
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema to which you want to add a parameter.
5. Click the Add button (+) next to the Entries label. A parameter called “key” is added to the parameter list (Figure 86-3).

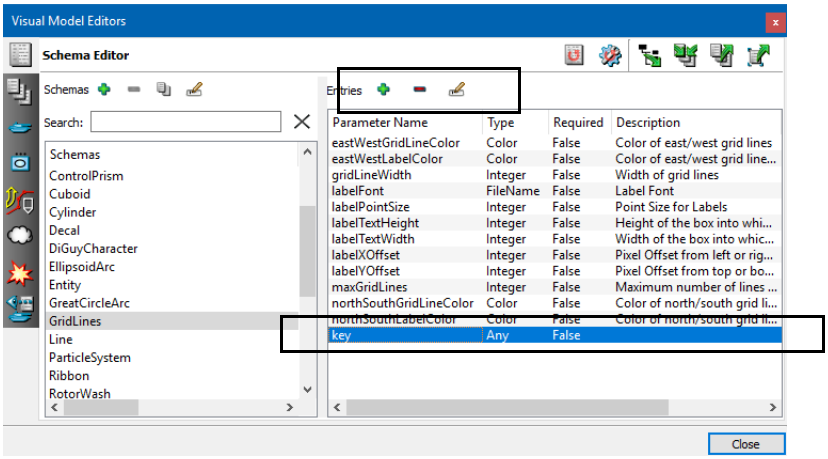


Figure 86-3. New schema parameter

6. Double-click “key”. The Rename Parameter dialog box opens.
7. Type a name for the parameter.
8. If the parameter has a specific type, click the value in the Type column and edit it.
9. If the parameter is required, click the value in the Required column and change it to True.
10. Optionally, add a description. The description provides context sensitive help on the element definition pages. To add a description:
    a. Click the empty field in the Description column. The Edit Text dialog box opens.
    b. Type the descriptive text.
    c. Click OK.

86.6. Deleting a Parameter from a Schema

> **!** If you delete a parameter from a schema, any element definitions that use the schema may become invalid. Invalid element definitions are shown in red in the list of definitions.

To delete a parameter from a schema:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 86-1).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema from which you want to delete a parameter.
5. Select the parameter that you want to delete.
6. Click the Delete button (−) next to the Entries label.

86.7. Editing a Schema Parameter

To edit a parameter for a schema:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 86-1).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema for which you want to edit a parameter.
5. Click the parameter attribute that you want to edit. The attribute becomes editable.
6. Type the new attribute value.



87. Model Definitions

In XIV, *Configuring Visual Models in the Visual Models Editor* we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit model definitions.

Creating and Editing Model Definitions 87-2
Work Flow Issues for Model Definitions 87-3
Creating a Model Definition 87-5
Deleting a Model Definition 87-6
Copying a Model Definition 87-7
Filtering the List of Model Definitions 87-8
Editing a Model Definition 87-8
- Adding Parameters to a Model Definition 87-8
- Editing a Model Definition Parameter 87-10
- Deleting Parameters from a Model Definition 87-10

Jumping to an Element Definition 87-11
Saving Model Definitions 87-11
Importing Model Definitions 87-12
- Importing Model Definitions from the Command Line 87-12

Exporting Model Definitions 87-13
- Exporting Selected Model Definitions 87-13

87.1. Creating and Editing Model Definitions

Many of the visual definitions in an element definition have a model definition parameter. A model definition specifies a set of parameters that are appropriate to the object type. (The parameters are determined by the model definition's schema. For information about model schemas, please see "Creating and Editing Schemas," on page 86-2.) For most VR-Forces users, the most important thing to know about model definitions is that for 3D models, the model definition specifies the 3D model, such as an OpenFlight file, that is used to display an entity. If you want to add new models to VR-Forces, you will probably have to create a new model definition for each one.

It is strongly recommended that you use the Simulation Object Editor to edit visual models, which includes the model definition.

VR-Forces comes with an extensive set of model definitions that are mapped to the included models. You can edit or delete these model definitions and you can create new ones.

Model definitions are created and edited on the Model Definition Editor page of the Visual Model Editors dialog box (Figure 87-1).

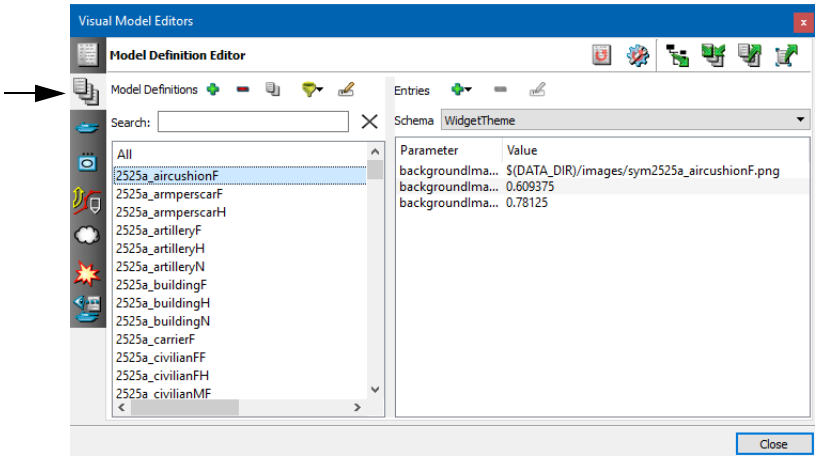


Figure 87-1. Model Definition Editor page

The model definitions provided with VR-Forces are stored in *./appData/definitions/ModelDefinitions*. When you open the Visual Model Editors dialog box, it reads the files in this directory in alphabetical order. If files contain duplicate entries, the entry in the last file loaded takes precedence. You can use command line options to change the location of the model definitions that get loaded at startup. For details, please see "Importing Model Definitions from the Command Line," on page 87-12.



- Any model definition that uses the Entity, Tree, or SpeedTree schema is automatically added to the Props Palette.
- The model definition for an entity includes the 3D model used to represent it. You can change the model in the Visual Model Editors dialog box. However, you can also quickly and easily change the 3D model for an entity in the Simulation Object Editor. For details, please see "Editing a Simulation Object's Visual Model," on page 68-13.
- Changes to element definitions do not take effect until you restart VR-Forces or reload the visual models. For information about reloading visual models, please see "Reloading Visual Models in the VR-Forces GUI," on page 68-25.
- You can also add model definitions in the Simulation Object Editor. However, you can only set the filename attribute.

87.2. Work Flow Issues for Model Definitions

A default VR-Forces installation has multiple model definition files. When you start VR-Forces they all get loaded. When you open the Visual Model Editors dialog box, all of the model definitions are listed. However, the editor does not keep track of which model definitions come from which model definition file. When you create or edit model definitions, VR-Forces does not automatically save them. Therefore, you are responsible for saving new and edited model definitions to the correct location.

The default model definition files are:

- *./appData/definitions/ModelDefinitions/AggregateLevelModel Definitions.ommx*. Model definitions specific to aggregate-level scenarios.
- *./appData/definitions/ModelDefinitions/baseModel Definitions.ommx*. Model definitions used by both entity-level scenarios and aggregate-level scenarios.
- *./appData/definitions/ModelDefinitions/coreModel Definitions.ommx*. Model definitions for objects that do not have *.entity* files.
- *./appData/definitions/ModelDefinitions/EntityLevelModel Definitions.ommx*. Model definitions for simulation objects in entity-level scenarios.

Before you add, delete, or edit a model definition, consider which of the following work flows best meets your need:

- If you want to create new model definitions, the preferred work flow is to export them to a new file. This is the easiest way to avoid overwriting other model definition files and makes them easily portable to newer releases of VR-Forces.

  If you want to add the new model definitions to an existing model definition file:

  a. Replace the model definitions in the editor by using the Load and Replace option to load the model definition file you want to edit. (For details, please see "Importing Model Definitions," on page 87-12.)

  b. Add the new model definitions.

  c. Export all changes back to the original model definition file.

- If you have edited existing model definitions, you can export them to their original file or to a new file:
  - If you export to a new file, make sure that the file name is alphabetically later in alphabetical order than the original model definition file. This ensures that when the new model definition is loaded it will overwrite the original one.
  - If you want to export to the original model definition file:

    a. Replace the model definitions in the editor by using the Load and Replace option to load the model definition file you want to edit. (For details, please see "Importing Model Definitions," on page 87-12.)

    b. Make your changes.

    c. Export all changes back to the original model definition file.

- If you want to delete a model definition, you must delete it from the source model definition file. Follow the work flow for editing an original model definition file.

For more information about how to export model definitions, please see "Exporting Model Definitions," on page 87-13.

! Do not use the Export All option if you are editing the combined model definition files that get loaded by default unless you are very certain about how you want to use the resultant file. If you export to a file in the *./appData/definitions/ModelDefinitions* directory, it will conflict with the other files in that directory.

87.3. Creating a Model Definition

> **!** VR-Forces does not save changes to model definitions automatically. If you want to keep changes, you must export them to a file.

To create a model definition:

1. Decide if you want to export the new model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the new model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 87-8.)
6. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens (Figure 87-2).

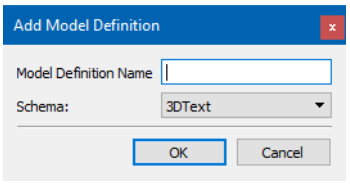


Figure 87-2. Add Model Definition dialog box

7. Type a name for the new model definition.

> **!** Model definitions for DI-Guy characters must end with the letters DIG. Model definitions for SpeedTrees must end in ST.

8. In the Schema list, select a schema for the model. (This list matches the list of schemas on the Schema Editor page.)
9. Add parameters to the model, as described in "Adding Parameters to a Model Definition," on page 87-8.
10. Export the model definition. (For details, please see "Exporting Model Definitions," on page 87-13.)

87.4. Deleting a Model Definition

If you want to delete a model definition, you have to delete it from its original model definition file. Since the Visual Model Editors dialog box loads multiple model definition files when it starts up, you must be sure that you only load the model definition file that you want to edit.



If a model definition is being used, you cannot delete it.

To delete a model definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Click the Import and Replace button (). The Import Model Definitions dialog box opens.
4. Select the model definition file you want to import.
5. Click Open.
6. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "Filtering the List of Model Definitions," on page 87-8.)
7. Select the model that you want to delete.
8. Click the Delete button () next to the Model Definitions label. You are prompted to confirm the deletion.
9. Click Yes. The model definition is removed from the list.
10. Click the Export All button (). The Export Model Definitions dialog box opens.
11. Select the model definition file to save to.
12. Click Save.

87.5. Copying a Model Definition

If you want a new model that is a variant of an existing one, it can be easier to copy the existing model and edit it, than to create a new model and add all of the parameters.

! VR-Forces does not save changes to model definitions automatically. If you want to keep changes, you must export them to a file.

To copy a model definition:

1. Decide if you want to export the new model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the new model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "Filtering the List of Model Definitions," on page 87-8.)
6. Select the model that you want to copy.
7. Click the Duplicate button ( ). The Duplicate Model Definition dialog box opens (Figure 87-3).

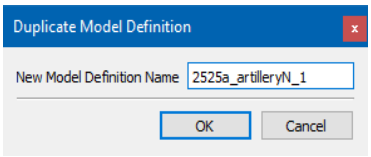


Figure 87-3. Copy model definition

8. Type a name for the copied version of the model.
9. Click OK.
10. The new model definition is added to the list.
11. Edit the parameters as described in "Editing a Model Definition," on page 87-8.
12. Export the model definition. (For details, please see "Exporting Model Definitions," on page 87-13.)

87.6. Filtering the List of Model Definitions

By default, the Model Definition Editor page lists all models that VR-Forces knows about. You can filter the list by schema to more easily work with a specific type of model.



You can also filter the list by typing a text string in the Search box. All model definitions that contain the text string are listed.

➤ To filter the list of model definitions, click the Filter button (▾) and select a schema type from the list.

87.7. Editing a Model Definition

You can add parameters to a model definition, you can delete parameters, and you can change their values. You can only add parameters that are appropriate to the schema type for a model.

VR-Forces does not save changes to model definitions automatically. If you want to keep changes, you must export them to a file.

87.7.1. Adding Parameters to a Model Definition

When you create a new model definition, you must specify the parameters required by its schema. You can only add parameters that are permitted by the schema.

To add a parameter to a model:

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "Filtering the List of Model Definitions," on page 87-8.)
6. Select the model for which you want to add a parameter.
7. Click the Add Parameter button (▾). A list of parameters is displayed (Figure 87-4).

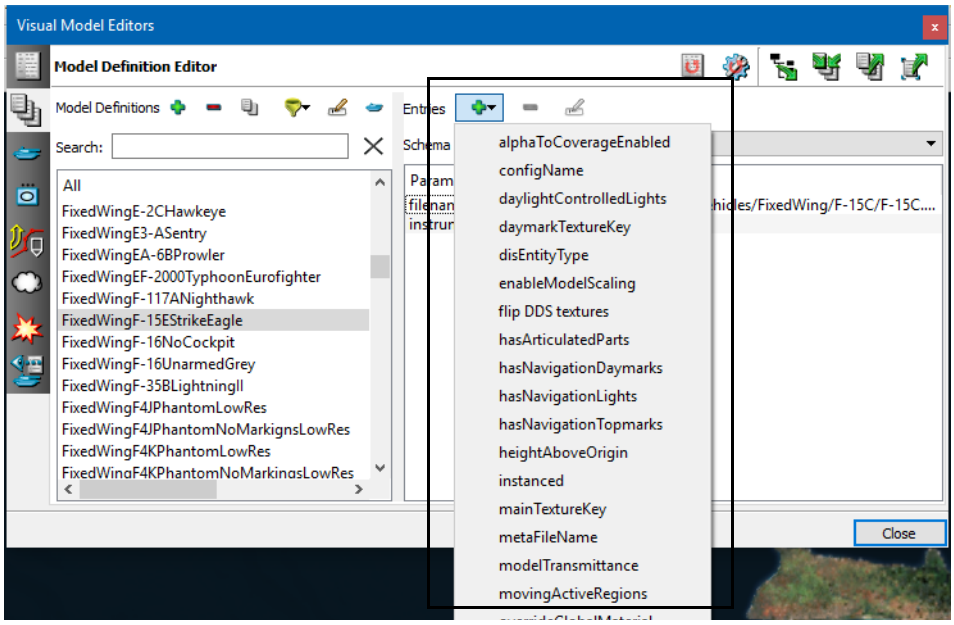


Figure 87-4. Add parameter list

8. Select the parameter you want to add. It is added to the parameter list.
9. Click the value for the parameter. The field becomes editable (Figure 87-5).

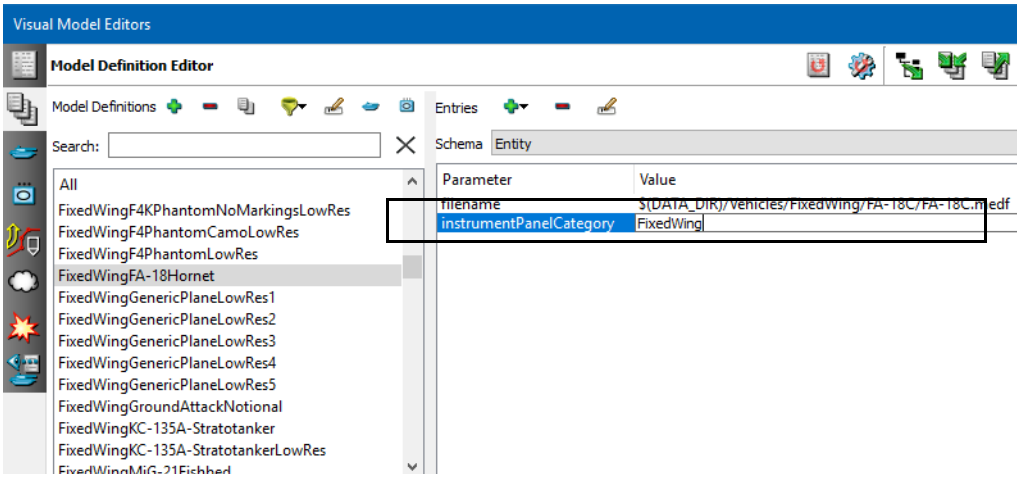


Figure 87-5. Editable parameter

10. Type or select a value for the parameter.
11. Click away from the field.
12. When you have finished editing the model definition, export it. (For details, please see "Exporting Model Definitions," on page 87-13.)

87.7.2. Editing a Model Definition Parameter

To edit a parameter:

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "Filtering the List of Model Definitions," on page 87-8.)
6. Select the model for which you want to edit a parameter.
7. Click the value for the parameter you want to edit. The field becomes editable (Figure 87-5).
8. Change the value for the parameter.
9. Click away from the field.
10. When you have finished editing the model definition, export it. (For details, please see "Exporting Model Definitions," on page 87-13.)

87.7.3. Deleting Parameters from a Model Definition

If you want to delete parameters from a model definition, you must export the changes back to the original file.

! If you delete a parameter from a model definition, the definition may become invalid. Invalid model definitions are shown in red.

To delete a parameter from a model:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Use the Load and Replace procedure to load the model definition file that has the model definition you want to edit.
4. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "Filtering the List of Model Definitions," on page 87-8.)
5. Select the model from which you want to delete a parameter.

6. Select the parameter that you want to delete.
7. Click the Delete button (➖) next to the Entries label. The parameter is removed.
8. Export the model definition to the original model definition file using Export All. (For details, please see "Exporting Model Definitions," on page 87-13.)

87.8. Jumping to an Element Definition

You can jump directly from a model definition to an associated element definition. Since some model definitions are used for multiple element definitions, the Visual Model Editor jumps to the first element definition that it finds that uses the model definition. This is primarily an issue for notional models.

To jump from a model definition to an element definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "Filtering the List of Model Definitions," on page 87-8.)
4. Select a model definition.
5. Click the Element Definition button (↩) on the Model Definitions button bar or right-click and select Element Definition. The Visual Model Editors dialog box switches to the selected element definition on the appropriate page.

87.9. Saving Model Definitions

If you add, edit, or delete model definitions, you must explicitly export the new or updated definitions to a file. For details, please see "Exporting Model Definitions," on page 87-13.

87.10. Importing Model Definitions

You can import model definitions and replace the loaded model definitions. You can also import model definitions and merge them with the loaded model definitions. For details about how merging works, please see "Managing VR-Forces Settings," on page 3-13.

To import model definitions and replace the existing definitions:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Click the Import and Replace button (). The Import Model Definitions dialog box opens.
4. Select the model definition file you want to import.
5. Click Open.

To merge model definitions:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Click the Import and Merge button (). The Merge Model Definitions dialog box opens.
4. Select the model definition file you want to merge.
5. Click Open.

87.10.1. Importing Model Definitions from the Command Line

You can merge in model definitions with the `--mergeModelDefs {`*`directory | filename`*`}` command-line option. If you specify a directory, VR-Forces merges all model definition files in the directory. If you specify a file, it just merges that file. You can have multiple instances of this option in a command.

You can specify that VR-Forces not load the model definitions in the *./appData/definitions* directory. To do this, start VR-Forces with the `--noAppDataModelDefs` command-line option.

87.11. Exporting Model Definitions

You can export all model definitions and you can export individual model definitions. This allows you to create your own library of custom model definitions that you can easily migrate to newer releases of VR-Forces or copy to other VR-Forces installs.

! The list of model definitions in a default VR-Forces installation is created by merging multiple model definition files. If you export all model definitions, they do not get exported back to their original files. They get exported to the file you select.

To export all model definitions:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Click the Export All button (). The Export Model Definitions dialog box opens.
4. Select the model definition file to save to or type a new file name.
5. Click Save.

87.11.1. Exporting Selected Model Definitions

To export selected model definitions:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 87-8.)
4. Select the model definitions you want to save.
5. Click the Export Selected button (). The Export Model Definitions dialog box opens.
6. Select a file or type a name for the model definition file.
7. Click Save.



88. Element Definitions

In Section XIV, "Configuring Visual Models in the Visual Models Editor" we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit element definitions.

Creating and Editing Element Definitions 88-2
How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions are Organized 88-3
Leaf File Descriptions 88-4
Work Flow Issues for Creating and Editing Element Definitions 88-5
Creating an Element Definition 88-7
Editing an Element Definition 88-8
Adding a Visual Definition 88-8
Editing a Visual Definition 88-9
Adding an Attribute to a Visual Definition 88-12
Deleting an Attribute from a Visual Definition 88-13
Jumping to a Model Definition 88-14
Deleting an Element Definition 88-14
Saving Element Definitions 88-15
Importing Element Definitions 88-15
Importing a Hierarchy File 88-16
Importing and Merging Element Definitions 88-16
Importing Element Definitions from the Command Line 88-17
Exporting Element Definitions 88-17
Exporting Interaction and Smoke Element Definitions to .elem Files .. 88-18
Exporting Entity, Unit, Spot Report, or Tactical Graphics Element Definitions 88-19

88.1. Creating and Editing Element Definitions

Element definitions define how VR-Forces renders scene elements (simulation objects, tactical graphics, interactions, cultural features, spot reports, and so on).

It is strongly recommended that you use the Simulation Object Editor to edit visual models, which includes the element definition.

Element definitions are organized in a tree-style hierarchy. At any level of the hierarchy an element definition inherits the visual definitions of its parents. This makes it possible to assign a visual definition to many elements at one time. You can change the inherited parameter values for a specific element at any level of the hierarchy. However, in most cases, you cannot delete an inherited visual definition.

You can create and edit element definitions. The procedures are the same for each element type except for the save procedure. Element definitions are not saved automatically. You must save them explicitly.



If you edit an element definition in the Simulation Object Editor and save the change, you can update an open scenario to use the new element definition. For details, please see "Reloading Visual Models in the VR-Forces GUI," on page 68-25.

88.2. How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions are Organized

Entity, unit, spot report, and tactical graphics element definitions are stored in two types of files, a hierarchy file (*.hier*) and one or more leaf definition files (*.leaf*). The hierarchy file contains the hierarchy of categories for the element definitions, such as Entity → Surface → Boat and the attribute inheritance information associated with each level of the hierarchy. The leaf files contain the definitions for each element at the leaf level[1].

Other types of element definitions do not save separate hierarchy and leaf files. They are saved to a file with the extension *.elem*.

When you add or edit entity, unit, spot report, and tactical graphics element definitions, you must save the changes to the hierarchy and leaf files separately. If you have only made leaf level changes, you do not have to save the hierarchy file.

You can create multiple hierarchy and leaf level files to meet your needs. However you can only load one hierarchy file at a time. When VR-Forces starts up, it loads the files in the various element definition directories. If it finds more than one hierarchy file for an element type, it only loads the first one (in alphabetical order). If it finds multiple leaf files, it loads them in alphabetical order. If there are any duplicate entries, the last file loaded takes precedence.

1. Technically, the leaf level of a tree structure is a node with no children. For element definitions, the leaf level consists of specific definitions. A category that has no members would be part of the hierarchy, not the leaf file.

88.2.1. Leaf File Descriptions

The element definitions for entity, unit, and tactical graphics are organized into multiple leaf files. This organization is to accommodate the different models in the AggregateLevel and EntityLevel SMSs. The element definitions are organized as follows:

- *./appData/definitions/Aggregate/ElementDefinitions*. Element definitions for hierarchical units such as platoons and companies:
  - *EntityLevelAggregateElementDefinitions.leaf*. Units used in entity-level scenarios.
  - *coreAggregateElementDefinitions.leaf*. Element definitions for objects that do not have *.entity* files.
  - *AggregateLevelAggregateElementDefinitions.leaf*. Units used in aggregate-level scenarios.
  - *baseAggregateElementDefinitions.leaf*. Unit elements used in both entity-level and aggregate level scenarios.
- *./appData/definitions/Entity/ElementDefinitions*. Element definitions for individual simulation objects such as humans, ships, and cars:
  - *EntityLevelEntityElementDefinitions.leaf*. Element definitions used in entity-level scenarios.
  - *coreEntityElementDefinitions.leaf*. Element definitions for objects that do not have *.entity* files.
  - *baseEntityElementDefinitions.leaf*. Element definitions used in both entity-level and aggregate level scenarios.
- *./appData/definitions/TacticalGraphics/ElementDefinitions*. Element definitions for tactical graphics:
  - *EntityLevelTacticalGraphicsElementDefinitions.leaf*. Tactical graphics used in entity-level scenarios.
  - *coreTacticalGraphicsElementDefinitions.leaf*. Element definitions for tactical graphics that do not have *.entity* files.
  - *AggregateLevelTacticalGraphicsElementDefinitions.leaf*. Tactical graphics used in aggregate-level scenarios.
  - *baseTacticalGraphicsElementDefinitions.leaf*. Tactical graphics used in both entity-level and aggregate level scenarios.
- *./appData/definitions/SpotReports/ElementDefinitions*. Element definitions for spot reports:
  - *EntityLevelSpotReportsElementDefinitions.leaf*. Tactical graphics used in entity-level scenarios.
  - *coreSpotReportsElementDefinitions.leaf*. Element definitions for tactical graphics that do not have *.entity* files.
  - *baseSpotReportsElementDefinitions.leaf*. Tactical graphics used in both entity-level and aggregate level scenarios.

88.3. Work Flow Issues for Creating and Editing Element Definitions

A default VR-Forces installation has multiple element definition files for entities, units, spot report, and tactical graphics. When you start VR-Forces they all get loaded. When you open the Visual Model Editors dialog box, all of the element definitions are listed. However, the editor does not keep track of which element definitions come from which element definition file.



Interaction and smoke element definitions do not have hierarchy and leaf files. They work as follows:

- They are loaded from files that have the extension *.elem*.
- You can export specific interaction element definitions to individual files. If you have more than one *.elem* file in a directory, all of them get loaded.
- You cannot export specific smoke element definitions to individual files. When you export them, all definitions get exported to the selected file.

Therefore, except for the need to explicitly save all changes, the considerations in this section do not apply to these types of element definitions.

When you create or edit element definitions, VR-Forces does not automatically save them. Therefore, you are responsible for saving new and edited element definitions to the correct location. This section describes the issues to consider when you work with entity, unit, spot report, and tactical graphics element definitions to get the results you want and avoid problems.

- If you have created new element definitions, the preferred work flow is to export them to a new leaf file. This is the easiest way to avoid overwriting other element definition files and makes them easily portable to newer releases of VR-Forces.

  If you want to add the new element definitions to an existing element definitions file:

  a. Load the hierarchy file. This removes all element definitions.

  b. Load the element definition leaf file you want to edit.

  c. Add the new element definitions.

  d. Export all changes back to the original element definition leaf file.

- If you have edited existing element definitions, you can export them to their original leaf file or to a new leaf file:
  - If you export to a new file, make sure that the file name is alphabetically later in alphabetical order than the original element definition file. This ensures that when the new element definition is loaded it will overwrite the original one.
  - If you want to export to the original element definition file, you must plan ahead:
    a. Load the hierarchy file. This removes all element definitions.
    b. Load the element definition leaf file you want to edit.
    c. Make your changes.
    d. Export all changes back to the original element definition file.

For more information about exporting element definitions, please see "Exporting Element Definitions," on page 88-17.

! Do not use the Export All Leaves option if you are editing the combined element definition files that get loaded by default unless you are very certain about how you want to use the resultant file. If you export to an existing element definitions file, it will conflict with the other files in that directory.

88.4. Creating an Element Definition

You can create new element definitions. Entity, unit and tactical graphics element definitions get exported to *.leaf* files. Smoke, spot report, and interaction element definitions get exported to *.elem* files.

VR-Forces does not save changes to element definitions automatically. If you want to keep changes, you must export them to a file.

To create an element definition:

1. If you are adding an entity, unit, spot report, or tactical graphic element definition, decide if you want to export the new element definition to its own leaf file or to an existing file. (For details about the element definition work flow, please see "Work Flow Issues for Creating and Editing Element Definitions," on page 88-5.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the page for the type of element definition that you want to create.
4. If you want to export the new element definition to an existing leaf file, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "Importing Element Definitions," on page 88-15.)
5. Expand the element definition hierarchy to the level at which you want to add an element definition and select the parent element (Figure 88-1).

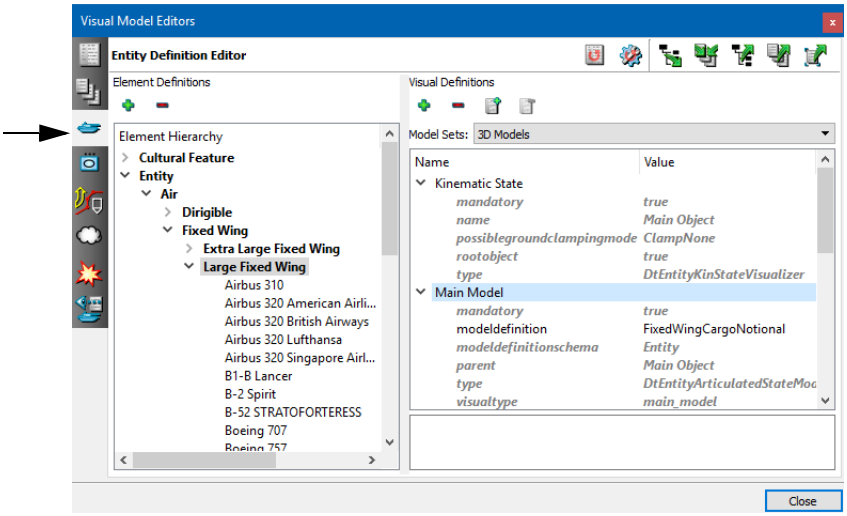


Figure 88-1. Entity Definition Editor page

6. Click the Add button (+). The Create Element Definition dialog box opens.

7. Type a name for the element definition. (You cannot edit the name after you create it, so try to avoid typing errors.)
8. Click OK. The new element is added to the list.
9. Optionally, edit the visual definitions assigned to the element. For details, please see "Editing a Visual Definition," on page 88-9.
10. Optionally, add a visual definition to the element. For details, please see "Adding a Visual Definition," on page 88-8.
11. Export the element definition. If you made any changes to an entity, unit, or tactical graphics hierarchy, you must export the hierarchy also. (For details, please see "Exporting Element Definitions," on page 88-17.)

88.5. Editing an Element Definition

An element definition consists of a list of visual definitions for that element. You can add visual definitions and you can edit their parameters. In most cases, you cannot delete an inherited visual definition.

VR-Forces does not save changes to element definitions automatically. If you want to keep changes, you must export them to a file.

88.5.1. Adding a Visual Definition

To add a visual definition to an element definition:

1. If you are editing an entity, unit, spot report, or tactical graphic element definition, decide if you want to export the edited element definition to its own leaf file or to an existing file. (For details about the element definition work flow, please see "Work Flow Issues for Creating and Editing Element Definitions," on page 88-5.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. If you want to export the edited element definition to an existing leaf file, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "Importing Element Definitions," on page 88-15.)
4. Select the page for the type of element definition that you want to edit (Figure 88-1).
5. Select the element definition that you want to edit.
6. Optionally, on the Model Sets list, select the model set you want to use.
7. Click the Add button () next to the Visual Definitions label. The Create Visualizer Definition dialog box displays a list of available visual definitions.

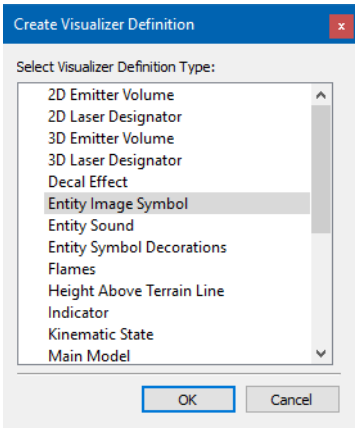


Figure 88-2. Create Visualizer Definition dialog box

8. Select the visual definition that you want to add.
9. Click OK. The visual definition is added to the element.
10. Optionally, edit the parameters for the visual definition. For details, please see "Editing a Visual Definition," on page 88-9.
11. When you have finished editing the element definition, export it. (For details, please see "Exporting Element Definitions," on page 88-17.)

88.5.2. Editing a Visual Definition

Visual definitions and their attributes are displayed either as gray italicized text or as black text (Figure 88-3). Italicized text indicates that the visual definition or attribute is inherited from a parent element. Visual definitions and attributes that are not italicized are particular to that element. (These entries could be inherited by child elements.)

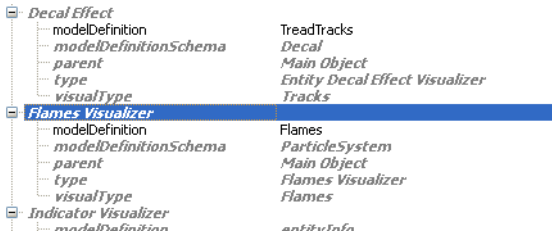


Figure 88-3. Visual definition entry

To edit a visual definition parameter:

1. If you are editing an entity, unit, spot report, or tactical graphic element definition, decide if you want to export the edited element definition to its own leaf file or to an existing file. (For details about the element definition work flow, please see "Work Flow Issues for Creating and Editing Element Definitions," on page 88-5.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the page for the type of element definition that you want to edit (Figure 88-1).
4. If you want to export the edited element definition to an existing leaf file, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "Importing Element Definitions," on page 88-15.)
5. Select the element definition that you want to edit.
6. In the Visual Definitions pane, click the value of the attribute that you want to edit. Depending on the attribute, a list of available values is displayed, a dialog box is displayed (Figure 88-4), or the field becomes editable.

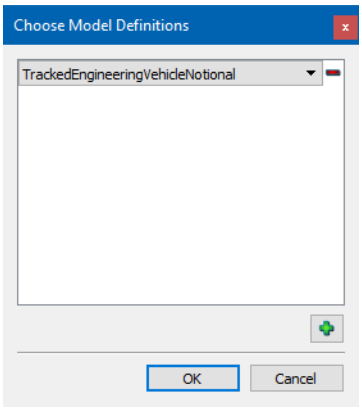


Figure 88-4. Choose Model Definitions dialog box

7. For a list, select the new value. For an editable field, type a new value. For a dialog box:
   a. If the dialog box window does not have a list, click the Add button (+). A list of values is added to the dialog box window.
   b. Select the value that you want.
   c. Click OK.
8. When you have finished editing the element definition, export it. (For details, please see "Exporting Element Definitions," on page 88-17.)

Mapping to Multiple Model Definitions or Schema

There may be cases where you want to map a visual definition to multiple model definitions or multiple schema. If you do this, VR-Forces randomly assigns the model definition or schema to instances of the object.

To map a visual definition attribute to multiple values:

1. If you are editing an entity, unit, spot report, or tactical graphic element definition, decide if you want to export the edited element definition to its own leaf file or to an existing file. (For details about the element definition work flow, please see "Work Flow Issues for Creating and Editing Element Definitions," on page 88-5.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the page for the type of element definition that you want to edit (Figure 88-1).
4. If you want to export the edited element definition to an existing leaf file, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "Importing Element Definitions," on page 88-15.)
5. Select the element definition that you want to edit.
6. In the Visual Definitions pane, click the value of the modeldefinition or modeldefinitionschema parameter that you want to edit. A dialog box opens. If the attribute already has values, they are listed in the dialog box window as a series of lists (Figure 88-5).

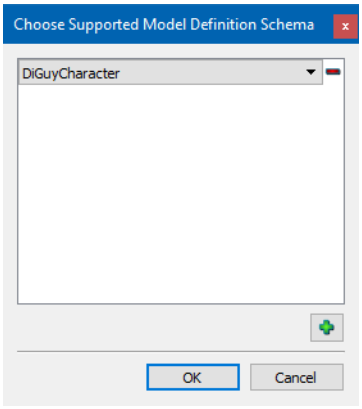


Figure 88-5. Parameter with multiple values

7. To add a value, click the Add button (+). A new list is added to the window.
8. Select a value from the new list.
9. To remove a value, click the Delete button (−) next to the list for that value.
10. Click OK. The visual definitions list does not show all of the values, but clicking it opens the dialog box again and shows the selected values.

11. When you have finished editing the element definition, export it. (For details, please see "Exporting Element Definitions," on page 88-17.)

88.5.3. Adding an Attribute to a Visual Definition

Each visual definition has a set of attributes. Some of them may be optional, so they are not specified as part of the factory definitions. You can add optional attributes to a visual definition.

! If you add an attribute to an element definition that is part of the element definition hierarchy, for example Entity → Land → Tracked, you must export all of the element definitions that inherit attributes from that definition. As a practical matter, this usually means exporting all of the definitions in that element definition file.

To add an attribute to a visual definition:

1. If you are editing an entity, unit, spot report, or tactical graphic element definition, decide if you want to export the edited element definition to its own leaf file or to an existing file. (For details about the element definition work flow, please see "Work Flow Issues for Creating and Editing Element Definitions," on page 88-5.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the page for the type of element definition that you want to edit (Figure 88-1).
4. If you want to export the edited element definition to an existing leaf file, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "Importing Element Definitions," on page 88-15.)
5. Select the visual definition that you want to edit.
6. Click the Add Attribute button (). The Add Visualizer Definition Attribute window opens. It lists the attributes that are available for the visual definition (Figure 88-6).

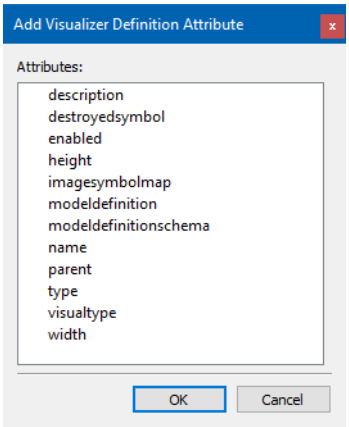


Figure 88-6. Add Visualizer Definition Attribute window

7. Select the attribute that you want to add.
8. Click OK. The attribute is added to the visual definition.
9. Specify a value for the attribute. This might require typing a value, selecting a value from a list, or selecting a value in another window.
10. When you have finished editing the element definition, export it. (For details, please see "Exporting Element Definitions," on page 88-17.)

88.5.4. Deleting an Attribute from a Visual Definition

You can delete individual attributes from a visual definition.

! The Delete button above the list of visual definitions deletes entire visual definitions. Do not click it thinking it will delete just the selected attribute.

To delete an attribute:

1. If you are editing an entity, unit, spot report, or tactical graphic element definition, decide if you want to export the edited element definition to its own leaf file or to an existing file. (For details about the element definition work flow, please see "Work Flow Issues for Creating and Editing Element Definitions," on page 88-5.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the page for the type of element definition that you want to edit (Figure 88-1).

4. If you want to export the edited element definition to an existing leaf file, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "Importing Element Definitions," on page 88-15.)
5. Select the visual definition that you want to edit.
6. Select the attribute that you want to delete.
7. Click the Delete Attribute button ( ).
8. When you have finished editing the element definition, export it. (For details, please see "Exporting Element Definitions," on page 88-17.)

88.6. Jumping to a Model Definition

You can jump from an element definition directly to any of the model definitions that it uses.

To jump from an element definition to an associated model definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to examine (Figure 88-1).
3. Right-click the element definition.
4. On the shortcut menu, select the model definition you want to jump to. The Visual Model Editors dialog box switches to the Model Definitions page and selects the chosen definition.

88.7. Deleting an Element Definition

VR-Forces does not save changes to element definitions automatically. If you want to keep changes, you must export them to a file.



You cannot delete an element definition if it is being used.

To delete an element definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to delete (Figure 88-1).

3. If you are deleting an entity, unit, spot report, or tactical graphic element definition, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "Importing Element Definitions," on page 88-15.)
4. Select the element definition that you want to delete.
5. Click the Delete button (−) next to the Entity Definitions label.
6. Confirm the deletion.
7. Export the element definition back to the original element definition file. (For details, please see "Exporting Element Definitions," on page 88-17.)

88.8. Saving Element Definitions

Changes to element definitions do not get saved automatically. You must explicitly export the changes to a file. For details, please see "Exporting Element Definitions," on page 88-17.

88.9. Importing Element Definitions

You can import element definitions and replace the loaded element definitions. You can import element definitions and merge them with the loaded element definitions.

If you want to import entity, unit, spot report, or tactical graphics element definitions, you can import hierarchy files, leaf files, or both. If you import a hierarchy file, it removes all leaf file entries and replaces the previously loaded hierarchy.

When you import a leaf file, it is merged according to the criteria described in "Managing VR-Forces Settings," on page 3-13.

You can also import or merge files in the legacy *.elem* format. If you import and replace entity, unit, or tactical graphics element definitions using an *.elem* file, the hierarchy and all element definitions are replaced with those in the elem file. If you merge an *.elem* file, the merge is additive only. Entities with names that already exist are skipped.



Interaction and smoke element definitions use the *.elem* format.

88.9.1. Importing a Hierarchy File

You can replace the current hierarchy with a different one by importing a hierarchy file or a file in the legacy *.elem* format.



When you import a hierarchy file, all leaf files are removed.

To import a hierarchy or element definition file and replace the existing definitions:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to import.
3. Click the Import and Replace button (). The Import Element Definitions dialog box opens.
4. Select the hierarchy file or element definition file (extension *.elem* or *.hier*) you want to import.
5. Click Open.

88.9.2. Importing and Merging Element Definitions

You can import and merge element definitions from leaf files or legacy element definition (*.elem*) files.



You cannot import a leaf file unless a hierarchy is already loaded.

To merge element definitions:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to import.
3. Click the Import and Merge button (). The Merge Element Definitions dialog box opens.
4. Select the element definition file (extension *.elem* or *.leaf*) you want to merge.
5. Click Open.

88.9.3. Importing Element Definitions from the Command Line

You can import and merge entity, unit, and tactical graphics element definitions when you start up VR-Forces using the following command-line options:

- `--entityDefsDir` *`directory`* merges in all *.hier* and *.leaf* files in the specified directory. The directory must be *./appData/definitions/Entity/ElementDefinitions*, *./appData/definitions/Aggregate/ElementDefinitions*, or *./appData/definitions/TacticalGraphics/ElementDefinitions* or be subordinate to one of them.
- `--mergeHierarchyEntityDef` *`filename`* imports the specified *.hier* file.
- `--mergeLeafEntityDef` *`filename`* merges in the specified *.leaf* file.
- `--mergeHierarchyTacticalGraphicsDef` *`filename`* merges in the specified tactical graphics *.hier* file.
- `--mergeLeafTacticalGraphicsDef` *`filename`* merges in the specified tactical graphics *.leaf* file.
- `--mergeHierarchyUnitDef` *`filename`* merges in the specified unit *.hier* file.
- `--mergeLeafUnitDef` *`filename`* merges in the specified unit *.leaf* file.

These commands merge element definitions into the default element definitions, which are usually those in *./appData/definitions*. You can also specify that the element definitions in *./appData/definitions* not be loaded. In that case, only those files specified on the command line would get loaded. If none are specified, no element definitions would be loaded.

To specify that the element definitions in *./appData/definitions* not be loaded, start VR-Forces with the `--noAppDataEntityDefs`, `--noAppDataUnitDefs`, or `--noAppDataTacticalGraphicsDefs` command-line options.

88.10. Exporting Element Definitions

Smoke and interaction element definitions get exported to a file with the extension *.elem*. Entity, unit, spot report, and tactical graphics element definitions, get saved to separate hierarchy (*.hier*) and leaf (*.leaf*) files.

You cannot save portions of the hierarchy. The entire hierarchy gets exported to file.

When the Visual Model Editors dialog box loads entity, unit, spot report, and tactical graphics element definitions it merges several files. When you export all element definitions, they do not get exported back to these files. They get exported to the file you specify. If you keep the new file in the same directory as the installed files, there might be conflicts between them when VR-Forces loads element definitions. The element definition files are sorted into alphabetical order and loaded in that order. In the case of conflicts, the last file loaded wins. Therefore:

- If you are only adding or editing a few element definitions, save them to one or more new leaf files rather than overwriting one of the default leaf files.
- When you export to new leaf files, consider carefully how you name them.

VR-Forces only ships one element definition file each for interactions (*InteractionsElementDefinitions.elem*) and tactical smoke (*SmokeElementDefinitions.elem*). However, you can save individual interactions, so it is possible to have more than one *.elem* file in the definitions directory for these element types. In that case, VR-Forces loads all of the *.elem* files it finds. It loads the files in alphabetical order. If there are duplicate element definitions, the last one loaded is used. Therefore if you customize an element definition and export it to a file and you want that definition to be used, give it a name that is later in the alphabet than *InteractionsElementDefinitions.elem* or *SmokeElementDefinitions.elem*.

88.10.1. Exporting Interaction and Smoke Element Definitions to *.elem* Files

When you export interaction element definitions or tactical smoke element definitions, they get exported to a file with the extension *.elem*. You can export all definitions to one file or you can export selected definitions to a file. If you have multiple element definition files in a directory, they get loaded in alphabetical order. If there are duplicate element definitions, the last one loaded is used.

To export all tactical smoke element definitions or interaction element definitions to a file:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the element definition type (Tactical Smoke Editor page or Interaction Definition Editor page).
3. Click the Export All button (). The Export Element Definitions dialog box opens.
4. Specify a file to export to.
5. Click Save.

To export selected interaction or tactical smoke element definitions to a file:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Interaction Definition Editor page.
3. Select the element definitions that you want to export.
4. Click the Export Selected button (). The Export Element Definitions dialog box opens.
5. Specify a file to export to.
6. Click Save.

88.10.2. Exporting Entity, Unit, Spot Report, or Tactical Graphics Element Definitions

If you want to export entity, unit, spot report, or tactical graphics element definitions, you must export the hierarchy and leaf information separately. If the hierarchy has not changed, you do not need to export it.

To export a hierarchy file:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to export.
3. Click the Export Hierarchy button (). The Export Element Definitions dialog box opens.
4. Select the hierarchy file you want to export to or type the name of a new file.
5. Click Save.

To export all definitions to a leaf file:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to export.
3. Click the Export All Leaves button (). The Export Element Definitions dialog box opens.
4. Select the leaf file you want to export to or type the name of a new file.
5. Click Save.

To export selected definitions to a leaf file:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to export.
3. Select the element definitions you want to export.
4. Click the Export Selected Leaves button (). The Export Element Definitions dialog box opens.
5. Select the leaf file you want to export to or type the name of a new file.
6. Click Save.



89. Mapping Models and Effects

This chapter explains how to map element definitions to DIS and HLA object types.

Introduction to Object Type Mapping 89-2
Work Flow Issues for Object Type Mappings 89-2
Adding an Object Type Mapping 89-4
Editing an Object Type Mapping 89-5
Filtering the Element Definition List 89-6
Jumping to an Element Definition 89-7
Deleting an Object Type Mapping 89-7
Saving Object Type Mappings 89-8
Importing Object Type Mappings 89-8
Exporting Object Type Mappings 89-8
Starting without Default Object Type Mappings 89-9

89.1. Introduction to Object Type Mapping

In order to visualize a scene element (entity, unit, detonation, and so on), VR-Forces must map its object type enumeration to an element definition. This chapter explains how to do that.



When you add simulation objects and other scene elements in the Simulation Object Editor, it creates model definitions and handles object type mapping. Although this chapter explains all of the details of the Simulation Object Type Mappings dialog box, using the Simulation Object Editor to add simulation objects is the preferred method.

You map object types to element definitions in the Simulation Object Type Mappings dialog box. It has pages for mapping:

- Cockpit displays.
- Detonation effects.
- Entities.
- Firing effects.
- Tactical graphics.
- Tactical smoke.
- Units.

The mapping process is the same for all object types. You can add new object type mappings, edit existing mappings, and delete mappings.



- You can map more than one element definition to an object type. If you do this, when entities of this type appear in a simulation, VR-Forces randomly assigns one of the element definitions to the entity.
- You can map an element definition to more than one object type.

89.2. Work Flow Issues for Object Type Mappings

A default VR-Forces installation has multiple object type mapping files. When you start VR-Forces they all get loaded. When you open the Simulation Object Type Mappings dialog box, all of the mappings are listed. However, the dialog box does not keep track of which mappings come from which mapping file. When you create or edit object type mappings, VR-Forces does not automatically save them. Therefore, you are responsible for saving new and edited mappings to the correct location.

The default object type mapping files are stored in *./appDate/definitions/<object_type>/TypeMap*, where <object_type> is Aggregate, Detonation, Entity, Fire, Smoke, or Tactical Graphics. Cockpit type mappings are in *./appData/settings/stealth/default_CockpitTypeMap.mhtx*. Detonation, fire, cockpits, and smoke have one default mappings file. The other object types each have four files, which contain the following types of mappings:

- Aggregate level mappings. Mappings for simulation objects in aggregate-level scenarios.
- Entity level mappings. Mappings for simulation objects in entity-level scenarios.
- Base mappings. Mappings for simulation objects used by both aggregate-level and entity-level scenarios.
- Core mappings. Mappings for simulation objects for which VR-Forces does not have *.entity* files.

Before you add, delete, or edit a simulation object type mapping, consider which of the following work flows best meets your need:

- If you have created new model mappings, the preferred work flow is to export them to a new file. This is the easiest way to avoid overwriting other mappings and makes them easily portable to newer releases of VR-Forces.

  If you want to add the new mappings to an existing mappings file:

  a. Use the Import and Replace option to load the mapping file you want to edit. (For details, please see "Importing Object Type Mappings," on page 89-8.)

  b. Add the new mappings.

  c. Export all changes back to the original mapping file.

- If you have edited existing mappings, you can export them to their original file or to a new file:
  - If you export to a new file, make sure that the file name is alphabetically later in alphabetical order than the original mapping file. This ensures that when the new mapping is loaded it will overwrite the original one.
  - If you want to export to the original mapping file:

    a. Use the Import and Replace option to load the mapping file you want to edit. (For details, please see "Importing Object Type Mappings," on page 89-8.)

    b. Make your changes.

    c. Export all changes back to the original mapping file.

- If you want to delete a mapping, you must delete it from the source mapping file. Follow the work flow for editing an original mapping file.

For more information about how to export mappings, please see "Exporting Object Type Mappings," on page 89-8.

89.3. Adding an Object Type Mapping

This procedure describes how to map an element definition to an object type enumeration. The element definition must already exist. The procedure is the same for any page on the Simulation Object Type Mappings dialog box.

To add an object type mapping:

1. Decide whether you want to save the new object type mapping to its own file or to an existing object type mapping file, as described in "Work Flow Issues for Object Type Mappings," on page 89-2.
2. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
3. Select the Entity Mapping Settings page (Figure 89-1). The page shows a list of object type enumerations in the Entity Type column. The entity definition mapped to the object type is listed in the Entity Definition column.

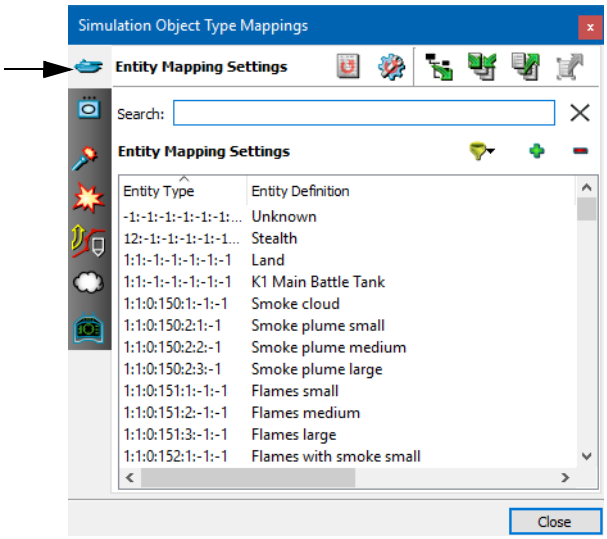


Figure 89-1. Entity Mapping Settings page

4. If you want to save the new object type mapping to an existing file, use the Import and Replace procedure to load the object type mapping file that you want to save to. (For details, please see "Importing Object Type Mappings," on page 89-8.)
5. Click the Add button (+). The Create New Entity Type Model Mapping dialog box opens (Figure 89-2). It lists a default object type enumeration.

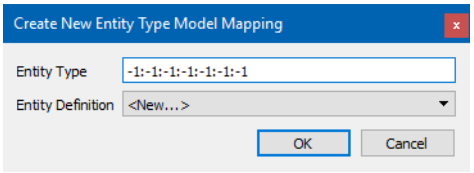


Figure 89-2. Create New Entity Type Model Mapping dialog box

6. In the Entity Type text box, type the enumeration you want to map to a model. (Separate each value with a colon(:).)
7. In the Entity Definition list, select the entity definition that you want to map to this object type.
8. Click OK.
9. The new model mapping is added to the list.
10. Export the new model mapping to a new or existing file. (For details, please see "Exporting Object Type Mappings," on page 89-8.)

89.4. Editing an Object Type Mapping

This procedure describes how to edit an object type mapping for a simulation object. The procedure is the same for any page on the Simulation Object Type Mappings dialog box.

To edit an object type mapping:

1. Decide whether you want to save the edited object type mapping to its own file or to an existing object type mapping file, as described in "Work Flow Issues for Object Type Mappings," on page 89-2
2. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
3. Select the Entity Mapping Settings page (Figure 89-1).
4. If you want to save the edited object type mapping to an existing file, use the Import and Replace procedure to load the object type mapping file that you want to save to. (For details, please see "Importing Object Type Mappings," on page 89-8.)
5. Optionally, filter the list by typing a text string in the Search box. All object types that include the text string in their name are listed.
6. Select the model mapping that you want to edit.
7. To change the enumeration associated with an element definition:
   a. In the Entity Type column, click the enumeration to make it editable.
   b. Change the values of the enumeration.
   c. Click someplace else in the window.

8. To change the element definition associated with an enumeration:
   a. Optionally, filter the list of available element definitions. (For details, please see "Filtering the List of Model Definitions," on page 87-8.)
   b. In the Entity Definition column, click the element definition to make it an editable list (Figure 89-3).
   c. Select the new element definition from the list.
   d. Click someplace else in the window.

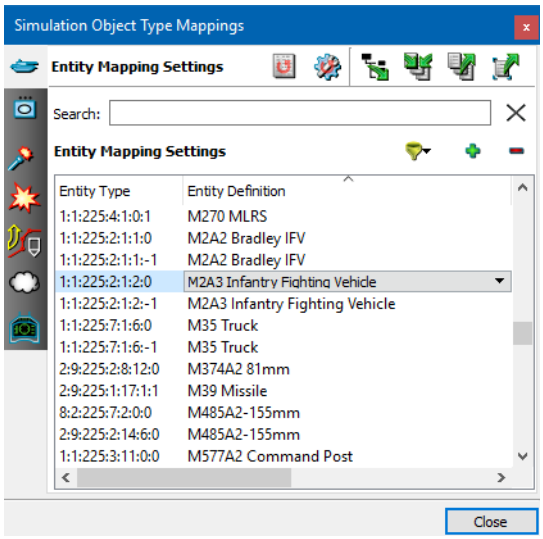


Figure 89-3. Editable entity definition

9. Export the model mapping to a new or existing file. (For details, please see "Exporting Object Type Mappings," on page 89-8.)

89.5. Filtering the Element Definition List

You can filter the list of element definitions that are available in the Entity Definition (Figure 89-3) list for mapping to entity types.



Filtering does not change the list of simulation objects in the Entity Mapping Settings window. It just changes the list of simulation objects in the Entity Definition list when you change a specific mapping. To change the list of objects in the window, type a text string in the Search box.

To filter the list of element definitions:

1. On any page of the Simulation Object Type Mappings dialog box, click the Filter button ( ). A menu is displayed. A check mark indicates the type of element definition that will be displayed when you make a element definition entry editable. (You may have to expand the hierarchy to see which elements are checked.)
2. Select the element definition type that you want to see in the list or select All to see all element definitions.

89.6. Jumping to an Element Definition

You can quickly jump from an object type mapping to the associated element definition in the Visual Model Editors dialog box.

To jump from an object type mapping to the associated element definition:

1. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the settings you want to examine.
3. Right click the entity type.
4. On the shortcut menu, select *object type* Definition. (For example, Entity Element Definition or Interaction Definition.) The Visual Model Editors dialog box opens to the selected definition.

89.7. Deleting an Object Type Mapping

The procedure for deleting an object type mapping is the same for any page on the Simulation Object Type Mappings dialog box.

To delete an object type mapping:

1. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 89-1).
3. Use the Import and Replace procedure to load the object type mapping file that you want to delete a mapping from. (For details, please see "Importing Object Type Mappings," on page 89-8.)
4. Optionally, filter the list by typing a text string in the Search box. All object types that include the text string in their name are listed.
5. Select the model mapping that you want to delete.
6. Click the Delete button ( ).
7. Export the model mapping to the file you imported. (For details, please see "Exporting Object Type Mappings," on page 89-8.)

89.8. Saving Object Type Mappings

New and changed simulation object type mappings do not get saved automatically. You must export them to the appropriate file. For details, please see "Exporting Object Type Mappings," on page 89-8.

89.9. Importing Object Type Mappings

You can import simulation object type mapping files. An imported file replaces or merges with the previously loaded mappings.

When you merge type mappings, mappings from the file being merged in overwrite matching mappings that have already been loaded. If you have multiple models mapped to an object enumeration, all of the mappings are replaced by the new mappings.

To import simulation object type mappings:

1. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the type of simulation object whose mappings you want to import.
3. To replace the current set of mappings, click the Import and Replace button (). The Import Mapping Settings dialog box opens.

   To merge mappings, click the Import and Merge button (). The Merge Mapping Settings dialog box opens.
4. Select the settings file you want to import.
5. Click Open.

89.10. Exporting Object Type Mappings

You must export all changes to simulation object type mappings. You can export all mappings to a file or you can select specific mappings and export them.

If you have not explicitly loaded and replaced the list of object type mappings, do not export them to an existing file. The existing file will be replaced either by all of the mappings loaded in the Simulation Object Type Mappings dialog box or by the selected object type mappings.

To export all simulation object type mappings to one file:

1. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the type of simulation object whose mappings you want to export.
3. Click the Export All button (). The Export Mapping Settings dialog box opens.
4. Specify the file to export to.
5. Click Save.

To export selected object type mappings to a file:

1. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the type of simulation object whose mappings you want to export.
3. Click the Export Selected button (). The Export Mapping Settings dialog box opens.
4. Specify the file to export to.
5. Click Save.

89.11. Starting without Default Object Type Mappings

When you start VR-Forces, you can specify that it not load the entity, unit, or tactical graphics type mappings in *./appData/definitions* or *./appData/settings/<application>*. To do this, start VR-Forces with the `--noAppDataEntityTypeMap`, `--noAppDataEntityTypeMap`, or `--noAppDataEntityTypeMap` command-line option.



90. Wakes, Cockpits, 2D Icons, and other Models

Section XIV, *Configuring Visual Models in the Visual Models Editor* provides a general introduction to model schemas, model definitions, and element definitions. This chapter describes how to configure several specific types of visual models.

Configuring 2D Icons 90-3
- Editing Font-Based 2D Icons 90-3
- Using Images for 2D Icons 90-4
- Adding Images for Simulation Object Icons 90-9

Adding New Cockpit Display Models 90-10
- Installing a Cockpit DLL 90-11
- Creating a Model Definition for a Cockpit 90-11

Adding Object Models as Props 90-13
- Adding a Category to the Props Palette 90-14

Configuring Light Points 90-14
- Configuring Light Point Type Models 90-16
- Configuring Light Point Group Models 90-16
- Adding a New Light Point Type 90-17

Configuring Wakes 90-18
- Configuring Tidal Stream Wakes 90-20

Adding Wind-based Controls to Models 90-21

Using Texture Atlases to Skin Models 90-22
- Configuring Texture Mapping in the Model Definition 90-22
- Metafiles for Texture Mapping 90-23

Adding Dynamically Changeable Markings to Models 90-24
- Adding @dis annotations to the Model 90-26
- The Metafile for Markings 90-27
- The Texture Mapping File 90-28

Flipping DDS Textures for a Model 90-29
Configuring SpeedTree Trees 90-30
Randomizing the Size of SpeedTrees 90-31
Displaying SpeedTrees in Planar Reflections 90-33
Best Practices for Creating Models 90-34
Disabling Indirect Rendering for an OpenFlight Model 90-35
Compressing Model Files 90-35

90.1. Configuring 2D Icons

The 2D icons in VR-Forces are configured in visual definitions. Font-based icons are configured in the Military Symbol Icon visual definition. Image-based icons use the Entity Image Symbol visual definition.



You can quickly change the 2D icon for a simulation object in the Simulation Object Editor. For details, please see "Editing a Simulation Object's Visual Model," on page 68-13.

90.1.1. Editing Font-Based 2D Icons

By default, 2D icons are configured in the Military Symbol Icon visual definition, which is part of a simulation object's entity definition. This visual definition uses font-based icons. You can change a simulation object's icon by changing the fillGlyph and outlineGlyph attributes.

To change a simulation object's icon:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 90-1).
3. Select the simulation object whose icon you want to edit.
4. In the Model Sets list, select 2D Icons.
5. To change the fill glyph:
   a. Under the Military Symbology Icon visual definition, select the fillGlyph attribute (Figure 90-1).

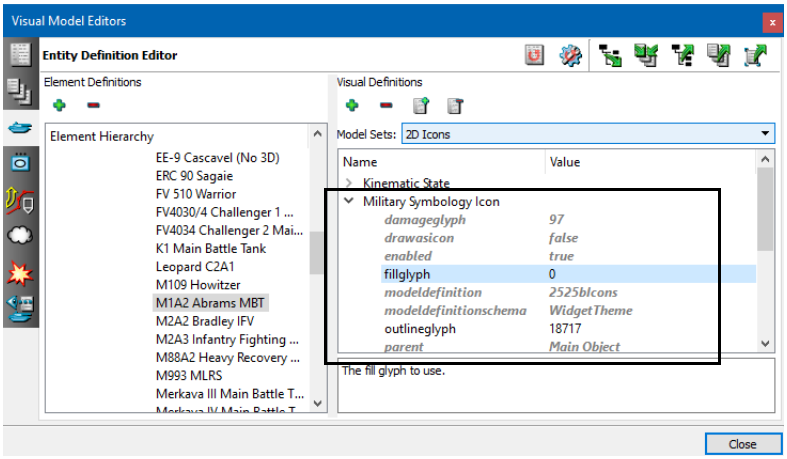


Figure 90-1. Military Symbol Icon visual definition

   b. Click the value in the Value column. The Choose Font Glyph dialog box opens (Figure 90-2).

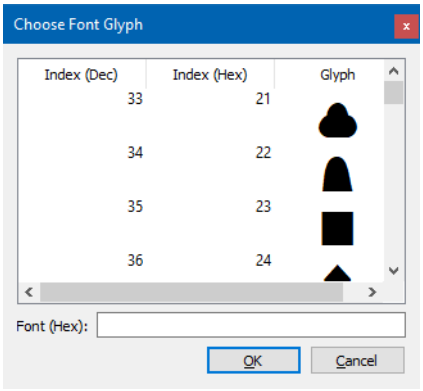


Figure 90-2. Choose Font Glyph dialog box

c. Select the glyph with the fill pattern you want to use. Click OK.

6. To change the outline glyph, follow the same procedure as changing the fill glyph.

90.1.2. Using Images for 2D Icons

By default, VR-Forces uses TrueType fonts for 2D icons. You can replace the font-based icons with images using the Entity Image Symbol visual definition.

If you configure images for icons, be sure to save the element definitions to a file other than the default settings. If you revert to factory defaults or otherwise switch back to font-based icons and you have not saved the image mappings, you will not be able to recover them without repeating the entire mapping process again.

To use images for entity icons:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 90-1). (If you want to change the icon for a unit, select the Unit Definition Editor page. To change the icon for a waypoint, select the Tactical Graphics page.)
3. In the Model Sets list, select 2D Icons.
4. In the Element Definitions hierarchy, select Entity. (If you are editing unit images, in the Element Definitions hierarchy select Unit. If you are editing tactical graphics images, select the specific tactical graphic or hierarchy level that you want to edit.)
5. In the Visual Definitions list, select Military Symbology Icon (Figure 90-3). (Some tactical graphics do not have this visual definition.)

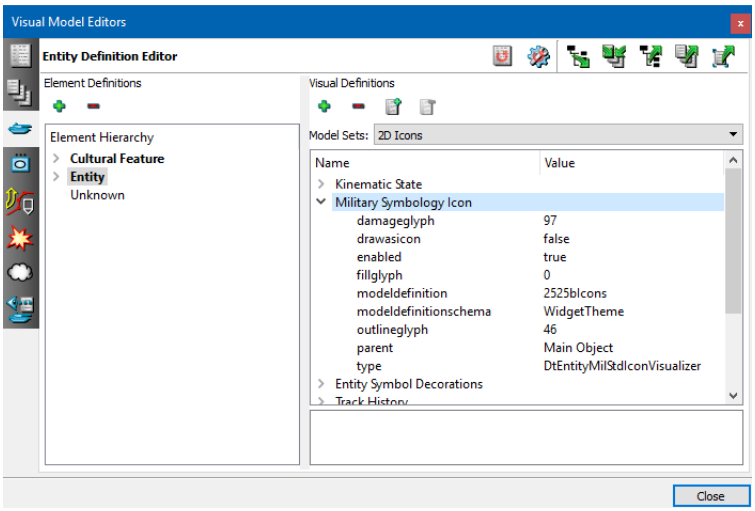


Figure 90-3. Entity element definition with Military Symbology Icon visual definition

6. Click the Delete button (−). The visual definition gets deleted.
7. Click the Add button (+). The Create Visualizer Definition window opens. It displays a list of visual definitions (Figure 90-4).

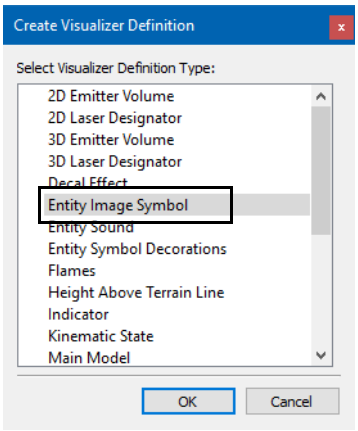


Figure 90-4. Create Visualizer Definition window

8. Select Entity Image Symbol. (If you are configuring images for units, select Aggregate Image Symbol; for tactical graphics, select Tactical Graphic Image Symbol.)
9. Click OK.
10. In the Visual Definitions list, select Entity Image Symbol (Figure 90-5).

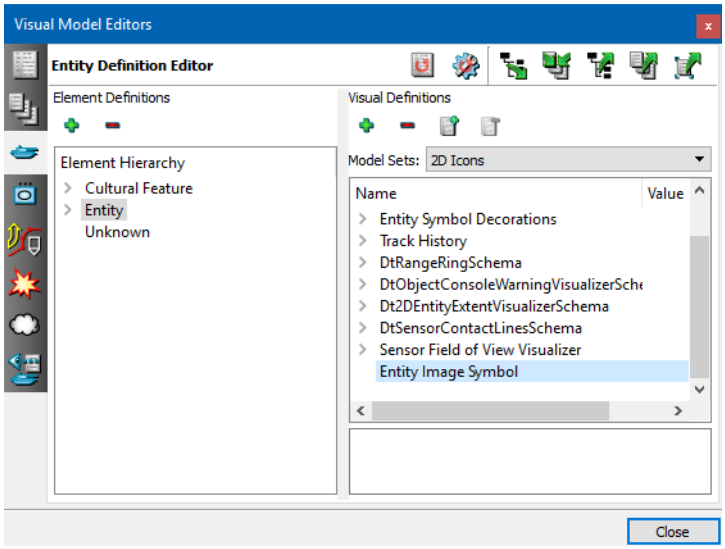


Figure 90-5. Entity element definition with Entity Image Symbol visual definition

11. Click the Add Attribute button ( ). The Add Visualizer Definition Attribute window is displayed (Figure 90-6).

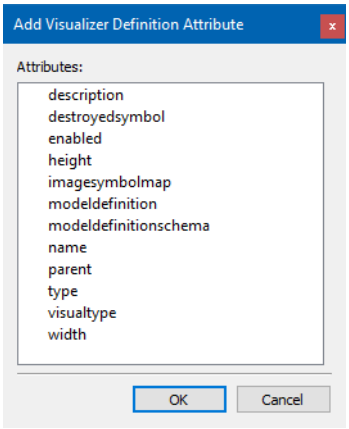


Figure 90-6. Add Visualizer Definition Attribute window

12. In the attribute list, select **parent**.
13. Click OK. The **parent** attribute is added to the visual definition.
14. Click the value field to make it editable.
15. Type Main Model and press **ENTER**.
16. Click the Add Attribute button.
17. In the attribute list, select **type**. The value is filled in automatically.
18. Click the Add Attribute button.

19. In the attribute list, select **modeldefinitionschema**. The Choose Supported Model Definition Schema dialog box opens (Figure 90-7).

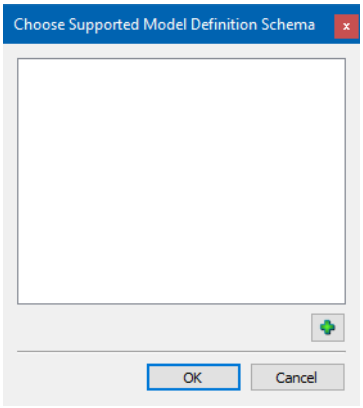


Figure 90-7. Choose Supported Model Definition Schema dialog box

20. Click the Add button. A list is added to the dialog box.
21. In the list, select **WidgetTheme** (Figure 90-8).

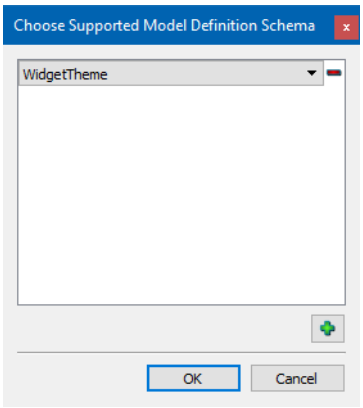


Figure 90-8. Completed Choose Supported Model Definition Schema dialog box

22. Click OK.
23. Click the Add Attribute button.
24. In the attribute list, select **imagesymbolmap**. The 2D Symbol Image Mapper dialog box opens (Figure 90-9).

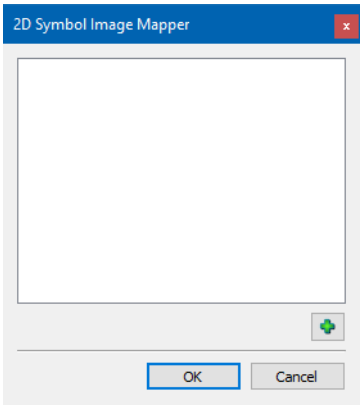


Figure 90-9. 2D Symbol Image Mapper dialog box

25. Click the Add button. A list is added to the dialog box. It lists the forces for which you can map images. and the images that you can apply to a particular force
26. Select a force and an image from the lists (Figure 90-10).

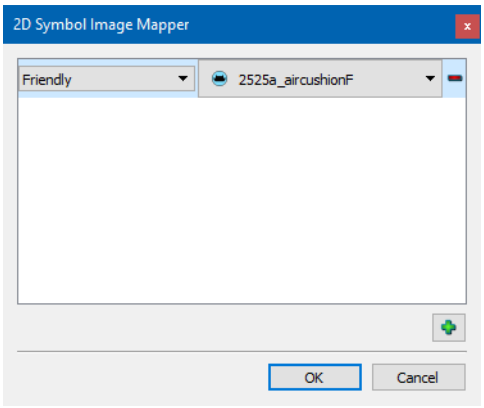


Figure 90-10. 2D Symbol Image Mapper dialog box with force chosen

27. Add an entry for each force that you want to represent with an image.
28. Click OK.
29. Click the Add Attribute button.
30. In the attribute list, select **destroyedsymbol**. The **destroyedsymbol** attribute is added to the visual definition.
31. In the **destroyedsymbol** list, select Destroyed. The completed visual definition should look similar to Figure 90-11.

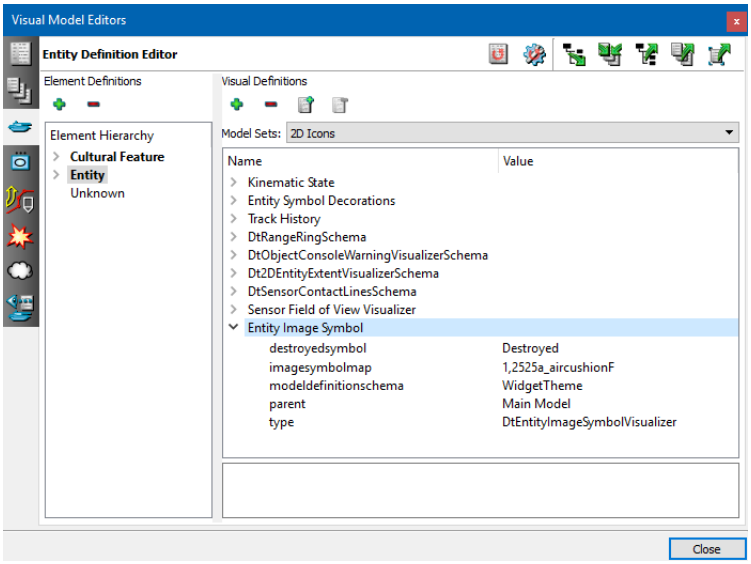


Figure 90-11. Entity Image Symbol visual definition

32. Export the updated setting.

The settings for the Entity Image Symbol visual definition are inherited by each entity in the hierarchy from the base Entity definition. Unless you configure the various entity categories, all entities use the images you configured for the Entity category. Therefore, at a minimum, change the **imagesymbolmap** values for each force at each generic object type, such as Fixed Wing and Life Form, to use the proper image for that object type. Optionally, you can specify an image for each entity definition at the leaf level.



When you edit the Entity Image Symbol visual definition attributes, be sure that you select the 2D Icons model set.

90.1.3. Adding Images for Simulation Object Icons

VR-Forces includes a representative set of MIL-STD 2525a images. They are in *./data/images*. If you want to add your own images, you can do so by adding a new model definition or changing the image mapping for an existing model definition. The new images are automatically added to the list of images for the **imagesymbolmap** in the Entity Image Symbol visual definition.

To add images for entities:

1. Copy the new images to *./data/images*.
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. In the Schema list, select WidgetTheme.

5. If you want to change the image used for an existing model definition:
   a. Select the model definition that you want to change, for example, 2525a_groundF.
   b. Change the file specified for the **backgroundImage** attribute.

   If you want to specify an image for a new model definition:

   a. Create a new model definition, as described in "Creating and Editing Model Definitions," on page 87-2. It is recommended that you copy an existing model definition for a 2525a symbol.
   b. Specify the image that you want to use for the new model definition.

90.2. Adding New Cockpit Display Models

VR-Forces supports cockpit displays created with GL Studio. A cockpit is implemented through a Reusable Software Object (RSO) DLL. A cockpit's values are changed through updaters. An updater is associated with a cockpit attribute through the cockpit model definition. Figure 90-12 shows the model definition for a cockpit.

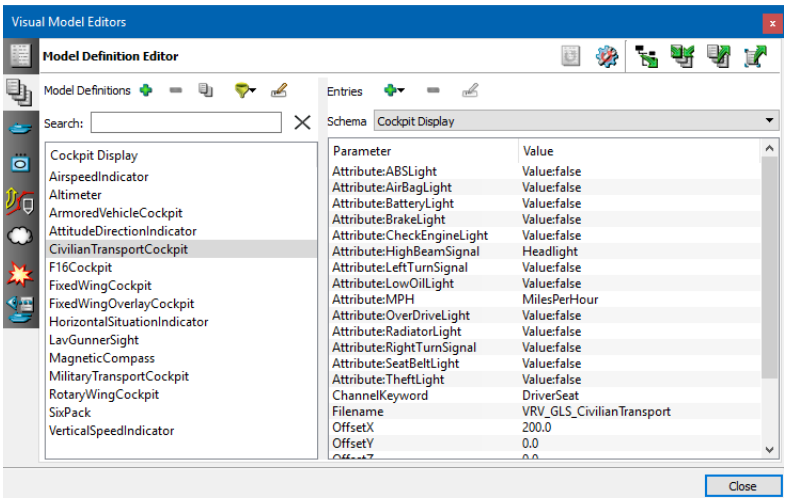


Figure 90-12. Cockpit model definition

Cockpit updaters are associated with an attribute name that is known to the RSO. The value you provide for a cockpit-specific attribute is the name of an updater for dynamically changing values or a constant value. Constant values are commonly used to configure a cockpit when it is loaded. VR-Forces provides a set of commonly used updaters. You can add additional updaters or override existing updaters through a plug-in.



The GL Studio license included with VR-Forces only covers the cockpits delivered with the product. If you add your own cockpits, you will need GL Studio run-time licenses.

The general procedure for adding a new cockpit model is:

1. Install the cockpit's DLL.
2. Create a model definition for the cockpit.
3. Map object types to the model definition, as described in "Adding an Object Type Mapping," on page 89-4.

90.2.1. Installing a Cockpit DLL

If you create your own cockpit, it is implemented as an RSO DLL that you must add to the VR-Forces file structure.

➤ To install a new cockpit, place the DLL for the cockpit in *./data/Huds* and create a model definition for it.

90.2.2. Creating a Model Definition for a Cockpit

Cockpit models are configured using schemas and model definitions, just like any other model in VR-Forces. This section provides additional details specific to cockpits.

To create a model definition for a cockpit:

1. Add a new model definition, as described in "Creating a Model Definition," on page 87-5. Be sure to select Cockpit Display as the schema. A new model definition is added to the display (Figure 90-13). The message window lists the parameters that you must add. These are the parameters defined by the schema.

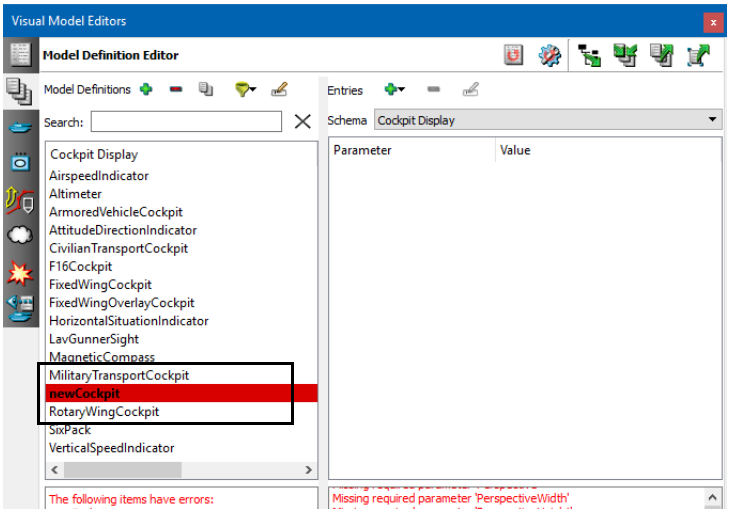


Figure 90-13. New cockpit model definition

2. Add the required parameters, as described in "Adding Parameters to a Model Definition," on page 87-8. Table 90-1 describes the required parameters.

Table 90-1: Required parameters for cockpit models

Parameter	Description
Filename	The name of the DLL without the file extension.
RSOName	The name of the class to be loaded from the DLL. Default: *filename*Class.
Perspective	2D or 3D. This parameter refers to the perspective of the cockpit display, not the terrain.
PerspectiveWidth PerspectiveHeight	Adjusts the orthogonal projection for a 2D cockpit. Adjusts the perspective projection for a true 3D cockpit.
ChannelKeyword	A keyword that tells a channel to display any object that has the keyword in its definition. Default: showCockpits. (This keyword means that any channel that has the showCockpits keyword will show cockpits.)
Offset-X Offset-Y Offset-Z	Moves the cockpit offset around the screen. The OffsetX value shifts the cockpit left or right. The OffsetY value shifts it up or down. OffsetZ shifts a 3D cockpit forward and back. It has no effect on 2D cockpits.

3. Add the parameters required by the cockpit RSO. To do this, add an Attribute parameter, as follows:
   a. Add an Attribute parameter, as described in "Adding Parameters to a Model Definition," on page 87-8.
   b. Double-click the parameter to make its name editable.
   c. Add a colon and the name of an RSO attribute that you need to map. For example, Attribute:Heading.
   d. In the Value column, click the value to make it editable. Type the name of an updater or specify a constant value in the format Value:*value*. You can specify an updater that you have added through a plug-in or one of the updaters provided with VR-Forces, as described in Table 90-2.
4. Map an object type to your definition, as described in "Adding an Object Type Mapping," on page 89-4.

Table 90-2: Updaters provided

Updater	Description
AOA	Angle of Attack. Returns the pitch of the entity clamped (5 to 30 degrees).
FeetAGL	Feet Above Ground Level. Returns the feet above the terrain (or buildings below).

Table 90-2: Updaters provided

Updater	Description
FeetMSL	Feet Above Mean Sea Level. Returns the altitude above the sea level.
FeetPerMinute	Feet Per Minute. Returns the climb rate.
KilometersPerHour	Speed of the entity in kilometers per hour.
MilesPerHour	Speed of the entity in miles per hour.
KnotsPerHour	Speed of the entity in nautical miles per hour.
Pitch	Returns the pitch of the entity clamped (-180 to 180 degrees).
Roll	Returns the roll of the entity clamped (-180 to 180 degrees).
Yaw	Returns the yaw of the entity clamped (0 to 360 degrees).
Headlight	Returns the headlight state of the entity as true or false.
Value	This updater sends any value to the cockpit once and then never again. It is used to configure initial setup and to turn display components on or off.

90.3. Adding Object Models as Props

Props are terrain objects that you can select independently of the terrain. "Extracting Props from a Terrain," on page 66-8 explains how to create them by extracting them from a terrain or a feature layer. You can also add props by creating model definitions for them. These props are added to the Props Palette and are then available to add to a terrain.

To add an object model as a prop:

1. Add a new model definition, as described in "Creating and Editing Model Definitions," on page 87-2. Use the Entity, Tree, or SpeedTree schema, as appropriate. When you name the model definition, if you preface it with one of the categories on the Props Palette, such as Buildings, it automatically is placed in that category on the palette. For example, BuildingsMyBuilding puts MyBuilding in the Buildings category.
2. Add the filename parameter.
3. Set the value for the filename to the location of your object model file. You can select any supported format. (Some formats, such as Collada (DAE) are not included in the file filter. To select such a format, choose All Files in the filters list.)

90.3.1. Adding a Category to the Props Palette

The list of categories on the Props Palette is controlled by the configuration file *./appData/settings/vrfGui/VRF3DTerrainPropCategories.xml*. To add a new category, edit the file. The format should be obvious from examining the existing XML elements.

If you edit *VRF3DTerrainPropCategories.xml*, be sure to update the <count> element to reflect additions or deletions.

90.4. Configuring Light Points

VR-Forces supports configuration of light points using light point types. A light point type is an optional configuration for a light point that allows consistent behavior across models and terrains.

Light point type settings are configured in *./appData/importConfig/Lights_DefaultValues_3_2.csv*. You can edit this file and add additional light point types. Table 90-3 describes the parameters specified in *Lights_DefaultValues_3_2.csv*.

Table 90-3: Light point type settings

Parameter	Descriptions
Code	Integer that specifies the light point type for this row.
Description	String that specifies a human readable description for the light point type. It is currently unused in VR-Forces.
Intensity	Floating point number that specifies the intensity of the light point. Usually (0-1.0) but does support values higher than 1.0.
Red	Floating point number that specifies the red color value for the light point. Usually (0-1.0), but supports values higher than 1.0.
Green	Floating point number that specifies the green color value for the light point. Usually (0-1.0), but supports values higher than 1.0.
Blue	Floating point number that specifies the blue color value for the light point. Usually (0-1.0), but supports values higher than 1.0.
Directionality	String that specifies the directionality of the light point. Valid strings are: ◆ `Omni-directional` - Light point has the same intensity when viewed from any direction. ◆ `Directional` - Light point has the regularly specified intensity when viewed from the within the lobe and the residual intensity when viewed outside the lobe. ◆ `Bidirectional` - Light point has the regularly specified intensity when viewed from the lobe or another lobe in the opposite direction and the residual intensity when viewed outside the two lobes.

Table 90-3: Light point type settings

Parameter	Descriptions
Horizontal Lobe	Floating point value in degrees for the angle that describes the horizontal lobe away from the light point's direction. Only used when Directionality is set to `Directional` or `Bidirectional`. (0.0 - 360.0)
Vertical Lobe	Floating point value in degrees for the angle that describes the vertical lobe away from the light point's direction. Only used when Directionality is set to `Directional` or `Bidirectional`. (0.0 - 360.0)
Residual Intensity	Floating point value that specifies the intensity of the light point when the light point is viewed from a direction that is outside the lobe. Only used when Directionality is set to `Directional` or `Bidirectional`. Usually (0.0 - 1.0) but does support values greater than 1.0.
Frequency	Floating point value that specifies the number of times per second that the light point should blink. 0.0 turns off blinking.
Duty Cycle	Floating point value that specifies the fraction of time that the light point should be on. (0.0 - 1.0)
Min Pixel Size	Positive floating point value that specifies the minimum number of pixels for the light point's diameter.
Max Pixel Size	Positive floating point value that specifies the maximum number of pixels for the light point's diameter.
Max Visible Distance	Positive floating point value that specifies the maximum distance in meters that the light point should be visible. This only has an effect if the light point is bright enough to be visible from this distance.
Radius	Positive floating point value that specifies the size of the light point in meters.
Daylight Controlled	Specifies whether or not the light point should be visible during the day. If true, then the light point turns off during the day.
Bulb Type	Specifies the type of bulb that the light point is using. It is currently only used by SensorFX and matches the entries in the *mm_lsources.dat* table. Please see SensorFX documentation for a description of how to configure light point bulb types in SensorFX.

90.4.1. Configuring Light Point Type Models

You can configure light point types for a particular light point in the OpenFlight model using Creator in a couple of different ways.

The simplest way is to use the light point palette manager in Creator to specify the Appearance name for a particular light point appearance index. The name can either be a CDB type name, which can be set using the **Edit → CDB Light Point Palette Names** menu or it can be an integer value that is one of the codes specified in *Lights_DefaultValues_3_2.csv*.

The CDB names are in *./appData/importConfig/Lights.xml*, which has a hierarchy defined using XML. For example the light point type with code 1 would have the CDB name of \Light\Platform based on the hierarchy in *Lights.xml*.

90.4.2. Configuring Light Point Group Models

You can group light points into light point groups, which let you control the color of a group of light points. These groups can group light points for things like runways, airports, parks, and cities.

Light point groups are automatically generated for the procedural airports and the CDB database airports using the runway and airport names.

A terrain that has an OpenFlight airport generated by the Creator Airport or Runway wizards will also automatically have light point groups if the airport name follows the format "Airport - `{name here}`" where `{name here}` specifies the airport name.

The default grouping name style for airports and runways is `{Airport Name}:{Runway Name}:{optional type}` where `{optional type}` is Approach or Runway.

Another way to set the light point group is to use comments on the light point group node. If the comment contains `@dis light_group {group name}` where `{group name}` is the name for this light point group, then all light points under that particular light point group node will be in that light point group.

You can set a light point group color using the VR-Forces view control messages, CIGI component control messages, or the VR-Forces developers toolkit.

90.4.3. Adding a New Light Point Type

You can add new light point types to the default configuration files. To add a new light point type, you add an entry to the end of *./appData/importConfig/Lights_DefaultValues_3_2.csv* and to *./appData/importConfig/Lights.xml*. You can add it anywhere in the *Lights.xml* hierarchy, but it is recommended that you add it to the end of the `Lights` element.

The following example shows how to add light point type 488 to the configuration.

1. Add the following to the bottom of *Light_DefaultValues_3_2.csv*:

```
488, Red Custom,0.6,1,0,0,Omnidirectional,57.2958,57.2958,0,0,0.5,1,20,
  100000,0.2
```

2. Change the bottom of the *Lights.xml* from:

```
  </Light>
</Lights>
```

   To:

```
  </Light>
  <Light type="Red_Custom" code="488">
    <Description>Red Custom Light Type</Description>
  </Light>
</Lights>
```

3. Create a model that uses the new custom light point type.

90.5. Configuring Wakes

When dynamic ocean is enabled, VR-Forces simulates wakes on surface entities. Wakes are configured in the Ship Wake visual definition of an entity's element definition. If you add your own model, you may want to change the default configuration values for its wake so that it more closely matches the dimensions and velocity of the object type being modeled.



You can also configure wakes in the Simulation Object Editor. For details, please see "Configuring Wakes," on page 69-20.

To configure a ship wake in the Visual Model Editors dialog box:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 90-14).

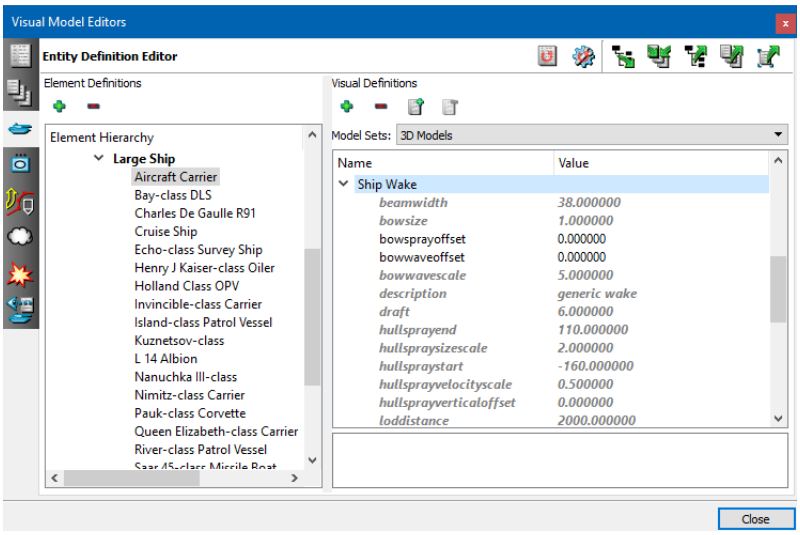


Figure 90-14. Ship Wake visual definition

3. In the Element Hierarchy, expand the Entity list to the Surface level and select the entity that you want to configure.
4. In the Visual Definitions list, select and expand the Ship Wake visual definition.

5. Edit the parameters. Table 90-4 describes each parameter.

Table 90-4: Ship Wake visual definition attributes

Parameter	Description
beamwidth	The width of the ship at its widest point.
bowsize	The size of the bow in world units. This affects the wavelength of the bow wave and the initial spread of spray particles at the bow. Use 0 for a pointy bow.
bowsprayoffset	An offset from the ship position from which to emit spray particles.
bowwaveoffset	An offset from the ship position, along the direction of travel, at which bow waves will originate.
bowwavescale	A scaling factor for the bow wave's amplitude at the bow of the ship.
draft	The depth of the hull underwater.
hullsprayend	The offset from the ship at which hull spray effects end.
hullspraysizescale	A scaling factor applied to the hull spray particles.
hullspraystart	The offset from the ship at which hull spray effects begin.
hullsprayvelocityscale	The initial velocity of hull spray particles as a percent of the ship velocity (0-1).
hullsprayverticaloffset	A vertical offset to the starting point of new hull spray particles.
loddistance	The distance from the camera beyond which wakes are not drawn. Cannot be greater than the value set by max-kelvin-wake-distance in the MAK Overrides section of *./data/Environment/Ocean/Triton.config*.
maxbowwave	The maximum amplitude of the bow wave, or -1.0 for unbounded.
numhullsprays	The number of spray particles emitted periodically along the hull of the ship.
parent	The visualizer's parent.
propwashenabled	Enable or disable propeller backwash effects.
propwashfadetime	Number of seconds during which the propeller backwash effects will dissipate.
propwashoffset	An offset from the ship position from which to generate propeller backwash effects.
shiplength	A value that, in combination with the entity's speed, is used to determine the length of the bow wake.
sprayenabled	If enabled, the wake emits spray particles.
spraysizescale	A scaling factor applied to the size of the bow spray particles and their random spread.

Table 90-4: Ship Wake visual definition attributes

Parameter	Description
sprayvelocityscale	A scaling factor for spray effects at the bow of the ship. This is applied to the initial velocity of the spray particles.
sternwaveenabled	Enable or disable the stern wake.
sternwaveoffset	An offset for the stern wakes.
type	The type of visualizer.
wakeoffset	An offset to the model's origin that all other offsets are relative to. Changing this offset lets you move all of the wake effects in tandem if the entity model changes.

90.5.1. Configuring Tidal Stream Wakes

Tidal stream wakes are supported as stand-alone models and are automatically attached to streamed buoys and beacons. Model definitions for wakes to be used with each of the supported buoy and beacon models are included (BuoysPillarWake, BeaconsLatticeWake, and so on). You may want to edit them. We recommend wake lengths of at least 20m for best results. (Some of the model definitions specify values less than that.)

By default, VR-Forces starts with a tidal stream current speed of 0. To see tidal stream wakes, change the speed on the Scene Settings dialog box, Environment Conditions Settings page. For more information, please see "Configuring Marine Conditions," on page 11-22.

90.6. Adding Wind-based Controls to Models

VR-Vantage can simulate the movement of flags and windsocks based on the current simulated wind speed and direction on terrains. It can also simulate movement of flags and windsocks on models added through connections or programmatically (this is not supported for props or external references). These features require models with appropriate switches and articulations, as well as model-specific or terrain-specific metafiles.

For generic information about the metafile format and syntax, please see section 8.6, Transforming Nodes Programmatically, in *VR-Forces Front-End Developers Guide*.

The metafiles for windsocks and flags have sections that configure wind control of nodes, as follows:

- Control switch nodes based on wind speed.
- Control the heading (Z-Axis) of articulating (degree of freedom) nodes based on wind direction.

The metafile used for wind features supports two features — WindSwitchPart and WindDirPart.

- WindSwitchPart. The WindSwitchPart feature changes the state of a switch node based on the current wind speed. The flag models used by MAK include a switch with three states. This feature has the following parameters:
  - **WindSpeed**. A list of wind speeds (in km/h).
  - **WindState**. A list of the switch state values that correspond to the values in the **WindSpeed** parameter.

  The number of items in each list should be the same and only contain switch states supported by the node specified in the `Linkage` entry.
- WindDirPart. The WindDirPart feature modifies the heading value (Z-Axis) of an articulation (degree of freedom) node based on wind direction. It has the following parameters:
  - **RotOffsetDeg**. Allows a rotation offset to be applied.
  - **RotDir**. Indicates if the node should be rotated in a clockwise or counter-clockwise direction.

Metafiles used to configure flags and windsocks include *./data/Terrain/Hawaii/Geometry/OpenFlightAlaMonana/AlaMoanaGeocentric.meta*, *./data/Structures/Poles/FlagPole/FlagPoleFlag.meta*, and *./data/Structures/Poles/WindSock.meta*.

90.7. Using Texture Atlases to Skin Models

When VR-Forces renders a vehicle model it takes into account information about the geometry of the vehicle, which specifies the dimensions, shape and so on, and textures, which specify the surface characteristics (in other words, how the vehicle looks.) If several models have the exact same geometry, but only differ in how they look (their textures), you can save disk space and some processing time by using the same geometry file for all of the models and a texture atlas to specify the textures to use for each model. (A texture atlas is a composite from multiple textures with the corresponding 3D models having uv coordinates shifted.)

For example, VR-Forces has four different versions of the Airbus 320 jet, one each for American Airlines, British Airways, Lufthansa, and Singapore Airlines. Instead of having four geometry files with associated textures, you can use one geometry file for all four models and reference the appropriate texture in a texture atlas.

Texture skinning requires a MEDF (geometry) file and a texture atlas that has been prepared for the entities that will share the texture atlas. The portions of the texture for a given model are accessed via UV offsets into the texture atlas. This information is stored in a metafile (*.meta*) that references a *.dtxc* file. You must also specify the metafile in the model definition for each model that is to use it and a configName parameter that is used to identify the correct texture to use.

90.7.1. Configuring Texture Mapping in the Model Definition

To use a texture atlas for a model, you must provide values for the metaFileName and configName parameters in the model definition.

To configure texture mapping in a model definition:

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 87-8.)
6. Select the model definition that you want to edit.
7. Click the Add Parameter button (). A list of parameters is displayed.
8. Select metaFileName. The metaFileName parameter is added to the model definition.

9. In the value column, specify the location of the metafile you want to use for this model.
10. Click the Add Parameter button and select configName. The **configName** parameter is added to the model definition.
11. In the value column, type a string that VR-Forces should look for in the metafile to identify the texture pattern to use.
12. Export the model definitions.

90.7.2. Metafiles for Texture Mapping

VR-Forces uses metafiles to configure texture skinning. For conceptual information about metafile format and syntax, please see section 8.6, Transforming Nodes Programmatically, in *VR-Forces Front-End Developers Guide*.

A metafile is associated with a terrain or model in one of the following ways:

- Place it in the same directory as the terrain or model's MEDF file. The metafile must have the same name as the MEDF file, but with a *.meta* extension.
- Specify it in the **metaFileName** parameter of the model definition used to import the model.

VR-Forces metafiles support two features that are relevant to texture mapping:

- UvAtlasMapper. Defines a texture atlas for texture skinning.
- SwitchStateMapper. Specifies the child of a switch node to use. For example, you could have an M1A1 with mine rollers, with mine plows, or DU armor, depending on the switch node used.

The following example shows the metafile for the Airbus 320 jet *Airbus-320a.meta*:

```
{
  "NodeFeatures": [
    {
      "Linkage": {
        "Comment": "Traversal with match up this feature based on
          NodeName match OR AnnotationName match",
        "NodeName": "",
        "AnnotationName": "@dis annotation texture"
      },
      "FeatureData": [
        {
          "FeatureName": "UvAtlasMapper",
          "AtlasFile": "$(DATA_DIR)\\Vehicles\\FixedWing\\Airbus320a\\
                       Airbus320a.dtxc",
          "Mappings": [
            {
              "ConfigName": "AmericanAirlines",
              "AtlasKeys": {
                "Pattern": "AmericanAirlines"
                }
            },
            {
              "ConfigName": "BritishAirways",
              "AtlasKeys": {
```

```
                    "Pattern": "BritishAirways"
                  }
                },
                {
                  "ConfigName": "Lufthansa",
                  "AtlasKeys": {
                    "Pattern": "Lufthansa"
                  }
                },
                {
                  "ConfigName": "SingaporeAirlines",
                  "AtlasKeys": {
                    "Pattern": "SingaporeAirlines"
                  }
                }
              ]
            }
          ]
        }
      ]
    }
```

When VR-Forces loads an Airbus 320, the **metaFileName** parameter in the model definition points it to the *Airbust320a.meta* file. The **configName** parameter in the model definition specifies which texture to use, for example, AmericanAirlines. VR-Forces reads *Airbust320a.meta* and associates the **configName** string, with the pattern value, also AmericanAirlines. Then it looks in *Airbus320a.dtxc* and gets the UV offset of AmericanAirlines so it knows which textures to use from the texture atlas.

The UV offsets for these models are configured in *Airbus320a.dtxc*. (The file contains explanatory comments.)

```
texture,COLPAT,AmericanAirlines,0.0,0.0
texture,COLPAT,Lufthansa,0.0,0.189612
texture,COLPAT,SingaporeAirlines,0.0,0.344823
texture,COLPAT,BritishAirways,0.0,0.500039
```

90.8. Adding Dynamically Changeable Markings to Models

Similarly to the way that you can use a texture atlas to configure texture skinning, you can use texture atlases to create dynamically changeable markings, such as country insignia and license plate numbers, on vehicle models. VR-Forces includes one model, the Fennek Armored Car, that has dynamically changeable markings (Figure 90-15).



Figure 90-15. Fennek armored car with different markings

To add markings to a model, you must:

1. Add `@dis annotation` switch attributes to the polygons on the model that you want to support markings.
2. Create a *.meta* file that specifies the UVAtlasMapper.
3. Create a *.dtxc* file for each set of markings that maps the annotation names to the coordinates of the associated textures.
4. Create a PNG file with the textures for each texture map.

To change the marking for a particular model, you can configure it in the model definition, as described in "Configuring Texture Mapping in the Model Definition," on page 90-22. You can also configure markings in the Simulation Object Editor and you can change the markings at runtime with the Set Markings set data request. (For details, please see "Specifying Visual Markings," on page 68-23 and "Markings," on page 34-27.)



All of the files for configuring markings on the Fennek are in *./data/Vehicles/Wheeled/FennekArmoredCar*.

90.8.1. Adding @dis annotations to the Model

In your modeling application, add @dis notes (part of the WRM Specification) to the polygons in your model. (For details about the WRM Specification, please see Chapter 92, *The WRM Specification (DIS Notes)*.)

Figure 90-16 shows the locations of polygons on the Fennek Armored Car for which annotations have been specified.

The annotations are in the format:

```
"@dis annotation annotation_name"
```

The specific annotations for the Fennek are:

```
"@dis annotation marking_platoon"
"@dis annotation marking_country"
"@dis annotation marking_id"
```

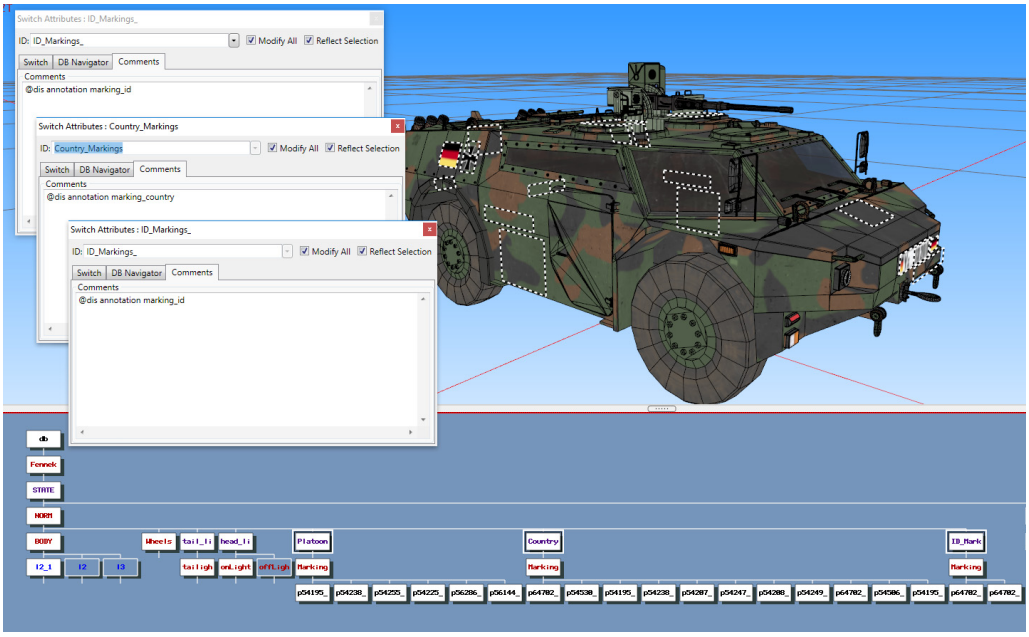


Figure 90-16. Specifying @dis annotations in modeling tool

90.8.2. The Metafile for Markings

As with texture skinning, you must create a metafile to configure markings. The following is a portion of the metafile for the Fennek (*./data/Vehicles/Wheeled/FennekArmoredCar/FennekArmoredCar.meta*). It shows some of the mappings for the `marking_country` annotation and the `marking_id` annotation.

```
{
  "NodeFeatures": [
    ...
    {
      "Linkage": {
        "LinkageName": "country",
        "Comment": "Traversal with match up this feature based on
            NodeName match OR AnnotationName match",
        "NodeName": "",
        "AnnotationName": "@dis annotation marking_country",
        "DisplayName": "Country Marking"
      },
      "FeatureData": [
        {
          "FeatureName": "UvAtlasMapper",
          "AtlasFile": "$(DATA_DIR)\\Vehicles\\Wheeled\\
              FennekArmoredCar\\CountryMarkings.dtxc",
          "Mappings": [
            {
              "ConfigName": "German",
              "AtlasKeys": {
                "Pattern": "German"
              }
            },
            {
              "ConfigName": "Canadian",
              "AtlasKeys": {
                "Pattern": "Canadian"
              }
            },
...
            }
          ]
        }
      ]
    },
    {
      "Linkage": {
        "LinkageName": "id",
        "NodeName": "",
        "AnnotationName": "@dis annotation marking_id",
        "DisplayName": "ID Marking"
      },
      "FeatureData": [
        {
          "FeatureName": "UvAtlasMapper",
          "AtlasFile": "$(DATA_DIR)\\Vehicles\\Wheeled\\
            FennekArmoredCar\\IDMarkings.dtxc",
          "Mappings": [
```

```
                {
                  "ConfigName": "03 20 01",
                  "AtlasKeys": {
                    "Pattern": "03 20 01"
                  }
                },
                {
                  "ConfigName": "21 12 02",
                  "AtlasKeys": {
                    "Pattern": "21 12 02"
                  }
                },
...
              ]
            }
          ]
        }
      ]
    }
```

90.8.3. The Texture Mapping File

For each type of marking you want to use, you need a *.dtxc* file. It maps the marking annotations to the UV coordinates of the textures in the texture file. The file *CountryMarkings.dtxc* maps the textures for the flags in *CountryMarkings_png.meif* (Figure 90-17). (Textures are created as PNG files. They are shipped in compressed format.)

```
marking_country,COLPAT,German,0.0,0.0
marking_country,COLPAT,Dutch,0.25,0.0
marking_country,COLPAT,Czech,0.5,0.0
marking_country,COLPAT,Canadian,0.75,0.0
marking_country,COLPAT,Swiss,0.0,0.25
marking_country,COLPAT,French,0.25,0.25
```

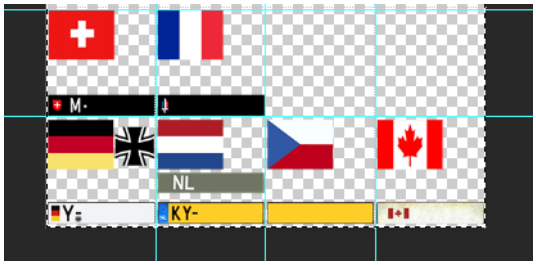


Figure 90-17. CountryMarkings textures

90.9. Flipping DDS Textures for a Model

If a model is displayed upside down because its DDS textures are not what VR-Forces expects, you can specify that the model be flipped. (For more information about DDS textures, please see "Displaying DDS Textures Correctly," on page 64-17.)

To flip DDS textures for a model:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 87-1).
3. Optionally, filter the model definitions in the list.
4. Select the model for which you want to change the DDS parameter.
5. If the model has the **flip DDS textures** parameter in its parameters list, set it to 1.

   If the model does not have the **flip DDS textures** parameter in its parameters list, click the Add Parameter button () to see if this is a valid parameter for this model. If it is, you can add the parameter and set it to 1.
6. Export the updated setting.

90.10. Configuring SpeedTree Trees

SpeedTree trees are high resolution models of trees. To improve performance, you can set clipping plane and level-of-detail (LOD) values to vary the resolution of the trees placed in the scene based on their distance from the observer. For details about clipping planes, please see "Setting the Clipping Planes," on page 82-9. Table 90-5 describes the settings you can change.

Table 90-5: SpeedTree settings

Parameter	Description
Clipping distance	The distance at which trees are clipped out of the scene.
Fade from High LOD to Low LOD	The range (near to far) in which high LOD trees are gradually replaced with low LOD trees.
Fade from Low LOD to Billboard	The range (near to far) in which low LOD trees are gradually replaced with billboard trees. This range is usually greater than the range for high LOD to low LOD.
Ambient Scalar	Multiplies the built-in ambient lighting value in each SpeedTree by the specified amount.
Diffuse Scalar	Multiplies the built-in diffuse lighting value in each SpeedTree by the specified amount.
Specular Scalar	Multiplies the built-in specular lighting value in each SpeedTree by the specified amount.

To configure SpeedTree trees:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the SpeedTree Settings page (Figure 90-18).

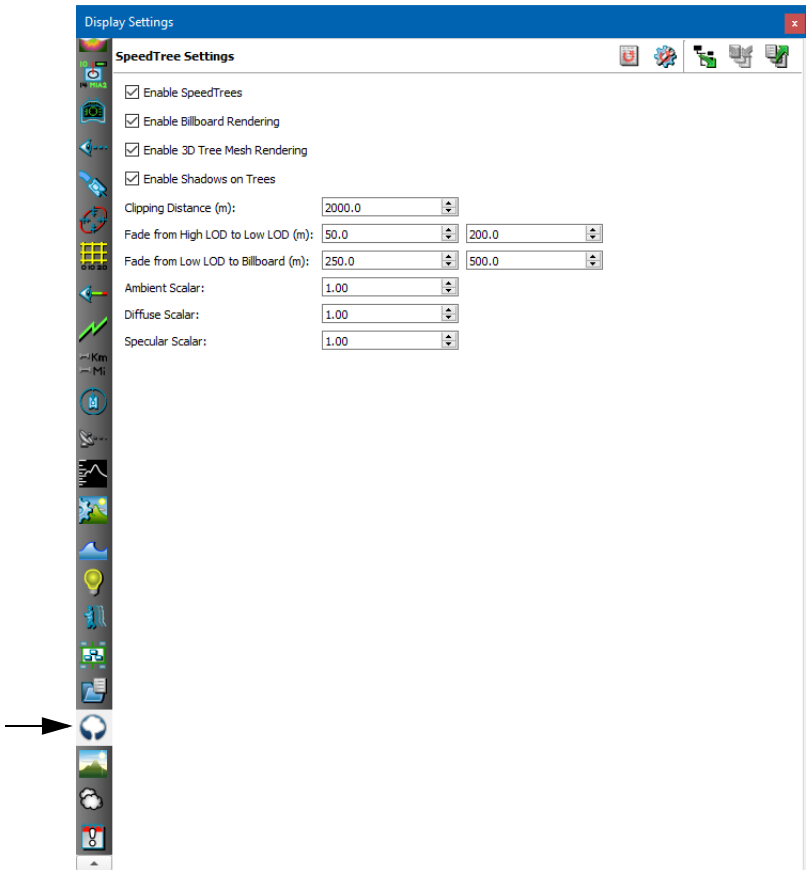


Figure 90-18. SpeedTree Settings page

3. Update parameters as desired.
4. Click Close.

90.10.1. Randomizing the Size of SpeedTrees

SpeedTrees can be created as props or streamed into the scene using an earth file. They have a default size. You can configure VR-Forces to scale the size of SpeedTrees by a fixed amount or you can scale them randomly within a specified range. The scale applied to a SpeedTree is the product of several factors.

If a SpeedTree is a prop, it is scaled as the product of the following settings:

- The `scale` attribute in its model definition. If the `scale` attribute is not part of the model definition, the value defaults to 1.0.
- A random scale factor as specified on the SpeedTree Randomization page on the Terrain Settings dialog box, if enabled.
- A feet-to-meters conversion factor of 0.3048.

If a SpeedTree is a streamed in using an earth file, it is scaled as the product of the following settings:

- The **scale** attribute in its model definition. If the scale attribute is not part of the model definition, the value defaults to 1.0.
- The **marker-scale** attribute in the `model` element that configures the trees.
- A random scale factor as specified on the SpeedTree Randomization page on the Terrain Settings dialog box, if enabled.
- A feet-to-meters conversion factor of 0.3048.

You can also randomize the orientation of SpeedTrees.

To specify randomization of SpeedTree scaling:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the SpeedTree Randomization page (Figure 90-19).

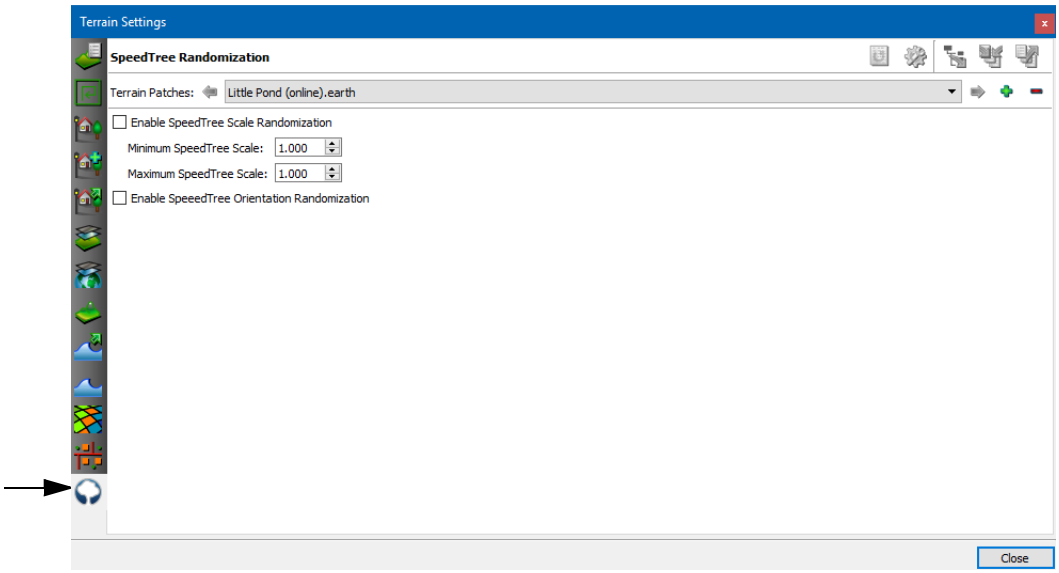


Figure 90-19. SpeedTree Randomization page

3. To enable randomization of the SpeedTree scale, select the Enable SpeedTree Scale Randomization check box.
4. If you enabled SpeedTree scale randomization, specify values for the Minimum SpeedTree Scale and Maximum SpeedTree Scale.
5. To enable randomization of tree orientation, select the Enable SpeedTree Orientation Randomization check box.

90.10.2. Displaying SpeedTrees in Planar Reflections

By default, SpeedTrees are not displayed in planar reflections. You can enable their display. For details, please see "Configuring Ocean Planar Reflections," on page 45-10.

90.11. Best Practices for Creating Models

If you build your own models for use in VR-Forces, following these guidelines will result in models that work efficiently:

- Keep all textures to powers of two, for example 256x256, 128x512, or 2048x2048:
  - Make sure textures use as many of the pixels as possible and that there are no large black areas in the texture.
  - Keep alpha (transparency) to a minimum.
- Minimize the number of textures:
  - Create a texture atlas that uses multiple textures combined into one.
  - Group polygons together that use the same texture.
  - Use the same textures for all LOD nodes.
- Minimize the number of nodes:
  - Keep the hierarchy as flat as possible.
  - Move all LOD nodes to the top of the hierarchy.
  - Minimize unnecessary transforms.
  - When you reuse geometry in the model, use instancing.
- Minimize the number of polygons that have different attributes:
  - Use the same material on polygons whenever possible.
  - Use two polygons with flipped normals instead of using the double sided attribute.
  - Use the same color on polygons whenever possible.
  - Group polygons with similar attributes.
- Consider using a more aggressive LOD structure:
  - Reduce the switchout distance to be smaller so that the LODs switch sooner.
  - Minimize the number of LODs.
  - Make sure that models switch out completely at a reasonable distance.
- Reduce the number of external references:
  - Merge external reference files.
  - Create instances from external merged reference files.

90.12. Disabling Indirect Rendering for an OpenFlight Model

In general, indirect rendering improves performance without affecting the quality of models. However, there may be cases where using indirect rendering causes artifacts in models. If there is a part of a terrain that rendered well in VR-Forces 4.6.1 or earlier, but does not render well with indirect rendering, you can disable indirect rendering for the OpenFlight model.

To be sure that indirect rendering is the cause of the problem, disable indirect rendering in the Add Terrain Patch dialog box. If the problem goes away, then you can disable indirect rendering for the affected portion of the model.

To disable indirect rendering within the model, add the tag `@dis DisableIndirect` to the comment for the appropriate node in the OpenFlight model. Indirect rendering will be disabled for that node and all of its child nodes. The rest of the model will continue to benefit from indirect rendering.

90.13. Compressing Model Files

The models and image files provided with VR-Forces are shipped in compressed formats (MAK Encrypted Data Format (MEDF) and MAK Encrypted Image Format (MEIF)). You may want to compress your models to save space or protect intellectual property. You can convert supported formats to MEDF and MEIF with *./bin64/medfTool.exe*. The compression tool supports the 3D modeling formats supported by VR-Forces, such as OpenFlight and 3DS and raster formats such as RGB and PNG.



When VR-Forces processes a request to load a a file, such as an external reference in a terrain, it checks the cache and the directory of the requested file for an MEDF version of the file. If it does not exist, it loads the file in the original format.

The syntax for medfTool is as follows:

```
medfTool [-b -d --directory directory --extendLodsEqual
   distance --extendLodsGreaterOrEqual distance
   --extendLodsTo distance -f filename --filter paths -h -i
   --genDDS -m extensions --norecurse -o -p <integer>
   -q simple_filename -r #comment --removeHighestLod
   -s directory --scaleLod scalar -t -- -V -v -x extensions
   -z optimization]
```

Table 90-6 describes the arguments.

Table 90-6: medfTool arguments

Argument	Description
(-- | --ignore_rest)	Ignore all arguments after this one.
(-b | --ignoreNonBase-Tex)	Ignores any non-base texture (diffuse texture) for a model. This is used when you do not want any multiple diffuse textures or any effect textures associated with it.
(-d | --dds_flip)	Flip DDS images before encrypting them.
--directory *directory*	Specifies the directory to search for image and geometry files to convert into MEDF and MEIF. All subdirectories are visited unless the --norecurse command line option is used.
--extendLodsEqual *distance*	Match LODs with switch-in distance equal to this argument and extend them to the value from the --extendLodTo argument
--extendLodsGreater-OrEqual *distance*	Match LODs with switch-in distance equal to or greater than this argument and extend them to the value from the --extendLodsTo argument
--extendLodsTo *distance*	The LOD switch-in distance that --extendLodsEqual and --extendLodsGreaterOrEqual extend to.
(-f | --file) *filename*	Convert just this file. The extension of the file must be specified using -x or -m to indicate it is an image or a geometry file.
--filter *paths*	Regular expression to match for filtering directories. Matches are not recursed into. Accepted multiple times.
--genDDS	Generate DDS files.
(-h | --help)	Displays usage information.
(-i | --ignore_errors)	Ignore errors in the encryption process and continue without exiting.
(-m | --image_exten-sions) *extension*	Specifies the extensions of image formats which will be converted into MEIF format. Use this argument to encrypt a directory. Can be used multiple times.
--norecurse	Do not recurse through subdirectories. Default: recurse through subdirectories.

Table 90-6: medfTool arguments

Argument	Description
`(-o | --onlyIfNew)`	Only generates an encrypted file if the expected output file does not exist.
`(-p | --threads)` *threads*	Specifies the number of threads to run. Specifying zero results in the number of threads equaling the number of logical processors the host machine has.
`(-q | --forceOpaque)` *simple_filename*	Specifies a filename and extension, but no path, to force polygons with these filenames to be opaque. Accepted multiple times.
`(-r | --removeNodes)` *#comment*	Removes any any node tagged with the comment from the scene graph in a model. This is used in special cases to remove nodes that are not needed. Example, `-r #missionfunctiononly`.
`--removeHighestLod`	Removes the highest LOD under a group if it has more than one LOD.
`(-s | --search)` *directory*	Adds a directory to the search path when resolving external references. During processing, any external references not found are removed from the generated files. Files in the search path are not converted.
`--scaleLod` *scalar*	Scales the switch in/out distance of all the children under an osg::LOD node by the specified value.
`(-v | --version)`	Display version information and exit.
`(-V | --verbose)`	Verbose output.
`(-x | --extensions)` *extension*	Specifies the extensions of geometry file formats that will be converted into MEDF format. Use this argument to encrypt a directory. Can be used multiple times.
`(-z | --optimize)` *optimization*	Specifies the optimization level. Maximum optimization may provide better optimization than the default setting, but it is significantly slower. Possible values are: ◆ 0 = None. ◆ 1 = Default. ◆ 2 = Maximum optimization.

The following example recursively compresses all FLT, RGB, RGBA, PNG, and DDS files in *%MAK_DATADIR%/Terrain*:

```
./bin64/medfTool.exe
  --directory %MAK_DATADIR%/Terrain --extensions flt
  --image_extensions rgb --image_extensions rgba
  --image_extensions png --image_extensions dds
```

The following example compresses all FLT, RGB, RGBA, PNG, and DDS files in *%MAK_DATADIR%/Lifeforms*. It does not compress files in subdirectories.

```
./bin64/medfTool.exe
  --directory %MAK_DATADIR%/Lifeforms --norecurse
  --extensions flt --image_extensions rgb
  --image_extensions rgba --image_extensions png
  --image_extensions dds
```



91. Configuring Emitter Volumes

This chapter explains how to configure display of emitter volumes. For information about how to configure simulation of emitters, please see Chapter 79, *Configuring Emitters and Radios*.

Configuring Emitter Volumes 91-2
Configuring Emitter Volume Color 91-2
Controlling Emitter Volume Radius 91-3
Configuring Emitter Volume Segments 91-4

91.1. Configuring Emitter Volumes

You can configure how VR-Forces colors emitter volumes, how they are drawn, and how they are scaled for visibility. You can configure emitter volumes by editing the DefaultEllipsoidArc model definition or by creating customized ellipsoid arc model definitions and applying them to different entity models.



In order for an entity to display emitter volumes in 2D observer modes and 3D observer modes, it must have a 2D Emitter Volume visualizer in its 2D model set and 3D Emitter Volume visualizer in its 3D model sets. The DefaultEllipsoidArc model definition is an attribute of the visualizer.

91.1.1. Configuring Emitter Volume Color

By default, emitter volumes are displayed using fixed colors for high frequency and low frequency. You can configure emitter volumes to be colored based on the emission frequency or pulse rate. When emitter volume color represents frequency, color is determined by the frequency field of the EM Emission update message. In both cases, you specify a low frequency or pulse rate value and a high value.

If the emitter frequency or pulse rate is lower than your specified low value, the emitter volume is red. As frequency rises above the specified low value, emitter volume color changes, progressing through the colors of the visible spectrum until the specified high value is reached. At the high value and above, the emitter volume is violet.

To change coloring from the default, you must add the appropriate attributes to the emitter volume visualizer for the entities whose model definitions you want to change. Table 91-1 describes the emitter volume visualizer attributes that affect emitter volume color.

Table 91-1: Emitter volume visualizer attributes

Attribute	Description
colorByFrequency	If True, color by frequency instead of using static colors.
lowFrequency	The low frequency threshold for coloring by frequency. Default: 800 MHz.
highFrequency	The high frequency threshold for coloring by frequency. Default: 1 GHz.
usePulseFrequency	If colorByFrequency is True and usePulseFrequency is True, color emitter volumes by pulse frequency.

Table 91-1: Emitter volume visualizer attributes

Attribute	Description
lowPulseFrequency	The low pulse rate threshold for coloring by frequency. Default: 100 Hz.
highPulseFrequency	The high pulse rate threshold for coloring by frequency. Default: 3 KHz.

91.1.2. Controlling Emitter Volume Radius

A emitter volume's radius (that is, the distance it extends from the emitter) is based in part on emitter power and in part on elevation and azimuth sweep data obtained from EM-Emission state messages. A higher emitter power value results in a longer radius, while a wider field of view results in a shorter radius (because the radiated power is distributed over a wider area).

If VR-Forces relied exclusively on these values, the range of sizes among different emitter types could become too wide to be practical, with some emissions being barely visible, and other emissions covering the entire virtual world. Therefore, a configurable exponent weight and scale factor have been added to help fine tune the emitter beam radii. The radius exponent weight (a value between 0 and 1) slows the growth of the radius when power or area is greater than 1 and increases the growth of the radius when power or area is less than 0. The scale factor increases the radii of all beams by the given value.

To change the scale factor and exponent from the default, you must add one or both of the following attributes to the emitter volume visualizer for the entities whose model definitions you want to change:

- linearScale. The amount of linear scaling applied to all emitter volumes. Default: 100.
- exponentialScale. The amount of exponential scaling applied to emitter volumes. Default: 0.1.

How VR-Forces Calculates the Radius

VR-Forces calculates emitter volume radius by approximating the spread of the emissions based on the elevation and azimuth sweeps. If the approximate coverage is zero, it sets the value to 1. Then it applies the linear and exponential scaling values. The code for implementing this is as follows:

```
double tempVar = 8*elevationSweep*sin(2*azimuthSweep);
if (tempVar <=0) tempVar=1.0;
scale = linearScale * pow(sqrt(power/tempVar), exponentialScale );
```

91.1.3. Configuring Emitter Volume Segments

VR-Forces displays an emitter volume as a group of spherical sections projecting from the sensing device. By default, for each 10 degrees (or portion thereof) in the emitter's field of view, VR-Forces draws one segment. Figure 91-1 shows segments set to 10° (the default) and 30°.

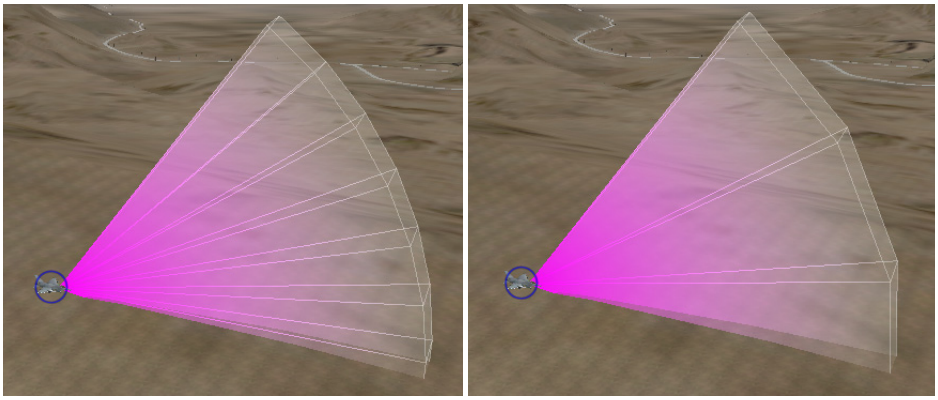


Figure 91-1. Emitter volume segments

Emitter volume segments are managed by the EllipsoidArc schema and model definitions based on it. VR-Forces ships with one ellipsoid arc model definition – DefaultEllipsoidArc. You can edit this model definition or create copies of it and customize them. If you want to have different emitter volumes for different entities, assign a different ellipsoid arc model definition to each one.

Specifying the Segment Angle

To specify the segment angle:

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Click the Filter button (▾) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
6. Select DefaultEllipsoidArc (Figure 91-2).

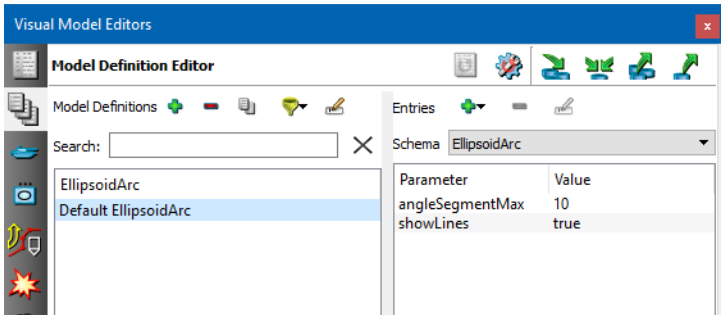


Figure 91-2. EllipsoidArc model definition

7. Change the value of the **angleSegmentMax** parameter. The minimum value is 10 (degrees); the maximum is 45.
8. Export the updated setting to the loaded definition file or to a separate file.

Displaying Segment Outlines

By default, VR-Forces displays opaque outlines showing the segment edges along the surface of the emitter volume.

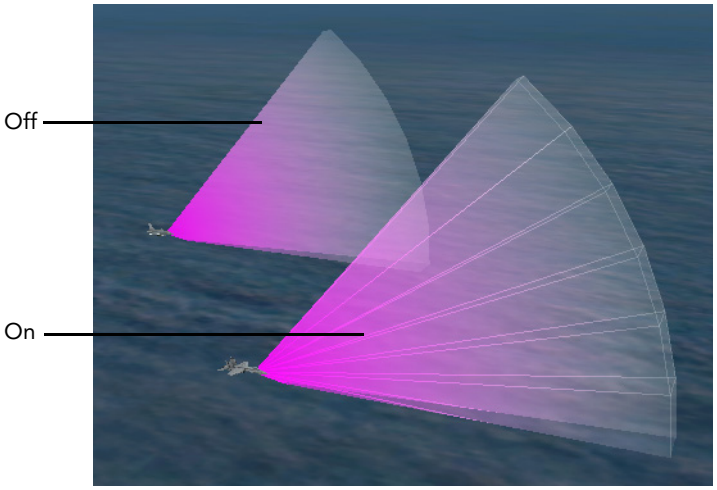


Figure 91-3. Segment outlines

To display or hide segment outlines:

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Click the Filter button () and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
6. Select DefaultEllipsoidArc.
7. Change the value of the showLines parameter.
8. Export the updated setting to the loaded definition file or to an individual file.

Adding a Custom EllipsoidArc Model Definition

If you want to customize the segment angle and segment outlines for a particular entity without changing the values for other entities, you must create a separate ellipsoid arc model definition and add it to the entity's model definition.

To create a custom ellipsoid arc model definition:

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, please see "Work Flow Issues for Model Definitions," on page 87-3.)
2. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 87-1).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions," on page 87-12.)
5. Click the Filter button () and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
6. Select DefaultEllipsoidArc.
7. Click the Duplicate Model Definition button (). The Duplicate Model Definition dialog box opens with a default definition name.
8. Accept the generated name or type a new name for the model definition.
9. Click OK. The model definition is added to the list.
10. Change the parameter values for the new model definition as desired.
11. Select the Entity Definition Editor page.

12. Select the entity whose emitter volume you want to change.
13. Select the 3D Sensor Volume visualizer for the entity.
14. Click the Value column for the modeldefinition attribute. The Choose Model Definition dialog box opens. It has a list of model definitions.
15. Select the model definition that you just created.
16. Click OK.
17. Export the updated setting to the loaded definition file or to an individual definition file.
18. If you are connected to an exercise, disconnect and then reconnect to enable the new model definition.



92. The WRM Specification (DIS Notes)

Some OpenFlight files contain information specified by the *WRM Entity Model Specification for DIS Interoperability*. This information makes it possible for VR-Forces to depict the state of an entity's articulated and attached parts correctly and to switch among different entity appearances.

This chapter contains information that can help you build entity models that comply with the WRM specification, or add that compliance to existing models.

The WRM specification was developed by MAK Technologies (now VT MAK), Paradigm Simulation Inc., MultiGen Inc., and SAIC, for the ARPA War Breaker project. (Paradigm Simulation and MultiGen merged and were later acquired by Presagis.)



Although the WRM specification was developed for DIS, it is also used in the HLA RPR FOM.

Introduction 92-2
The OpenFlight File Format 92-3
- Node Name and Comment Fields 92-3
- External References and Instancing 92-3

Modeling for DIS Interoperability 92-4
- The DIS Attribute Lexicon (DAL) 92-4
- DAL Keywords 92-5
- Model Coordinate Systems 92-6

Articulated Parts 92-8
Entity Appearances 92-9
Animation 92-12
DIS Keyword Values 92-13

92.1. Introduction

In order for visualization applications such as VR-Forces to change the state of a model in response to network messages, they must have certain information about the model. For example, which piece of the model's geometry is associated with which articulated part, and which piece of a model is associated with a particular entity appearance attribute.

This chapter describes a method of encoding information in an OpenFlight format model so that applications can make these associations. For the most part, information is associated with a MultiGen node by including it as a comment associated with the node. Comments that contain this information begin with the token, `@dis`.

VR-Forces depicts the movement of articulated parts and changes in entity appearances only when using models constructed to this specification.

This chapter represents a subset of the Model Specification for DIS Interoperability (version 001). For more details, please refer to that document.

Although VR-Forces implements only a subset of the functionality enabled by the WRM Model Specification, we recommend that models conform to as much of the specification as is appropriate for the model, to promote compatibility with future versions of VR-Forces, and with visualization applications from other vendors.

92.2. The OpenFlight File Format

The OpenFlight file format organizes geometry into the following types of nodes:

- Face nodes represent individual polygons.
- Object nodes represent face nodes that are logically or geographically related. Object nodes can have only face nodes (polygons) as children.
- Level of detail (LOD) nodes provide a means to select alternate geometries based on a user-definable range. Using lower resolution geometry for objects that are far from the observer saves memory and improves performance.
- Degree of Freedom (DOF) nodes specify translational or rotational freedom, or both, for portions of a model. They can be used for DIS articulated parts or animated special effects. They specify a type of transformation and its limits, such as "rotate about the x axis between 45 and 135 degrees". The OpenFlight approach to degrees of freedom is hierarchical.
- Group nodes represent logically-related or geographically-related object nodes. Group nodes are the most general, and may have any type of nodes, except face nodes, as children.

92.2.1. Node Name and Comment Fields

Each node contains a comment field for user-definable data. The WRM Specification defines keywords that, when placed in a comment field, provide the information that DIS applications need to properly render a model.

92.2.2. External References and Instancing

The following OpenFlight constructs provide a means to avoid duplicating massive amounts of geometry:

- An external reference (or xref) is a pointer to another OpenFlight file that contains additional geometry.
- A transformation matrix is usually attached to an xref to locate the position and orientation of the referenced object's coordinate system geometry. Each instance may have a transformation matrix controlling the position and orientation of the child's coordinate system.

92.3. Modeling for DIS Interoperability

The WRM Specification provides uniform standards for locating and orienting the model coordinate system and a way to map DIS codes to associated models and parts. By conforming to these standards, applications that support the WRM Specification do not need customized software to interpret the multiple potential model formats that might otherwise exist.

92.3.1. The DIS Attribute Lexicon (DAL)

DIS attributes are specified using the following syntax:

```
@dis keyword values # comment
```

where:

- `@dis` must precede the keyword.
- *`keyword`* is one of the DAL keywords.
- values can be a number or string. Depending on the requirements of the keyword, the values specified can be an individual value, a comma separated list, or a range.
- `#` indicates the start of a comment. Comments are case sensitive. A comment continues until the next `@dis` entry.

Remember that all comments will be parsed by VR-Forces, so spelling counts.

92.3.2. DAL Keywords

The WRM Spec supports the following keywords:

- `articulated_part` *part*. Identifies the movable parts of a model with a unique DIS Articulated Part code. This keyword is placed in a Group node that is a child to the DOF node that identifies an articulated part, and above the geometry that is going to be articulated. This keyword is always followed by the motion keyword. The articulated part codes are specified in DIS Standard SISO-REF-010-2006 Enumerations and Bit Encoded Values Articulated Parts. They are listed in Table 92-12.

  This keyword is always followed by the switch keyword.

- `switch` *appearance*. Identifies a model appearance attribute with a DIS keyword. This keyword is placed in a Switch node in the model above the geometry that is going to be switched.

  The switch keyword takes the name value in DIS Standard SISO-REF-010-2006 Enumerations and Bit Encoded Values Entity Appearance (General and Specific), joined by the underscore character (_) if the name is multiple words, as its value. Note that frozen_status, power_plant_status, and state are preceded by an abbreviation for their domain, as follows: gm_ for guided munitions and lf_ for life forms. For a list of supported values, please see Table 92-6.

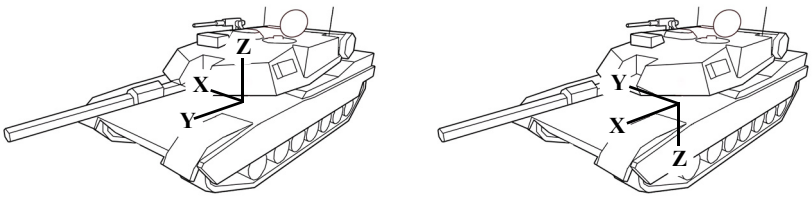
- `state` *state*,*state*... . Specifies the state of the appearance attribute with a DIS appearance state code. This keyword is placed in a Group node just below the switch node. For a list of supported values, please see Table 92-6.

92.3.3. Model Coordinate Systems

OpenSceneGraph (OSG) and the DIS Standard use different right-hand coordinate systems. Table 92-1 describes the two systems. Figure 92-1 illustrates them.

Table 92-1: Model coordinate systems

Coordinate	OpenSceneGraph	DIS
X	Right	Forward
Y	Forward	Right
Z	Up	Down



OSG coordinate system DIS coordinate system

Figure 92-1. Model coordinate systems

When you create models, it is recommended that you use the OpenSceneGraph coordinate system. The simulation application will convert the coordinates, as follows:

- xmodel = yDIS.
- ymodel = xDIS.
- zmodel = -zDIS.

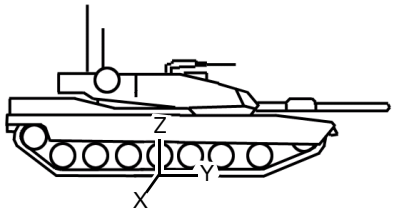
Location of the Origin for Models

The origin of the model's coordinate system should be located as follows, depending on the domain of the entity:

- Land: On the base of the vehicle, on the axis of yaw rotation.
- Surface: On the axis of the center of buoyancy, in the plane of the waterline for unloaded ship.
- Subsurface: On the axis of the center of buoyancy, in the plane of the waterline for surfaced submarine.
- Air and Space: At the center of gravity.

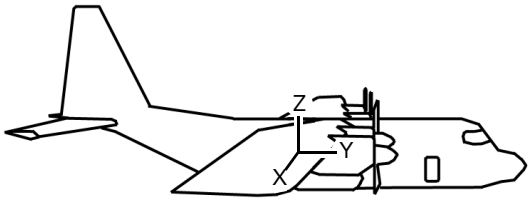
Figure 92-2 illustrates the location of a model's origin for each platform domain.

Land



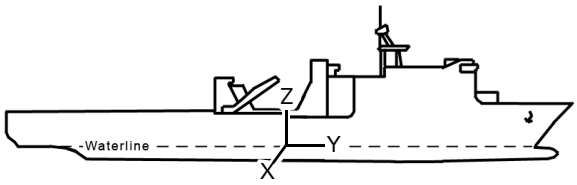
On the plane of ground contact, at the axis of yaw rotation.

Air



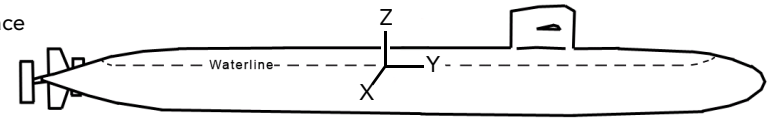
At the center of gravity.

Surface



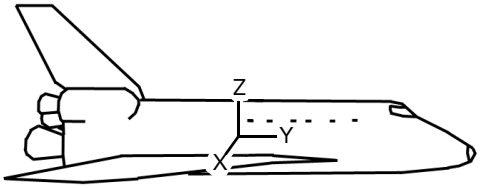
On the axis of the center of buoyancy, in the plane of the waterline for an unloaded ship

Subsurface



On the axis of the center of buoyancy, in the plane of the waterline of a surfaced submari

Space



At the center of gravity.

Figure 92-2. Location of origin by domain

92.4. Articulated Parts

In order for VR-Forces to properly depict the state of articulated parts, models must conform to the following specifications:

- There must be one DOF node for each articulated part. (This means that if there are several LODs, they should be children of the DOF.) Each articulation DOF should be a descendant of the DOF for the part to which it is attached (unless the parent part is just the base model itself). For example, the DOF for a gun should be a descendant of the DOF for a turret.
- Each articulated part DOF node must have all of the geometry and other nodes associated with the part underneath it. This DOF node must have a comment associated with it. The comment must be of the form:

  `@dis articulated_part` *`part-number`*

  where *`part-number`* is the enumeration for the particular type of part that this part of the model represents. For example, the part number of a turret is 4096, and the part number of a gun is 4416.

  Therefore, the child group of a turret's DOF must be commented with:

  ```
  @dis articulated_part 4096
  ```

  The child group of a gun's DOF must be commented with:

  ```
  @dis articulated_part 4416
  ```

- The WRM Model Specification specifies that information about the legal motions of each articulated part be encoded in the comment field. Currently, VR-Forces does not use this information.
- Do not include static offsets in the DOF attributes. That is, when you set all of the DOF parameters (X, Y, Z, heading, pitch, roll) to 0, the model should appear in its neutral position.

92.5. Entity Appearances

VR-Forces supports switching between different pieces of a model's geometry based on appearance attributes. For instance, a tank's turret should look different if the hatch is open or closed. An entire tank should look different if it is damaged or destroyed.

Switch nodes are used to represent a choice between several different children, based on a particular appearance attribute.

We define a switch node as a group node with a comment of the form:

```
@dis switch attribute
```

where *attribute* is the name of the attribute that should be used to select a child. (For example, damage, smoke, hatch, and so on.)

The children of the switch node represent the different portions of a model among which VR-Forces will select, based on the appearance attribute. If a switch node is commented with:

```
@dis switch hatch
```

it may, for example, have three children containing geometry for a closed hatch, popped hatch, and open hatch.

These children of the switch node must be commented so that VR-Forces knows which child matches which values of the appearance attribute. The comment is of the form:

```
@dis state value-list
```

`value-list` is of the form *m[,m]+* where *m* is one of the following:

- An integer representing the enumeration of a value that an appearance attribute may take on, for example, 1 for hatch closed.
- Two integers, a and b, separated by a hyphen, representing all enumerations from a through b, for example, 2-5 for all states whose enumerations are 2, 3, 4, or 5.
- The integer -1, which indicates that this child is the default child, to be used when the attribute state matches no values in any children.

For instance, the child of the hatch switch node that represents a closed hatch might be used when the hatch attribute value is either 0 (not-applicable), 1 (closed), or any value other than those enumerated in other children. This child would be commented:

```
@dis state 0,1,-1
```

The child with the popped hatch geometry should be used when the value is 2 (popped) or 3 (popped with person visible) and is commented:

```
@dis state 2,3
```

The child that contains the open hatch geometry should be used when the value is 4 (open) or 5 (open with person visible) and is commented:

```
@dis state 4,5
```

There may be many switch nodes for the same attribute. For example, there may be several pieces of a tank's geometry that are affected by the destroyed attribute.

The WRM Model Specification also discusses the inheritance of appearance attributes from ancestor nodes. VR-Forces does not support this functionality. To tell VR-Forces to, for example, display a camouflaged tank when the paint scheme attribute is set to camouflage, and a normal tank otherwise, you must use the switch node method described previously.

In Figure 92-3, the turret DOF node is attached to the body Group node. The barrel, gun, and hatch DOF nodes are attached to the turret. This allows the turret to be positioned in the coordinate system of the body, and the barrel, gun, and hatch to be positioned in the coordinate system of the turret.

This model also contains a switch node that is flagged with animation. The comment on this node is `@dis switch moving`. This switches between two Group nodes – Stop and Go. Under the Stop node there is a single mesh of the treads of the tank. Under the Go node there is a Group node, with the animation flag set on. This creates a flipbook animation of all nodes below this Group node. There are three meshes in different states of moving: Frame 1, Frame 2 and Frame 3. At runtime OpenSceneGraph will animate these meshes when the velocity is not 0.

If multiple color schemes are needed to represent camouflage, winter, and desert versions, another switch is needed.

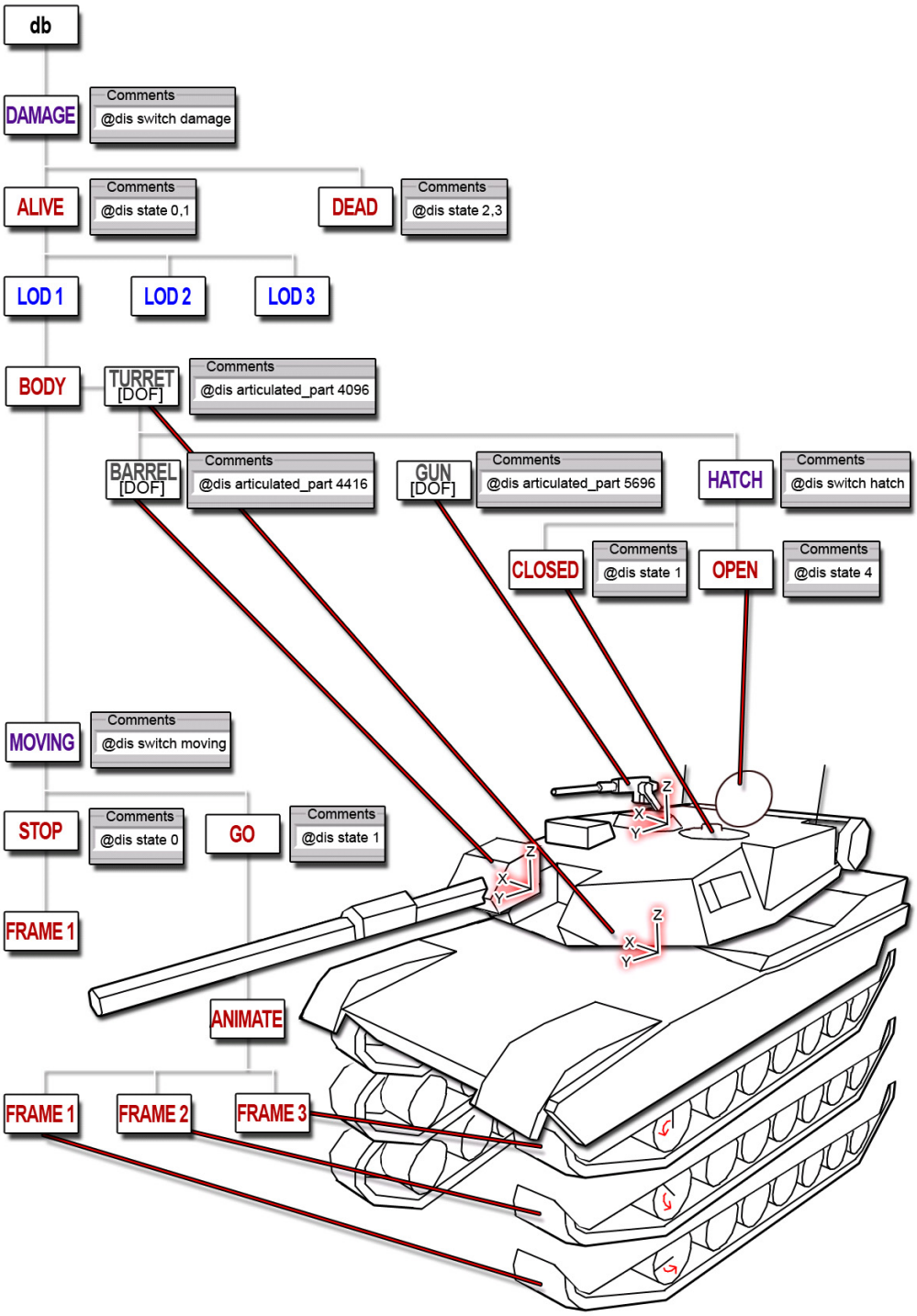


Figure 92-3. 3D model structure

92.6. Animation

By default, animation sequences are displayed as quickly as possible. However, you can configure the animation sequence rate using a WRM animation comment in the animation node's comment field.

The comment syntax is as follows:

```
@dis animation num_frames_per_second [ skip | noskip ]
```

where:

- `animation` tells VR-Forces that animation instructions follow.
- *`num_frames_per_second`* is the rate at which VR-Forces should display the frames.
- `skip` and `noskip` indicate whether the elapsed time should be used to choose the next frame, or whether at most one frame should be advanced each frame.

You can also specify how many times to loop through an animation and how long to hold the last frame with the following comments:

- `@dis loopCount` *`count`*, where *`count`* is the number of times to loop through the animation. Default: -1 (loop forever). 0 means do not loop.
- `@dis lastFrameTime` *`time`*, where *`time`* specifies how long, in seconds, to hold the last frame of the animation.

The `loopCount` and `lastFrameTime` comments only work with non-instanced models. You must use them with an `animation` comment, for example:

```
@dis animation 2 skip @dis loopCount 3 @dis lastFrameTime 11.5
```

92.7. DIS Keyword Values

Table 92-6 through Table 92-11 list the values that you can use for the `@dis switch` keyword by domain. For any switch keyword, the supported state values (`@dis state [value 1],[value 1]...`) are listed in the State column. The Bit column lists the appearance bit for setting DIS appearances. (Some bit number are repeated because they are valid only for a particular platform.)

- DIS appearance and states - land domain
- DIS appearance and states - air domain
- DIS appearance and states - surface domain
- DIS appearance and states - subsurface domain
- DIS appearance and states - space domain
- DIS appearance and states - lifeform domain
- DIS appearance and states - munition domain
- DIS appearance and states - environmental domain
- DIS appearance and states - cultural feature
- DIS appearance and states - sensor domain
- DIS Articulations

Table 92-2: DIS appearance and states - land domain (Sheet 1 of 3)

Values	Description	State/ Description	Bit
blackout_brake_lights	State of blackout brake lights.	0 = Off 1 = On	27
blackout_lights	State of blackout lights.	0 = Off 1 = On	26
brake_lights	State of brake lights.	0 = Off 1 = On	14
camouflage_type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused	17-18
concealed	Type of concealment.	0 = Not Concealed 1 = In Concealed Position	19
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
fire_power	Characteristics of fire-power kill.	0 = No 1 = Yes	2

Table 92-2: DIS appearance and states - land domain (Sheet 2 of 3)

Values	Description	State/ Description	Bit
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
hatch	State of the primary hatch.	0 = Not Applicable 1 = Closed 2 = Popped 3 = Popped and Person Visible 4 = Open 5 = Open and Person Visible 6 = Custom1 7 = Custom2	9-11
head_lights	State of headlights.	0 = No 1 = Yes	12
interior_lights	State of interior lights.	0 = Off 1 = On	29
launcher	State of the platform's primary launcher.	0 = Not Raised 1 = Raised	16
masked_cloaked	Describes whether or not the entity is masked or cloaked.	0 = No 1 = Yes	31
mobility	Characteristics of mobility kill.	0 = No 1 = Yes	1
paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage	0
power_plant_status	State of the entity’s power-plant.	0 = Off 1 = On	22
ramp	State of a ramp.	0 = Down 1 = Up	25
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6
spot_lights	Describes whether spot-lights are on or off.	0 = Off 1 = On	28

Table 92-2: DIS appearance and states - land domain (Sheet 3 of 3)

Values	Description	State/ Description	Bit
tail_lights	State of tail lights.	0 = Off 1 = On	13
tent	State of a tent extension.	0 = Not Extended 1 = Extended	24
trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large	7-8

Table 92-3: DIS appearance and states - air domain (Sheet 1 of 2)

Values	Description	State/ Description	Bit
afterburner	State of an air platform's afterburner.	0 = Off 1 = On	16
anticollision_lights	State of anti-collision lights.	0 = Off 1 = On	14
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
formation_lights	State of formation lights.	0 = Off 1 = On	24
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
hatch	State of the primary hatch.	0 = Not Applicable 1 = Closed 2 = Popped 3 = Popped and Person Visible 4 = Open 5 = Open and Person Visible 6 = Custom1 7 = Custom2	9-11
interior_lights	State of interior lights.	0 = Off 1 = On	29

Table 92-3: DIS appearance and states - air domain (Sheet 2 of 2)

Values	Description	State/ Description	Bit
landing_lights	State of landing lights.	0 = Off 1 = On	12
lights	State of lights.	0 = Off 1 = On	
mobility	Characteristics of mobility kill.	0 = No 1 = Yes	1
navigation_lights	State of navigation lights.	0 = Off 1 = On	13
paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage	0
power_plant_status	State of the entity's power-plant.	0 = Off 1 = On	22
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6
spot_lights	Describes whether spot-lights are on or off.	0 = Off 1 = On	28
trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large	7-8

Table 92-4: DIS appearance and states - surface domain (Sheet 1 of 2)

Values	Description	State/ Description	Bit
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
interior_lights	State of interior lights.	0 = Off 1 = On	29

Table 92-4: DIS appearance and states - surface domain (Sheet 2 of 2)

Values	Description	State/ Description	Bit
lights	State of lights.	0 = Off 1 = On	
mobility	Characteristics of mobility kill.	0 = No 1 = Yes	1
paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage	0
power_plant_status	State of the entity's power-plant.	0 = Off 1 = On	22
running_lights	State of running lights.	0 = Off 1 = On	12
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6
spot_lights	Describes whether spot-lights are on or off.	0 = Off 1 = On	28
trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large	7-8

Table 92-5: DIS appearance and states - subsurface domain (Sheet 1 of 2)

Values	Description	State/ Description	Bit
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
hatch	State of the primary hatch.	0 = Not Applicable 1 = Closed 2 = Popped	9-11

Table 92-5: DIS appearance and states - subsurface domain (Sheet 2 of 2)

Values	Description	State/ Description	Bit
		3 = Popped and Person Visible 4 = Open 5 = Open and Person Visible 6 = Custom1 7 = Custom2	
mobility	Characteristics of mobility kill.	0 = No 1 = Yes	1
paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage	0
power_plant_status	State of the entity's power-plant.	0 = Off 1 = On	22
running_lights	State of running lights.	0 = Off 1 = On	12
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6

Table 92-6: DIS appearance and states - space domain (Sheet 1 of 2)

Values	Description	State/ Description	Bit
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
mobility	Characteristics of mobility kill.	0 = No 1 = Yes	1
paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage	0

Table 92-6: DIS appearance and states - space domain (Sheet 2 of 2)

Values	Description	State/ Description	Bit
power_plant_status	State of the entity's power-plant.	0 = Off 1 = On	22
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6

Table 92-7: DIS appearance and states - lifeform domain (Sheet 1 of 2)

Values	Description	State/ Description	Bit
camouflage_type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused	17-18
compliance	Compliance of lifeform.	0 = Lifeform None 1 = Detained 2 = Surrender 3 = Using Fists 4 = Verbal Abuse 1 5 = Verbal Abuse 2 6 = Verbal Abuse 3 7 = Passive Resistance 1 8 = Passive Resistance 2 9 = Passive Resistance 3 10 = Non-Lethal Weapon 1 11 = Non-Lethal Weapon 2 12 = Non-Lethal Weapon 3 13 = Non-Lethal Weapon 4 14 = Non-Lethal Weapon 5 15 = Non-Lethal Weapon 6	
concealed	Type of concealment.	0 = Not Concealed 1 = In Concealed Position	19
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4

Table 92-7: DIS appearance and states - lifeform domain (Sheet 2 of 2)

Values	Description	State/ Description	Bit
flashlight	State of lights.	0 = Off 1 = On	
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
life_form_state	Lifeform action.	0 = None 1 = Upright Still 2 = Upright Walking 3 = Upright Running 4 = Kneeling 5 = Prone 6 = Crawling 7 = Swimming 8 = Parachuting 9 = Jumping 10 = Sitting 11 = Squatting 12 = Crouching 13 = Wading 14 = Surrender 15 = Detained	
paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage	0
weapon_1	State of primary weapon.	0 = None 1 = Stowed 2 = Deployed 3 = Firing Position	
weapon_2	State of secondary weapon.	0 = None 1 = Stowed 2 = Deployed 3 = Firing Position	

Table 92-8: DIS appearance and states - munition domain

Values	Description	State/ Description	Bit
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
launch_flash	Describes the presence of a guided munitions launch flash.	0 = No 1 = Yes	
masked_cloaked	Describes whether or not the entity is masked or cloaked.	0 = No 1 = Yes	31
power_plant_status	State of the entity's power-plant.	0 = Off 1 = On	22
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6
trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large	7-8

Table 92-9: DIS appearance and states - environmental domain

Values	Description	State/ Description	Bit
density	Density of environmental.	0 = Clear 1 = Hazy 2 = Dense 3 = Very Dense 4 = Opaque 5 = Unused	
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
masked_cloaked			

Table 92-10: DIS appearance and states - cultural feature

Values	Description	State/ Description	Bit
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
exterior_lights	Describes whether the exterior lights are on or off.	0 = Off 1 = On	28
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
interior_lights	State of interior lights.	0 = Off 1 = On	29
internal_heat_status	Describes whether the internal heat is on or off.	0 = Off 1 = On	22
masked_cloaked	Describes whether or not the entity is masked or cloaked.	0 = No 1 = Yes	31
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6

Table 92-11: DIS appearance and states - sensor domain (Sheet 1 of 2)

Values	Description	State/ Description	Bit
antenna_raised	Describes whether the antenna is raised or not	0 = Lowered 1 = Raised	16
blackout_lights	State of blackout lights.	0 = Off 1 = On	26
concealed	Type of concealment.	0 = Not Concealed 1 = In Concealed Position	19
camouflage_type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused	17-18

Table 92-11: DIS appearance and states - sensor domain (Sheet 2 of 2)

Values	Description	State/ Description	Bit
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
frozen_status	Frozen status of an entity.	0 = Not Frozen 1 = Frozen	21
interior_lights	State of interior lights.	0 = Off 1 = On	29
lights	State of lights.	0 = Off 1 = On	
mission_killed	Ability to carry out mission.	0 = No 1 = Yes	2
mobility	Characteristics of mobility kill.	0 = No 1 = Yes	1
paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage	0
power_plant_status	State of the entity's power-plant.	0 = Off 1 = On	22
smoke	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6
tent	State of a tent extension.	0 = Not Extended 1 = Extended	24
trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large	7-8

Table 92-12 lists the supported values for the `@dis articulated_part` keyword. String values are part of the WRM specification, but are not currently supported.

Table 92-12: DIS Articulations (Sheet 1 of 5)

Value	String	Description
3296	bridge_launcher	Position of the bridge launcher.
3328	bridge_section_1	Position of bridge section 1.
3360	bridge_section_2	Position of bridge section 2.
3392	bridge_section_3	Position of bridge section 3.
7616	fuselage_fold	Fold on plane's body.
1184	helicopter_main_rotor	Describes main rotor rotation.
1216	helicopter_tail_rotor	Describes tail rotor rotation.
3072	landing_gear	Landing gear on a plane.
3520	launcher_arm_primary	Position of the primary launcher arm.
1120	left_aileron	Position of the left aileron.
1056	left_flap	Position of the left flap.
3168	left_weapon_bay_door	Left door to weapon bay.
3552	other	Position of other articulated parts.
3424	primary_blade_1	Position of the primary blade 1.
3456	primary_blade_2	Position of the primary blade 2.
3488	primary_boom	Position of the primary boom.
5056	primary_defense_systems_1	Position of the primary defense systems 1.
5088	primary_defense_systems_2	Position of the primary defense systems 2.
5120	primary_defense_systems_3	Position of the primary defense systems 3.
5152	primary_defense_systems_4	Position of the primary defense systems 4.
5184	primary_defense_systems_5	Position of the primary defense systems 5.
5216	primary_defense_systems_6	Position of the primary defense systems 6.
5248	primary_defense_systems_7	Position of the primary defense systems 7.
5280	primary_defense_systems_8	Position of the primary defense systems 8.
5312	primary_defense_systems_9	Position of the primary defense systems 9.
5344	primary_defense_systems_10	Position of the primary defense systems 10.
4416	primary_gun_number_1	Position of the primary gun number 1.
4448	primary_gun_number_2	Position of the primary gun number 2.
4480	primary_gun_number_3	Position of the primary gun number 3.
4512	primary_gun_number_4	Position of the primary gun number 4.

Table 92-12: DIS Articulations (Sheet 2 of 5)

Value	String	Description
4544	primary_gun_number_5	Position of the primary gun number 5.
4576	primary_gun_number_6	Position of the primary gun number 6.
4608	primary_gun_number_7	Position of the primary gun number 7.
4640	primary_gun_number_8	Position of the primary gun number 8.
4672	primary_gun_number_9	Position of the primary gun number 9.
4704	primary_gun_number_10	Position of the primary gun number 10.
4736	primary_launcher_1	Position of primary launcher 1.
4768	primary_launcher_2	Position of primary launcher 2.
4800	primary_launcher_3	Position of primary launcher 3.
4832	primary_launcher_4	Position of primary launcher 4.
4864	primary_launcher_5	Position of primary launcher 5.
4896	primary_launcher_6	Position of primary launcher 6.
4928	primary_launcher_7	Position of primary launcher 7.
4960	primary_launcher_8	Position of primary launcher 8.
4992	primary_launcher_9	Position of primary launcher 9.
5024	primary_launcher_10	Position of primary launcher 10.
5376	primary_radar_1	Position of the primary radar 1.
5408	primary_radar_2	Position of the primary radar 2.
5440	primary_radar_3	Position of the primary radar 3.
5472	primary_radar_4	Position of the primary radar 4.
5504	primary_radar_5	Position of the primary radar 5.
5536	primary_radar_6	Position of the primary radar 6.
5568	primary_radar_7	Position of the primary radar 7.
5600	primary_radar_8	Position of the primary radar 8.
5632	primary_radar_9	Position of the primary radar 9.
5664	primary_radar_10	Position of the primary radar 10.
4096	primary_turret_number_1	Position of the primary turret number 1.
4128	primary_turret_number_2	Position of the primary turret number 2.
4160	primary_turret_number_3	Position of the primary turret number 3.
4192	primary_turret_number_4	Position of the primary turret number 4.
4224	primary_turret_number_5	Position of the primary turret number 5.
4256	primary_turret_number_6	Position of the primary turret number 6.
4288	primary_turret_number_7	Position of the primary turret number 7.

Table 92-12: DIS Articulations (Sheet 3 of 5)

Value	String	Description
4320	primary_turret_number_8	Position of the primary turret number 8.
4352	primary_turret_number_9	Position of the primary turret number 9.
4384	primary_turret_number_10	Position of the primary turret number 10.
1152	right_aileron	Position of the right aileron.
1088	right_flap	Position of the right flap.
3200	right_weapon_bay_doors	Right door to the weapon bay.
7584	rotor_fold	Fold on helicopters rotors.
1024	rudder	Position of the rudder.
6656	secondary_defense_systems_1	Position of the secondary defense systems 1.
6688	secondary_defense_systems_2	Position of the secondary defense systems 2.
6720	secondary_defense_systems_3	Position of the secondary defense systems 3.
6752	secondary_defense_systems_4	Position of the secondary defense systems 4.
6784	secondary_defense_systems_5	Position of the secondary defense systems 5.
6816	secondary_defense_systems_6	Position of the secondary defense systems 6.
6848	secondary_defense_systems_7	Position of the secondary defense systems 7.
6880	secondary_defense_systems_8	Position of the secondary defense systems 8.
6912	secondary_defense_systems_9	Position of the secondary defense systems 9.
6944	secondary_defense_systems_10	Position of the secondary defense systems 10.
6016	secondary_gun_number_1	Position of the secondary gun number 1.
6048	secondary_gun_number_2	Position of the secondary gun number 2.
6080	secondary_gun_number_3	Position of the secondary gun number 3.
6112	secondary_gun_number_4	Position of the secondary gun number 4.
6144	secondary_gun_number_5	Position of the secondary gun number 5.
6176	secondary_gun_number_6	Position of the secondary gun number 6.
6208	secondary_gun_number_7	Position of the secondary gun number 7.
6240	secondary_gun_number_8	Position of the secondary gun number 8.

Table 92-12: DIS Articulations (Sheet 4 of 5)

Value	String	Description
6272	secondary_gun_number_9	Position of the secondary gun number 9.
6304	secondary_gun_number_10	Position of the secondary gun number 10.
6336	secondary_launcher_1	Position of the secondary launcher 1.
6400	secondary_launcher_3	Position of the secondary launcher 3.
6432	secondary_launcher_4	Position of the secondary launcher 4.
6464	secondary_launcher_5	Position of the secondary launcher 5.
6496	secondary_launcher_6	Position of the secondary launcher 6.
6528	secondary_launcher_7	Position of the secondary launcher 7.
6560	secondary_launcher_8	Position of the secondary launcher 8.
6592	secondary_launcher_9	Position of the secondary launcher 9.
6624	secondary_launcher_10	Position of the secondary launcher 10.
6368	secondary_launcher_2	Position of the secondary launcher 2.
6976	secondary_radar_1	Position of secondary radar 1.
7008	secondary_radar_2	Position of secondary radar 2.
7040	secondary_radar_3	Position of secondary radar 3.
7072	secondary_radar_4	Position of secondary radar 4.
7104	secondary_radar_5	Position of secondary radar 5.
7136	secondary_radar_6	Position of secondary radar 6.
7168	secondary_radar_7	Position of secondary radar 7.
7200	secondary_radar_8	Position of secondary radar 8.
7232	secondary_radar_9	Position of secondary radar 9.
7264	secondary_radar_10	Position of secondary radar 10.
5696	secondary_turret_number_1	Position of the secondary turret number 1.
5728	secondary_turret_number_2	Position of the secondary turret number 2.
5760	secondary_turret_number_3	Position of the secondary turret number 3.
5792	secondary_turret_number_4	Position of the secondary turret number 4.
5824	secondary_turret_number_5	Position of the secondary turret number 5.
5856	secondary_turret_number_6	Position of the secondary turret number 6.
5888	secondary_turret_number_7	Position of the secondary turret number 7.
5920	secondary_turret_number_8	Position of the secondary turret number 8.
5952	secondary_turret_number_9	Position of the secondary turret number 9.
5984	secondary_turret_number_10	Position of the secondary turret number 10.

Table 92-12: DIS Articulations (Sheet 5 of 5)

Value	String	Description
3104	tail_hook	Tail hook on a plane for aircraft carrier landings.
7584	wing_fold	Fold on plane's wings.



XV. Appendixes

The appendixes contain supplementary information.



A. MTL Syntax

This chapter describes the syntax of VR-Forces MTL files.

Introduction A-2
MAK Technologies Lisp (MTL) Files A-3
Using Environment Variables in an MTL File A-3
Specifying Lists of Lists A-4
Using Conditional Statements A-4

A.1. Introduction

VR-Forces uses many different configuration files to manage element definitions, model mapping, back-end configurations, menus, and so on. Almost all of the configuration files for the front-end are XML files that get created and updated through the Display Settings dialog box, Terrain Settings dialog box, and the other front-end configuration dialog boxes. Although you could edit these files with text editors, we recommend that you not edit them by hand and we do not document their contents and syntax. When you save settings, they are saved to files with the correct extension.



Developers may need to create and edit configuration files using the VR-Forces Front-End API. Please see *VR-Forces Front-End Developers Guide* and the class documentation for details.

The back-end configuration files mostly use the MAK Technologies LISP (MTL) syntax. The major back-end configuration file is *vrfSim.mtl*. It is described in Appendix A, *MTL Syntax*. Other files are described throughout this manual in the appropriate section.

A.2. MAK Technologies Lisp (MTL) Files

Many of the configuration files for MAK products are ASCII text files that use MAK Technologies LISP (MTL) to encode configuration information. MTL files have the extension *.mtl*. You can edit them in any text editor. Be sure that your text editor uses the proper end-of-line characters for your platform.

Most parameter entries take the format:

```
(setq parameter_name value)
```

For example:

```
(setq EnableWireFrame 1)
```

Or:

```
(setqb parameter_name value)
```

The setqb syntax indicates a bound variable.

To comment out a line, precede the text with two semi-colons (;;). You can also add a comment at the end of a line by preceding it with two semi-colons.

In most cases, the parameter names are known to the MAK application and the MTL commands simply assign values to them. However, you can use the `setq` command to create arbitrary symbolic names or aliases that are then used in later commands. For example, in the following two commands, the symbolic name VaporTrail is assigned to an OpenFlight file. Then the symbolic name is used in the `trail-map` command. The name VaporTrail has no intrinsic meaning. It is given meaning by the `setq` command, which allows it to be used later in the `trail-map` command.

```
(setq VaporTrail (list "./../data/models/makTrails/Vapor-
  Trail" "VaporTrail.flt" 6.0 1.5 1.5 4.0 1.5 0.5))
(trail-map (list 1 2 -1 0 -1 -1 -1)(list VaporTrail 0.0 0.0
  0.0))
```

In addition to the `setq` command, a commonly used command is the `load` command. This command instructs the application to load the file specified, for example:

```
(load mtl-path "params.mtl")
```

A.2.1. Using Environment Variables in an MTL File

If you want to use an environment variable in an MTL file, use the following syntax:

```
(setqb parameter_name (getenv (quote env_var)))
```

For example:

```
(setqb disDestAddr (getenv (quote DEST_ADDR)))
```

A.2.2. Specifying Lists of Lists

The example of setting a symbolic name in a previous section uses a list. You can also specify a list of lists, for example:

```
(trail-map (list 1 2 -1 -1 -1 -1 -1)
   (list vaporTrail -8.0 -3.0 0.0)
   (list vaporTrail -8.0 3.0 0.0)
)
```

A parameter entry can also have multiple individual lists:

```
(trail-map (list 1 2 -1 0 -1 -1 -1)(list VaporTrail 0.0 0.0
  0.0))
```

A.2.3. Using Conditional Statements

MTL commands can be conditional. Conditional statements are used frequently when parameters are different for DIS and HLA. A simple conditional statement has the format:

```
(if (condition) (statement_to_evaluate))
```

If you have more than one statement to evaluate, use a sequence block:

```
;; HLA specific parameters
(if (equal HLA 1)
   (sequence
      (setqb fomMapperLib "")
      (setqb fomMapperInitData "")
      (setqb rprFomVersion 1.0)
      (setqb pathToSublistFile "sublist.mtl")
      (setqb ignoreAdvisories 0)
      (setqb fedLookahead 1.0)
   )
)
```



B. The vrfSim.mtl Configuration File

This chapter describes the *vrfSim.mtl* file.

The vrfSim.mtl Configuration File B-2

B.1. The vrfSim.mtl Configuration File

The file *./appData/settings/vrfSim/vrfSim.mtl* configures back-ends. Table B-1 describes the parameters in *vrfSim.mtl*. The Description column includes the command-line argument equivalent for parameters that have them.

Table B-1: *vrfSim.mtl* parameters (Sheet 1 of 7)

Parameter	Description
additionalSystemScriptsDirectory	Directories other than the default script directory where VR-Forces should look for system scripts.
aggregateSpatialModelTickPeriodUsesRealTime aggregateSpatialModelTickPeriod aggregateSpatialModelTickVariance aggregateSpatialModelTickAsFastAsPossibleWhilePaused	Configure aggregate spatial model tick rates. Default is to run the aggregate spatial model every frame. Tick period is given in seconds, with the special value of -1 indicating tick every frame.
appNumber	Specifies the application number. Default: 3001. Command line: `-a | -p`
assertOnBlockingTerrainCalls	When set to true (1), causes blocking terrain calls to assert in debug mode if the simulation is running. Blocking terrain calls can cause long ticks in a running simulation which can have very negative effects on the models. This parameter can help developers find places in the code that are making the offending calls. Note that calling the blocking form of any terrain intersection function causes an assertion; it is not necessary for the call to actually block and even non-paging implementations will assert. This is to make it easier to find potential problems regardless of the particular terrain or location being used. Default: 0. For more information, please see API documentation.
batchScenarioFileName	Specifies a batch file to run. Command line: `-B`
cgfDispatcherReceivePort	Specifies the port to be used by the *DtCgfDispatcher's* file transporter to receive scenario objects and plan files from the *DtVrfRemoteController*.
countryCodesMappingFile	The file VR-Forces uses to import country codes to map to object type enumerations.
daemon	Runs vrfSim as a daemon process (Linux only.) vrfSim does not try to read characters from a console, and forks a background process on startup. It exits cleanly on receipt of the SIGTERM, SIGQUIT, SIGINT signals. Default: Off. Command line: `--daemon`

Table B-1: *vrfSim.mtl* parameters (Sheet 2 of 7)

Parameter	Description
datumShiftX datumShiftY datumShiftZ	These parameters, along with the spheroidSemiMajor and spheroidSemiMinor parameters, let you change the reference ellipsoid used for geocentric to geodetic conversions. For more information, please see Choosing a Reference Ellipsoid in *VR-Link Developers Guide*. These parameters are used only if VRF_useCustomMapDatum is set to 1.
defaultParameterDatabasePath	Specifies the location to search for the object parameter database if the full path is not specified in a scenario file.
defaultTerrainDatabasePath	Specifies the location to search for the terrain database if the full path is not specified in a scenario file.
diGuyAnimationsFile	Specifies the DI-Guy animations file to load if useDIGuy is 1.
disableParallelTick	If enabled (1), VR-Forces ticks thread safe components in parallel. If disabled (0), all components are ticked serially in the main thread.
doNotUseConsole	Disables (1) or enables (0) writing of output to the console. Default: 0.
enableLogFileTimestamps	If enabled, puts timestamps in front of each line in for all log files.
enableNavigationLabDebugging	Enables debugging using the Navigation Lab application. A connection is made on an available port, starting with 4888, to enable Navigation Lab to connect to the simulation engine.
entVrfDataTimeThreshold	Specifies how often, in seconds, VR-Forces environmental and simulation object data is sent. If you specify -1.0, the update is sent only if some item in the vrfDataObject has changed.
forceParameterDbReload	If set to 1 (default), forces the back-end to reload the object parameter database (OPD) each time a scenario is loaded. If set to 0, if a scenario is started that uses the same SMS as the previous one (for example, in a rewind), the SMS is not reloaded.
gamewareMemorySize	Set the working memory size limit used internally by GameWare when calculating paths. Size is in MB. Increasing this limit can allow for path plans on larger navigation areas by more entities simultaneously. Note: Multiple working memory blocks are allocated with this limit, so this is not an overall limit for Gameware.
gamewareQueryTimeBudget	Set the time budget for GameWare navigation queries per frame. This is the amount of time GameWare is given to perform tasks such as path planning each simulation frame. The higher the value, the more responsiveness path planning will be, but the slower the VR-Forces simulation frame rate will be. Default: 1.0 ms.

Table B-1: *vrfSim.mtl* parameters (Sheet 3 of 7)

Parameter	Description
loadPluginIfVersionMismatch	Specifies whether (1) or not (0) to load a plug-in if there is a mismatch between how the plug-in was built and how VR-Forces was built. Mismatches can be build type (release versus debug), operating system, or protocol (DIS, HLA 1.3, HLA 1516, HLA Evolved.) Allowing a plug-in to load if there is a mismatch can cause VR-Forces to be unstable.
logFrameRateStatistics	Enables (1) or disables (0) logging of frame rate statistics to a file.
maxDrainInputTime	Sets the maximum time to wait in drainInput() before returning. For the MAK RTI, the default value drains the entire input stream before returning. For the Pitch RTI, the default value reads a single input from the input stream, so set to a value other than -1 to process more information. Use with DIS or HLA.
maxNavigationRegenera-tionThreads	Sets the maximum number of threads that can be running at a given time regenerating GameWare navigation data. If you have many available CPUs and are using dynamic terrain, increasing this number will speed up the regenera-tion of navigation data due to changes in the terrain. If you are more limited in CPU availability, increasing this value may have a negative affect on VR-Forces performance. Default: 2.
msdlSIDCMappingFile	The file VR-Forces uses to map between SIDC codes and object type enumerations when it imports MSDL files.
navigationRegenerationDelay	Sets the delay (in seconds) between the occurrence of an event that triggers navigation data regeneration and the start of the regeneration. Decreasing this improves the responsiveness of navigation to dynamic terrain changes; however, it increases the likelihood that navigation data will be generated multiple times if multiple events result in navigation changes in a particular area. Default: 2.5
notifyLevel	Sets the notification level (0-4) for information and warning messages. Default: 2. Command line: `-n`
objectConsoleNotifyLevel	Specifies the notification level (0-4) for messages sent to the object console. Default: 1.
pluginsDirectory	Specifies the directory in which configuration (MTL) files for plug-ins are located. The directory must end with a slash, for example, *./plugins/*
publishZeroVelocityWhenPaused	When set to 1, the default, VR-Forces sets simulation object velocities to 0 when it is paused so that simulation objects do not continue to dead reckon. To publish the current velocity, set to 0.

Table B-1: *vrfSim.mtl* parameters (Sheet 4 of 7)

Parameter	Description
regenerateNavigation-DataAtRuntime	Enables navigation data to be regenerated at runtime based on changes in dynamic terrain. This may have a negative effect on performance as it requires spawning threads to perform the regeneration. However, it allows entities to correctly navigate around and through the changes in the terrain. Default: 1.
rejectMismatchedScenarioMes-sages	If true (the default) reject mismatched scenario run messages based on unique scenario run id.
reportedHostAddressString	Specifies how VR-Forces reports its IP address to another VR-Forces application for the purpose of file transfers. It is only important when you have a GUI or remote controller on one side of a router and a simulation engine on the other, and there is some kind of network address translation occurring whereby they have to connect to one another using an IP address other than the one they recognize themselves at locally.
reuseEntityIdentifiersOnScenari-oLoad	If true, reuse object identifiers that might be part of the scenario. If the object identifier saved in the scenario objects file does not currently match the site/app of the back-end loading that object, then the user will be warned and the object will be assigned a new entity identifier that is compliant with that site/app. This will not apply to simulation object groups or scenario import. They will always get new entity identifiers.
scenarioFileName	Specifies the scenario file to load. Command line: `-L`
sendBackendLogToNetwork	Enables (1) or disables (0) sending simulation engine log messages to the front-end.
sessionId	Specifies the session ID. Command line: `-i`
setDeadReckoningAlgorithmTo-StaticOnPause	When the simulation is paused, set the dead reckoning algorithm for aggregates and entities to static. If enabled and publishZeroVelocityWhenPaused is disabled, the objects in a VR-Forces run simulation show current values for velocity, speed, acceleration, and so on.
setMainThreadToHighPriority	If set to 1, sets the main thread to THREAD_PRIORITY_-TIME_CRITICAL in Windows. Using this option on a single core processor will probably cause deadlock. (Windows only.)
simMgmtPduProcessLevel	Specifies how simulation management PDUs are processed: 1 – Processes PDUs sent to a specific back-end. Wildcards are ignored. 2 – Processes only wildcarded PDUs. Ignores those sent to specific back-ends. 3 – Processes all PDUs.

Table B-1: *vrfSim.mtl* parameters (Sheet 5 of 7)

Parameter	Description
simulationModelSet	Specifies the default simulation model set or sets. If specifying multiple SMSs, separate them by a semi-colon.
siteId	Specifies the site ID. 0 = use internal default. Command line: `-s | -t`
spheroidSemiMajor spheroidSemiMinor	These parameters, along with the datumShift parameters, let you change the reference ellipsoid used for geocentric to geodetic conversions. For more information, please see Choosing a Reference Ellipsoid in *VR-Link Developers Guide*. These parameters are used only if VRF_useCustomMapDatum is set to 1.
startInRunMode	Specifies that the simulation start in run mode (1), rather than pause mode (0). Default: 0.
startMinimized	If true (1), the back-end minimizes at startup.
statusUpdateHeartbeat	Specifies how often, in seconds, the back-end status heartbeat is sent.
suppressDataAvailableException	Prevent terrain interface non-blocking intersect calls from throwing an exception if the dataAvailable flag is not checked.
targetFrameRate	Specifies the frame rate for the current frame above which the back-end will sleep, yielding CPU time to other processes. This option applies only to variable frame mode. For more information, please see "Tuning the Target Frame Rate," on page 6-22. Default: 22.
taskVisualizationPublicationMode	Specifies whether or not to publish task visualizations and how to publish them, as follows: 0 = Never publish task visualizations. 1 = Standard mode. The VR-Forces GUI will request task visualizations only for objects for which they are enabled. 2 = Always publish task visualizations, regardless of whether they are being displayed in the GUI. Can be useful for logging purposes.
terrainDatabase	Specifies the terrain database to load. This parameter is overridden by the scenario file if one is specified in scenarioFileName. Command line: `-T`
useCustomMapDatum	Allows modification of the reference ellipsoid used for geocentric to geodetic conversion. If you enable this parameter, there are several sub-parameters to configure. For details, please see the comments in the file.
useDrawControlForTaskVisualizations	If true, publish task visualizations as VR-Vantage draw control objects instead of their standard publication method. This allows task visualizations to be displayed in any VR-Vantage based application; however the VR-Forces GUI will no longer be able to filter task visualizations by entity.

Table B-1: *vrfSim.mtl* parameters (Sheet 6 of 7)

Parameter	Description
useDIGuy	Specifies whether (1) or not (0) to use DI-Guy characters.
HLA Parameters	For details, please see "Command-Line Options for HLA Federations," on page 5-19
destroyFedExec	Specifies whether (1) or not (0) the *DtExerciseConn* destructor should try to destroy the federation execution. Default: 1.
execName	The federation execution name. Command line: `-x`
federateName	The federate name. Command line: `-N`
federateType	Lets you specify the federate type for HLA Evolved.
fedFileName	Specifies the FED file name. Command line: `-F`
fomMapperLib	A FOM Mapper library. Command line: `-f`
fomMapperInitData	An initialization string. Command line: `--fomMapperInitData`
hostAddressString	Specifies the host address. This is usually the same as the device address.
localSettingsDesignator	Specifies the local settings designator for HLA Evolved.
mimModule	Specifies a MIM (MOM and Initialization) Module for HLA Evolved.
minDrainInputTime	The minimum time to stay in drainInput() before returning.
rprFomVersion	The RPR FOM version. (0.5, 0.8, 1.0, 2.0) Command line: `--rprFomVersion`
runInTimeManagementMode	Enables (1) or disables (0) use of time management.
sendFedTime	Enables (1) or disables (0) sending and receipt of interactions in time stamp order. Default: 0.
useAbsoluteTimeStamps	If set to 1, use absolute time stamps. If set to 0, use relative time stamps. Default: 0.
useAdvisories	Enables (1) or disables (0) HLA advisory messages.
DIS Parameters	Please see "Command-Line Options for DIS Exercises," on page 5-21.
additionalDestinationAddresses	Specifies additional destination addresses to send to. Accepts multiple IP addresses separated by semi-colons (;).
addMulticastAddr	The multicast address. Command line: `-S`
appIdRange	Specifies the application ID range to use to identify simulation objects generated by VR-Forces.
defaultHeartbeatThreshold	Specifies the DIS heartbeat threshold, in seconds. A heartbeat is sent when the threshold is reached, even if no data has changed.

Table B-1: *vrfSim.mtl* parameters (Sheet 7 of 7)

Parameter	Description
defaultObjectTimeout	Specifies the DIS object timeout threshold, in seconds. If VR-Forces does not receive a heartbeat for an object for the time period specified, it removes the object from the exercise.
destAddrString	Destination address for point-to-point communications mode. Command line: `-A`
deviceAddress	Specifies the address of the network card to use for UDP traffic.
disPort	The port number. Command line: `-P`
exerciseId	The exercise ID. Command line: `-x`
heartbeatVariance	When a simulation object is created, a random value between -heartbeatVariance and +heartbeatVariance is added to the heartbeat. This spreads heartbeats over time instead of them all being at the same time.
hostAddressString	Specifies the host address. This is usually the same as the device address.
maxDrainInputReads	The maximum number of reads to make in drain input before returning
mcastTtl	Specifies the multicast time to live. Command line: `--mcastTtl`
recvBufferSize	Specifies the receive buffer size, as a positive value. A value of -1 means use the system default. Command line: `--recvBufferSize`
sendBufferSize	Specifies the send buffer size, as a positive value. A value of -1 means use the system default. Command line: `--sendBufferSize`
subnetMask	When the destination address is not specified ("0.0.0.0"), the subnet mask is used in conjunction with the network interface address to determine the destination broadcast address. If neither is specified (subnet mask is blank), the network interface's default broadcast address is used.
suppressSelfReflect	Prevent VR-Forces from reflecting DIS PDUs.
useAbsoluteTimeStamps	If set to 1, use absolute time stamps. If set to 0, use relative time stamps. Default: 0.
useAsyncIO	Enables (1) or disables (0) use of asynchronous I/O, which can help reduce dropped packets under heavy network load. Default: 0. Command line: `-I`
useIpv6	Use the IPV6 protocol.



C. Object Parameters

This appendix describes the high-level parameters in the object parameter database. It also has reference material for local and remote subcomponents and object type enumerations.

The Structure of a Simulation Object Parameter C-2
The Parameter Type Hierarchy C-3
Vrf Base Object Parameters C-3
Vrf Object Parameters C-7
Moving Object Parameters C-8
Parameter Types for Entity Platforms C-8
Object Type Enumerations C-9
Object Super-type C-9
AggregateKind C-9
ObjectKind C-10
Unit Category C-10
Unit Specific C-11
Point Object Category C-12
Linear Object Category C-12
Areal Object Category C-12
Interaction C-12

C.1. The Structure of a Simulation Object Parameter

Many of the files in the object parameter database uses MAK Technologies LISP (MTL). (For details about MTL, please see "MAK Technologies Lisp (MTL) Files," on page A-3.) A parameter block has the form:

```
(block_name
  ( parameter_name value)
  ...
  (parameter_name
    (sub-parameter value)
  )
  and so on
)
```

Entries have the following requirements:

- The block_name can be anything that you like. It must not start with a number.
- The parameter_name must be exactly as documented (except that in some cases, a parameter may contain an arbitrary length list of sub-parameters, for which the parameter names are not used – these will be noted). Parameter names must not contain spaces, and must not start with a digit (or a dash followed by a digit, for example, -1).
- The first parameter in any parameter block must be **parameter-type**.
- You can include comments in the file. A comment begins with a semicolon(;) and continues to the end of the line.
- All numerical parameters use standard MKS values: meters, kilograms, seconds, radians, liters, and so on.
- Real constants must begin with a digit and contain a decimal point. For example, 0.5 and 0. are acceptable, but .5 and 0 are not.



.entity files use XML format, not MTL.

C.2. The Parameter Type Hierarchy

Figure 74-4, in "Parameter Types," on page 74-8, illustrates the hierarchy of parameter types. In this section, we describe the parameters associated with each parameter-type.

C.2.1. Vrf Base Object Parameters

All object parameters inherit from the parameter type vrf-base-object-param. The parameters associated with vrf-base-object-param are potentially applicable to any type of object (ground, rotary-wing, control object, and so on). In addition to parameters, each object has an object type and parameter type, as described in Table C-1.

Table C-1: Object and parameter type

Parameter	Description
object-type	(Required) 8-digit object type. Not actually a parameter. Specifies the key used to look up the entry. For more information, please see "Object Types," on page 74-4.
parameter-type	(Required) One of the following (string): ◆ "vrf-base-object-param" ◆ "vrf-object-param" ◆ "moving-object-param" ◆ "fixed-wing-entity-param" ◆ "ground-vehicle-param" ◆ "rotary-wing-entity-param" ◆ "missile-param" ◆ "surface-entity-param" ◆ "subsurface-entity-param" ◆ "human-param" ◆ "aggregate-object-param" ◆ "cultural-feature-param" ◆ "bomb-param" ◆ "global-environment-param" ◆ "circle-object-param" ◆ "point-object-param" ◆ "scripted-movement-object-param" ◆ "tactical-smoke-param". The value must be in quotation marks.

Local Objects Subcomponent Parameters

The local-objects parameter specifies the types of subcomponents to create for locally simulated objects. You can omit optional items, or set them with an empty string “”. Table C-2 lists the subcomponent parameters and their possible values.

Table C-2: Local-objects subcomponent parameters (Sheet 1 of 2)

Subcomponent parameter	Description
state-repository	(Required) Maintains the state information for the object. An object that has components (sensors, controllers, and actuators) for simulation sets and retrieves information in the repository. Some simulation components expect a specific kind of state repository. In general, you will want to specify the state repository type that most closely matches the kind of simulation object you expect to simulate. May be one of the following types: ◆ "vrf-object-base-state-repository": Used for the base object in the object parameter database. ◆ "vrf-object-state-repository": Derived from vrf-object-base-state-repository, it adds the concepts of resources, sector of responsibility, engagement rules, and targeting. Because there are more simulation object-specific state repository types that are derived from this type, this kind of repository is not typically specified. ◆ “vrf-overlay-object-state-repository”: For tactical graphics. ◆ "vrf-moving-object-state-repository": For any object that you expect to simulate movement. Units in entity-level scenarios have this kind of state repository. The more simulation object-specific repository types that follow inherit this kind of state repository, so choose one of those when simulating a specific type of individual entity. ◆ "ground-entity-state-repository": For ground vehicles, such as tanks, trucks, dismounted infantry. ◆ "rotary-wing-entity-state-repository": For rotary-wing aircraft. ◆ "fixed-wing-entity-state-repository": For fixed-wing aircraft. ◆ "missile-entity-state-repository": For missile and rocket entities. ◆ "surface-entity-state-repository": For surface entities. ◆ "subsurface-entity-state-repository": For subsurface entities. ◆ “human-state-repository”: For lifeform entities. ◆ “vrf-aggregate-state-repository”: For units. ◆ “cultural-feature-state-repository”: For cultural features that are created as entities.

Table C-2: Local-objects subcomponent parameters (Sheet 2 of 2)

Subcomponent parameter	Description
network-interface	(Optional) The network interface handles the publication of the object over the network. It acts as an interface between a simulation object's state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of simulation object you are simulating. May be one of the following types: ◆ "aggregate-local-entity-net-interface": For units. ◆ "pseudo-aggregate-local-entity-net-interface": For units that have subordinate simulation objects that are separately simulated. ◆ "individual-local-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. ◆ "local-environmental-net-interface": For simulation objects that are best represented as an environmental process. Tactical graphics use this type of network interface.
net-interface-min-tick-period	(Optional) Specifies the shortest period of time, in seconds, that can elapse between ticks of a simulation object's network interface.
net-interface-min-tick-period-variance	(Optional) Specifies an amount within which the net-interface-min-tick-period can vary. Defined as: plus or minus rand(net-interface-min-tick-period/2)
component-manager	(Optional) A component manager contains the lists of sensor, controller, and actuator components used to simulate the behavior of a given object (typically simulation objects). Expected value: "component-manager".
plan-manager	(Optional) The simulation object plan manager. If you want a simulation object to be able to execute a plan, it must have a plan manager.
task-manager	(Optional) If a component manager is specified for an object, then you must specify a task manager. The task manager works in conjunction with the component manager to manage the processing and message handling associated with execution of a task. Expected value for individual entities or units: "local-task-manager".

Remote Object Subcomponent Parameters

The remote-objects parameter specifies the types of sub-components to create for remote objects. Remote objects are those objects simulated by other applications on the network (including other VR-Forces applications). You can omit optional items, or set them with an empty string "". Table C-3 lists the subcomponent parameters and their possible values.

Table C-3: Remote object subcomponent parameters (Sheet 1 of 2)

Subcomponent parameter	Description
state-repository	(Required) Maintains the state information for the object. Unlike the local-object entry, it is not as important to specify the closest match based on type. For entities, the "vrf-moving-object-state-repository" is sufficient. A more specific state repository may be specified. However, some of the simulation object-specific data may not be set. May be one of the following types: ◆ "vrf-object-state-repository": Derived from vrf-object-base-state-repository, it adds the concepts of resources, sector of responsibility, engagement rules, and targeting. Because there are more simulation object-specific state repository types that are derived from this type, this kind of repository is not typically specified. ◆ "vrf-overlay-object-state-repository": For tactical graphics. ◆ "vrf-moving-object-state-repository": For any object that you expect to simulate movement, including individuals and units. ◆ "ground-entity-state-repository": For ground vehicles, such as tanks, trucks, dismounted infantry. ◆ "rotary-wing-entity-state-repository": For rotary-wing aircraft. ◆ "fixed-wing-entity-state-repository": For fixed-wing aircraft. ◆ "missile-entity-state-repository": For missile and rocket entities. ◆ “human-state-repository”: For lifeform entities. ◆ “vrf-aggregate-state-repository”: For units. ◆ “cultural-feature-state-repository”: For cultural features that are created as entities. ◆ "surface-entity-state-repository": For surface entities. ◆ "subsurface-entity-state-repository": For subsurface entities.

Table C-3: Remote object subcomponent parameters (Sheet 2 of 2)

Subcomponent parameter	Description
network-interface	(Optional) The network interface is the interface between the remote network (VR-Link) representation of the object and its state repository. It acts as an interface between a simulation object's state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of simulation object you are simulating. May be one of the following types: ◆ "vrf-aggregate-remote-entity-net-interface": For remote units. ◆ "vrf-individual-remote-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. ◆ "vrf-remote-environmental-net-interface": For simulation objects that are best represented as an environmental process. Tactical graphics use this type of network interface.
net-interface-min-tick-period	(Optional) Specifies the shortest period of time, in seconds, that can elapse between ticks of a simulation object's network interface.
net-interface-min-tick-period-variance	(Optional) Specifies an amount within which the net-interface-min-tick-period can vary. Defined as: plus or minus rand(net-interface-min-tick-period/2)
plan-manager	Do not specify a plan manager for remote objects. (Give it the value "" (empty string). If you specify a plan manager, it is ignored.)
component-manager	Do not specify a component manager for remote objects. (Give it the value "" (empty string). If you specify a component manager, it is ignored.)
task-manager	(Optional) Do not specify a task manager for remote objects. (Give it the value "" (empty string). If you specify a task manager, it is ignored.)

C.2.2. Vrf Object Parameters

The parameter type vrf-object-param inherits from vrf-base-object-param and is the parent of the moving object parameter type. Vrf object parameters include resources, engagement rules, and target acquisition time. The vrf-object-param parameter type represents an intermediate class in the parameter hierarchy. It is not usually used to represent objects in VR-Forces. You will probably want to instantiate a derived type, (for simulation objects) or you will want to specify vrf base object parameters (for case of control objects.)

Control objects have the parameter type vrf-base-object-param.

C.2.3. Moving Object Parameters

Moving objects have the parameter type moving-object-param. Moving objects inherit parameters from vrf objects and vrf base objects. The moving-object-param parameter type is used for objects that are capable of movement, such as simulation objects. Parameters include support points, maximum speed, maximum reverse speed, turning radius, maximum slope, ordered speed, maximum acceleration, maximum deceleration, and fuel efficiency. Units are moving objects and have this parameter type. All of the platform-specific parameter types derive from this one.

C.2.4. Parameter Types for Entity Platforms

Table C-4 lists the parameter types used by each entity type.

Table C-4: Parameter types for simulation objects

Platform	Parameter Type	Inherits from
Ground vehicle	ground-vehicle-param	moving-object-param
Fixed-wing	fixed-wing-entity-param	moving-object-param
Human	human-param	moving-object-param
Rotary-wing	rotary-wing-entity-param	moving-object-param
Surface	surface-entity-param	moving-object-param
Subsurface	subsurface-entity-param	moving-object-param
Cultural feature	cultural-feature-param	moving-object-param
Missile	missile-param	moving-object-param
Unit	aggregate-param	moving-object-param
Unit Container	moving-object-param	vrf-object-param

C.3. Object Type Enumerations

This section lists the object type enumerations as listed in the header file *objType-Enums.h*. It is presented here primarily for customers who have purchased VR-Forces, but not the toolkit. Therefore we have departed from the typical header file format for readability.

C.3.1. Object Super-type

The header file *objTypeEnums.h* contains the VR-Forces extensions to the DIS/RPR FOM enumerations. The enumerations specify individual, unit, and other object types. The sim object super-type is an enumeration used in the *DtObjectType* class (an extension of the *DtEntityType* class). It extends the DIS/RPR FOM entity type enumeration by specifying the sort of object associated with the *DtEntityType*. Because object types are not guaranteed to be unique across all of the possible individual and unit types, the extra enumerated field provides a way to uniquely identify all objects in a VR-Forces simulation.

VR-Forces uses the value 3 in the object super-type for all of the units that it simulates.

Other	0
Individual	1
Unit	2
Disaggregated Unit	3
Aggregated Unit	4
Interaction	5

C.3.2. AggregateKind

The enumeration for *DtAggregateKind* should start with 0, but VR-Forces starts with 10, to ensure that simulation object types will be unique.

Other	10
MilitaryHierarchy	11
CommonType	12
CommonMission	13
SimilarCapabilities	14
CommonLocation	15
DtSimArtPartKind	20

C.3.3. ObjectKind

Extension of entity kind to represent non-simulation object objects, such as control objects (control points, routes, phase lines, obstacles, and so on.)

DtPointObjectKind	16
DtLinearObjectKind	17
DtArealObjectKind	18
DtCircleObjectKind	19
DtTextObjectKind	20

C.3.4. Unit Category

VR-Forces extends the enumeration to include *DtAggregateCategoryForce*, which represents the force level. All of the entries from Force through the end of the list, are extensions to the standard.

Other	0
IndividualVehicle	1
Element	2
Platoon	3
Battery	4
Company	5
Battalion	6
Regiment	7
Brigade	8
Division	9
Corps	10
Force	11
Team	12
Squad	13
Section	14

Unit Subcategory

The Subcategory field enumerations for Kind field = Military Hierarchy specify the Echelon Type of the unit. Not all of the echelon types are applicable to each of the categories.

In the following list, ForceFriendly, ForceOpposing, ForceNeutral, and ForceOther are extensions to the DIS/RPR FOM enumerations.

Other	0	ArmyAviationFixedWing	13
CavalryTroop	1	ArmyAviationRotaryWing	14
Armor	2	ArmyAttackHelicopter	15
Infantry	3	AirCavalry	16
MechanizedInfantry	4	ArmorHeavyTaskForce	17
Cavalry	5	MotorizedRifle	18
ArmoredCavalry	6	MechanizedHeavyTaskForce	19
Artillery	7	CommandPost	20
SelfPropelledArtillery	8	CEWI	21
CloseAirSupport	9	TankOnly	22
Engineer	10	ForceFriendly	23
AirDefenseArtillery	11	ForceOpposing	24
AntiTank	12		

C.3.5. Unit Specific

The Specific field enumerations for Kind field = Military Hierarchy specify whether the unit contains a headquarters, and is enumerated as follows.

Noheadquarters	0
ContainsHeadquarters	1

C.3.6. Point Object Category

Subcategories for use with the Kind field = *DtPointObjectKind*.

Other	0
ControlPoint	1
TargetPoint	2
Marker	3

C.3.7. Linear Object Category

Subcategories for use with the Kind field = *DtLinearObjectKind*.

Other	0
PhaseLine	1
Route	2

C.3.8. Areal Object Category

Subcategories for use with the Kind field = *DtArealObjectKind*.

Other	0
ControlArea	1
Obstacle	2

C.3.9. Interaction

Kinds for Supertype interaction.

Other	0
Fire	1
Detonate	2



D. Deprecated Weapon System and Munition Damage Configuration

This appendix describes methods for configuring weapon systems used prior to release 4.7 and damage systems prior to release 4.5. For conceptual detail about weapon systems and munition damage, please see Chapter 76, *Configuring Weapon Systems*.

Configuring Direct Fire Weapons D-2
- Sensing Targets D-4
- Ammo Select Tables D-6
- Hit Probability Tables D-7
- Damage Probability Tables D-8

Configuring Indirect Fire Weapons D-9

Configuring Fixed Weapons D-11

Configuring Bomb Bay Weapon Systems D-12

Calculating Damage from Munitions D-13

Configuring Damage-by-Power D-14
- Configuring the Power of a Detonation D-15
- The Damage File for Damage-By-Power D-18
- Adding a New Power Type D-20

Configuring Damage-by-Ammo-Type D-22
- Configuring Damage for Damage-by-Ammo-Type D-23

Direct Fire Configuration Examples D-24
- Direct Fire Weapon Damage-By-Power Example D-25
- Direct Fire Weapon Damage-by-Ammo-Type Example D-30

D.1. Configuring Direct Fire Weapons

This section explains how direct fire weapons were configured in VR-Forces release 4.6.1 and earlier. This method of configuring systems is supported in VR-Forces 4.7 and later for legacy purposes.

Entities that have direct fire weapons can fire in response to a Fire at Target task and they can detect targets through their sensors and automatically fire at the targets. Entities have to be able to:

- Know what targets they can fire at.
- Detect the targets.
- Determine the ammunition to use.
- Acquire the target and fire at it.

If you want to create a completely new type of weapon system, you must write code for the sensors, controllers, and actuators that it requires. However, if you need a system that is similar to an existing system, you can often create the new system by copying and modifying the configuration files of the existing system. To do so, you must understand the relationships between the different configuration files to allow you to:

- Configure a weapon system.
- Specify the simulation object types that can use the system.
- Specify the simulation object types that can be damaged by the system.
- Specify the amount of damage a simulation object receives from a munition.

The files that manage these relationships are:

- The weapon system. Defines the system, the simulation object types that it can target, and the simulation object types that can use it.
- The ammo select file. Specifies the type of munition to use for each type of simulation object that a weapon can fire at.
- The damage system. Specifies the types of munitions that a simulation object is vulnerable to and the damage file to use to determine the probability of damage.
- The damage file. Specifies the probability that a simulation object will sustain damage and the type of damage it will sustain when hit by a particular munition in a particular location.
- The hit file. Specifies the probability that a ballistic gun will hit a target at different distances. The hit file is not used for projectile weapons and indirect fire.

For detailed examples of how to configure a direct fire weapon, please see "Direct Fire Weapon Damage-By-Power Example," on page D-25 and "Direct Fire Weapon Damage-by-Ammo-Type Example," on page D-30.

The weapon does the following:

1. Acquires the target.
2. Selects the ammunition for the target using the ammo select file.
3. Checks that it has that resource.
4. Determines if it hits target.
5. Sends a fire interaction.
6. After a delay based on distance and muzzle speed, it sends a detonation interaction.

D.1.1. Sensing Targets

Simulation objects that have direct fire weapons detect targets through their sensors and automatically fire at the targets. Sensor systems can specify what types of targets to detect (using the **detection-types** parameter). The weapon systems can also tell the sensors what they want to target.

Each weapon system has an associated weapon interface. The **targeting-control** parameter takes a list of desired target types and creates a target selection criteria object, which is then set in the weapon interface. The following code shows the targeting control for *handheld-AK47.sysdef* file. It also shows the ammo select file.

```
(main-gun-control
  (component-descriptor-type "ballistic-gun-ctlr-descriptor")
  (component-type  "ballistic-gun-controller")
  ...
  (ammo-select-table-file
    (filename  "$(ammoselect-dir)\AKA762.asl")
  )
  (rounds-per-trigger-pull  5)
  (targeting-control
    (target-priorities
      (entity-priority
        (entity-type  3 1 -1 1 -1 -1 -1)
        (priority  1)
      )
      (entity-priority
        (entity-type  3 -1 -1 -1 -1 -1 -1)
      (priority  2)
    )
  )
  ...
)
```

The platform file for the entity uses its target selection controller to iterate through each weapon interface to create a list of object types to detect, which it outputs on its object-types-to-detect port.

Each sensor on the entity connects to this port to know what types of objects to detect (Figure D-1).

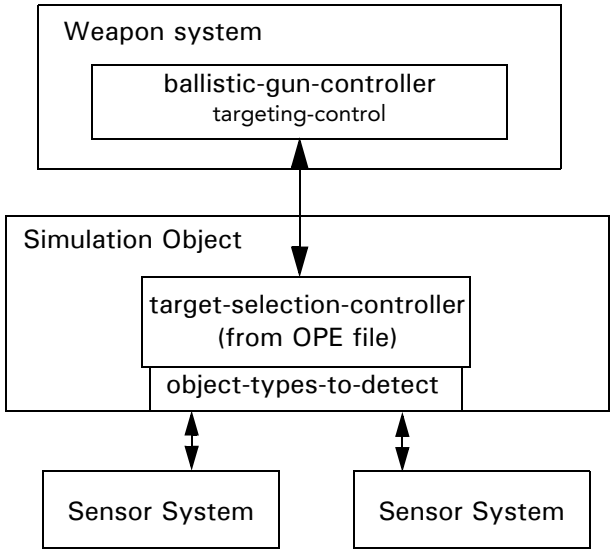


Figure D-1. Direct fire sensor input

The simulation object gathers the detected objects from all the sensors and stores it in the entity's state repository. (A state repository is a read-write repository of information that is available to all entity components.) The target selection controller selects a target for the weapon.

D.1.2. Ammo Select Tables

The ballistic gun controller descriptor refers to an ammo select table file, which contains the ammo select table for that controller. Each entry in this file specifies a target type (using the best match method) and a prioritized list of ammunition that should be used for that type of target (highest priority listed first). (The best match method is described in "The Best Match Method," on page 74-7.) The name of the entry is not used, and so may be something descriptive for readability. Each entry has the following fields:

- target-type – a 7-digit entity type enumeration, which can use -1s to denote wildcards.
- ammo – list of ammunition entries. The names of the ammunition entries are assumed to be the resource names. If the resource name is found in the Object Resource Manager, the amount of the ammunition is decremented appropriately, and is not used if the resource amount is 0. In this case, the next ammunition entry (if any) in the ammunition list is used. Each ammunition entry has the following fields:
  - munition-type – 7 digit object type enumeration. This is the object type that is used in the fire and detonation interactions created, and should NOT contain -1s.
  - tracer-type – 7 digit object type enumeration. This parameter is optional. Use it if you want a weapon to support tracer rounds.
  - warhead – VR-Link warhead enumeration value to use in fire and detonation interactions for this munition.

The following is an example of an Ammo Select Table (*M16.asl*):

```
(ammo-select-table
   (lifeform ;; matches any lifeform
      (target-type  3 -1 -1 -1 -1 -1 -1)
      (ammo
         (M16A2-556mm ;; these names need to match resource names
            (munition-type  2 8 225 2 1 1 0)
            (tracer-type    2 8 225 2 1 2 0)
            (warhead  0)
         )
      )
   )
   (civilian-vehicle  ; "technical"
      (target-type  1 1 -1  27 -1 -1 -1)
      (ammo
         (M16A2-556mm ;; these names need to match resource names
            (munition-type  2 8 225 2 1 1 0)
            (tracer-type    2 8 225 2 1 2 0)
            (warhead  0)
         )
      )
   )
)
```

D.1.3. Hit Probability Tables

A weapon system definition file specifies a hit probability file. This file contains a hit probability table that is used by the system's actuator to determine whether or not it hits a target that it fires upon. A hit probability table has one or more entity-range entries. The name of the **entity-range** entry is not used, and can be something descriptive. An **entity-range** entry has the form:

- **entity-type** – 7 digit entity type enumeration, which may use -1s as wildcards.
- **range-determinant** – a range determinant can be either a list of probability-range entries with the name "range-list", or a probability-coefficient list with the name "coefficients".
- **range-list** – a list of probability-range entries. A probability-range entry has a name, "range", a range value (in meters) and a list of probability entries. The entry applies to targets in the range from the last entry to the current one (or from 0.0 in the case of the first entry). Each probability entry has a name that the table user uses to find the probability value it is interested in and the value itself (a real number between 0.0 and 1.0.) The ballistic gun looks for the probability entry with the name "probability".



- For ranges that are greater than the last specified range, the probability is zero.
- The range for missiles is the range from the missile to the target, not from the entity that fired the missile.

- **coefficients** – a coefficient probability list is a list of coefficient probability entries. Each entry has a probability name and the coefficients of a standard polynomial that can be used with a given range to determine the specified probability. The order of the polynomial is determined by the number of coefficients provided. For example, an entry of:

  ```
  (probability -5.00000E-008 -300000E-012 1.0000)
  ```

  indicates that the formula to use for a given range *r* would be:

  $(-5.00000E-008 * r^2) + (-300000E-012 * r) + 1.0$

  Values calculated will be constrained to be between 0.0 and 1.0. Zero values are factored out, for example:

  ```
  (probability -5.00000E-008 0.0000 1.0000)
  ```

  indicates that the formula to use for a given range *r* would be:

  $(- 5.00000E-008 * r^2) + (0.0 * r) + 1.0$

The following is an example of the *M1A2MainGun.hit* hit probability table:

```
(hit-probability-table
  (entity-range
    (entity-type  1 1 -1 -1 -1 -1 -1)
    (range-determinant
      (range-list
        (range 1000.000000
          (probability 0.90000)
        )
        (range 2000.000000
          (probability 0.8000)
        )
        (range 3000.000000
          (probability 0.75000)
        )
        (range 4000.000000
          (probability 0.70000)
        )
      )
    )
  )
)
```

D.1.4. Damage Probability Tables

The damage actuator uses damage probability tables specified by its component descriptor to determine direct and indirect fire damage. The damage probability table has a list of surface entries, which provide a sub-table for each surface upon which a round might detonate. The name of the surface entry is the surface name used in the damage model.

For the damage-by-power damage system, VR-Forces has damage probability tables for each power type. For more information, please see "Configuring the Damage File for Damage-By-Power," on page D-19.

For the damage-by-ammo-type damage system, VR-Forces has damage probability tables for direct fire and collateral damage. A munition type does not need to have both types of probability table if one of the types is not appropriate. For example, an M16 does not have a collateral damage file. Conversely, an artillery shell does not have a direct fire probability table. For more information, please see "Configuring Damage-by-Ammo-Type," on page D-22.

D.2. Configuring Indirect Fire Weapons

Indirect fire weapon systems use the indirect-fire-weapon-controller. Indirect fire weapons fire in response to an indirect fire task, such as Fire-for-Effect or Throw Grenade. They do not fire on their own, like many direct fire weapons do.

When a simulation object receives an indirect fire task, its indirect-fire-controller queues the task and checks for availability of an indirect fire weapon. It continues to check until a weapon becomes available. Then it sends the fire task to the weapon through the indirect weapon interface. If a task requires multiple weapons, the indirect-fire-controller continues this process until the task is complete. (The indirect-fire-controller and indirect weapon interface are analogous to the target-select-controller and weapon interface described for direct fire weapons.)

When an indirect-fire-weapon-controller gets a task, it reads the fire mission from the indirect weapon interface. The indirect fire weapon controller tries to align the turret and load the munition. When the weapon is ready to fire, it passes the information to its indirect-fire-actuator.

The munition is specified as part of the task. The indirect-fire-actuator creates the desired munition and sends out the fire interaction.

Figure D-2 illustrates this process.

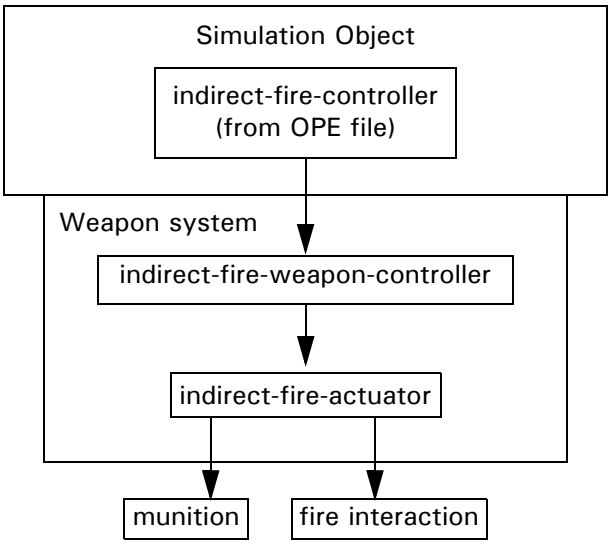


Figure D-2. Indirect fire



Do not confuse the indirect-fire-controller and the indirect-fire-weapon-controller. The indirect-fire-controller is on the simulation object that has the weapon system (it is in the platform OPE file). The indirect-fire-weapon-controller is in the indirect fire weapon system. The controllers are illustrated in Figure D-2.

Examples of indirect fire weapons include the M284-155mm cannon (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/M284-155mm-cannon.sysdef*), the M252-81mm mortar (*M252-81mm-mortar.sysdef*), and MK45 naval gun (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/MK45NavalGun.sysdef*). Also, although hand grenades would not normally be thought of as indirect fire weapons, the hand grenade tasks call the fire for effect tasks. Therefore, they use the same controllers and actuators as indirect fire weapons.

The munitions used by the indirect fire weapon systems are configured with explosive power in *detonationParams.mtl*. Therefore simulation objects that are susceptible to explosive power can be damaged by them. Their damage system definition files will point to the appropriate damage table files to assess damage from these munitions.

As with other weapon systems, the system definition files for indirect fire weapons have parameters that control when they can be used and how effective they are. For example, they have minimum and maximum ranges, as shown in this snippet from the cannon-controller in *M284-155mm-cannon.sysdef*:

```
(load-ammo-time  15.000000)
(unload-ammo-time  5.000000)
(default-ammo-list
  (m107  "M107-155mm")
  (m485  "M485-Illumination")
)
(num-rounds-per-mission  5)
(min-range  2000.000000)
(max-range  30000.000000)
(min-altitude 200.0)
(max-altitude 12500.0)
(preferred-altitude 1000.000000)
(muzzle-speed-list )
(range-name  "M284-15mm Cannon")
```

The following snippet is from the M284-cannon actuator in the same system definition file:

```
(projectile-start-speed  0.000000)
(muzzle-offset  6.090000 0.000000 0.000000)
(probable-error-in-range  15.000000)
(probable-error-in-displacement  4.000000)
(probable-error-in-burst-height  5.000000)
(simulate-munition True)
```

D.3. Configuring Fixed Weapons

Fixed munitions, such as mines and IEDs can be triggered in the following ways:

- Immediately, by a detonate message.
- By proximity of a valid target. The weapon needs a sensor system or a proximity controller that can detect targets.
- Timed detonation, from when the system is armed.

Fixed munitions have a detonation controller. The detonation controller processes set detonation messages to determine how it is to be triggered and whether or not it is armed. It tells the warhead-detonation-actuator when to arm the weapon. For immediate and timed detonation, it tells the warhead actuator to detonate the device.

For proximity detonations, the detonation controller takes input from the external sensor or the proximity sensor. The warhead-actuator checks for armed and trigger messages from the detonation controller. It also checks for collisions with the terrain and simulation objects. If a collision occurs or it is triggered by the detonation controller, it sends a detonation interaction over the network.

VR-Forces has two types of fixed weapons, naval mines and IEDs.

IEDs are simulation objects that have an explosive device system (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/explosive-device.sysdef*). These objects need a sensor to detect targets. The Roadside IED object has a visual sensor. The detonate controller lists the object types to detect:

```
(fuse-type  "Proximity Fuse")
(detonation-proximity-distance  2.000000)
(armed True)
(object-types-to-detect
  (object-type  1 (1 -1 -1 -1 -1 -1 -1))
  (object-type  1 (3 -1 -1 -1 -1 -1 -1))
  (object-type  1 (5 -1 -1 -1 -1 -1 -1))
)
```

Naval mines use the *./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/naval-mine-explosive-device.sysdef* system definition file. It has a proximity sensor built into the system definition, so it does not need an external sensor to detect targets. The detonate controller does not list the object types to detect:

```
(fuse-type  "Proximity Fuse")
(detonation-proximity-distance  25.000000)
(armed True)
(object-types-to-detect )
```

However, the proximity sensor component does list object types to detect:

```
(default-proximity 150)
(detect-only-hostile-forces True)
(object-types-to-detect
  (object-type  1 (1 3 -1 -1 -1 -1 -1))
  (object-type  1 (1 4 -1 -1 -1 -1 -1))
)
```

You can turn an entity into an IED by adding the explosive device system in the Simulation Object Editor. You can configure cultural features as IEDs by adding the Other Explosive Device system.

D.4. Configuring Bomb Bay Weapon Systems

The bomb-bay weapon system releases bombs in response to Release Bomb tasks. Bombs can drop directly onto targets or be laser guided. Once a bomb is released, it uses the *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement/smart-bomb-dynamics.sysdef* movement system to control movement. It has a warhead actuator that is similar to the warhead actuator in missiles. Its ability to navigate toward and hit its target is controlled by parameters that you can set in the Simulation Object Editor. For details, please see "Release Bomb Tasks," on page 27-20.

D.5. Calculating Damage from Munitions

VR-Forces has the following ways to calculate damage:

- Damage-by-power. Each munition is configured for its power in the explosive, kinetic, fragmentation, and armor-piercing domains as appropriate. (You can add new domains without writing code.) Damage systems (*.sysdef* files) reference the damage files (*.dmg*) for the type of munition power that can affect simulation objects that use the system. Damage files configure the amount of damage that occurs when different levels of power are applied. The damage-by-power model is used for all entity-level simulation objects in VR-Forces 4.5 and later and is the preferred method for calculating damage.
- Damage-by-ammo-type. Each munition that can affect a simulation object is configured individually in the damage system used by that object and refers to a damage file that is specific to that type of munition. This model was used in VR-Forces 4.4.2 and earlier. It is supported in VR-Forces 4.5 and later.

The damage-by-power method of configuration makes it much easier to add new munitions than damage-by-ammo-type. You add the munition to one file and the generic power calculations in the damage files translate the power of the munition to damage. However, if you have specific requirements for the effects of a munition on different simulation objects, you can use the damage-by-ammo-type method together with damage-by-power. If VR-Forces finds an `ammo-entry` block in the damage system file, it uses that entry and its related damage file for that ammo type instead of the damage-by-power damage file that would otherwise be used.

For more information, please see "Configuring Damage-by-Power," on page D-14 and "Configuring Damage-by-Ammo-Type," on page D-22.

D.6. Configuring Damage-by-Power



This section is largely the same as the description in "Configuring Damage-by-Power," on page 77-3. However, there are some differences between how damage-by-power was configured in VR-Forces 4.6.1 and earlier and how it is configured in VR-Forces 4.7 and later, so this information is presented in its entirety.

In the damage-by-power method of calculating damage, each munition has a `detonation-power-entry` block in *./data/simulationModelSets/base/detonationParams.mtl*. The following is the entry for the 7.62 mm bullet:

```
(detonation-power-entry
  (munition
    ;; 7.62 mm
    (munition-type  2 8 222 2 2 -1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (kinetic
      (direct 2.24) ;; Kilojoules
    )
  )
)
```

The `munition` block identifies the munition type using the DIS/RPR FOM object enumeration. (The default configuration does not use the **warhead** or **guidance-mode** parameters.)

The `power-list` block is a list of domains and the configuration for a determinant that produces a power value based on an input range. In this example, the 7.62 mm shell has power in the kinetic domain.

The default configurations support the following domains:

- Explosive. Munitions that affect simulation objects within some range of the detonation. Explosive munitions represent power in kiloPascals (kPa) of incident pressure.
- Kinetic. Munitions that directly strike a simulation object. Kinetic munitions represent power in kilojoules (kJ) of energy.
- Armor-piercing. Munitions that strike an armored object and penetrate it. Armor piercing munitions represent power in millimeters of penetration.
- Fragmentation. Explosives that cause damage by fragmenting, such as grenades.



You can add other domains without writing code. For details, please see "Adding a New Power Type," on page D-20.

The power values used in the default configuration are based on publicly available information about the power of the various types of munitions and the values used try to be consistent in their relative power.

D.6.1. Configuring the Power of a Detonation

To determine the power of a detonation at a specific location a power determinant must be defined. The determinant maps a range value to a power value. Range can have different possible meanings, depending on how the **range-from** parameter is defined for the determinant. If **range-from** is undefined, `impact-point` is assumed unless the determinant does not support this setting.

- If **range-from** is `impact-point`, the determinant determines power based on the range from the impact point of the detonation to some other location in the vicinity. This is used for the explosive and fragmentation domains, but could be used for any power that has some area of effect (for example, perhaps shrapnel or incendiary damage). This is equivalent to collateral damage as defined in the damage-by-ammo-type damage system.

  Example (from *detonationParams.mtl*):

```
(detonation-power-entry
  (munition
     ;; 85 mm HEAT
    (munition-type  2 2 222 2 8 1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (range-list
        (range 3.5
          (power 70.0)
        )
        (range 4.6
          (power 40.0)
        )
        (range 5.7
          (power 27.5)
        )
        ...
        (range 35.0
          (power 2.4)
        )
      )
    )
    (kinetic
      (range-from "shooter-to-impact")
      (range-coefficients -2.6600E-004   0.0000   5319.0)
    )
    (armor-piercing
      (range-from "shooter-to-impact")
      (range-coefficients -9.0000E-006   0.0000   180.0)
    )
  )
)
```

- If **range-from** is `shooter-to-impact` (as in the previous example), the determinant determines power based on the range from the entity that fired the round to the point of impact. Any points outside of the point of impact are not affected by the detonation. This is used for the kinetic and armor-piercing domains, but could be used for any power that is directed at a particular target and has no area of effect. This is equivalent to direct damage as defined in the damage-by-ammo-type damage system.
- If **range-from** is `unused`, the determinant determines power without using the input range. This is used by the direct determinant, which ignores the **range-from** setting completely. (For more information, please see "Determinant Types," on page D-16.)
- If **range-from** is `shooter-and-impact-point`, the determinant determines power based on both the range from the shooter and from the impact point. This option is not implemented by VR-Forces.

Determinant Types

VR-Forces supports the following types of determinant.

- The **range-coefficients** determinant calculates the power based on a series of coefficients. Configuring coefficients works exactly the same way as the hit probability coefficients, except that the output value can be any number 0 or greater, not just a probability for 0.0 to 1.0. (For details about hit probability, please see "Hit Probability Tables," on page D-7.) For example:

```
(kinetic
  (range-from "shooter-to-impact")
  (range-coefficients -1.3000E-003   0.0000   26005.0)
)
```

- The **range-list** determinant determines the power based on a step function. This is configured exactly the same way as the hit probability **range-list**, except that the output value can be any number 0 or greater, not just a probability for 0.0 to 1.0. For example:

```
(explosive
  (range-from "impact-point")
  (range-list
    (range 4.5
      (power 70.0)
    )
    (range 6.0
      (power 40.0)
    )
    ...
    (range 47.0
      (power 2.4)
    )
  )
)
```

- The direct determinant always produces the same power at the point of impact, regardless of the range from the shooter. It produces no power at any range from the point of impact. As a result, the range-from setting is ignored, because the range input is unused. For example:

```
(kinetic
  (direct 272.873) ;; Kilojoules
)
```

- The equivalent-tnt-mass determinant allows you to define an amount of TNT, in kilograms, that produces an equivalent explosion. VR-Forces calculates power at a specified range in kPa. This determinant is used only for the explosive damage category. The following example shows its use for M107 HE munition:

```
(detonation-power-entry
  (munition
    ;; M107 HE (155 mm)
    (munition-type  2 9 225 2 14 2 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (equivalent-tnt-mass 5.74)
    )
  )
)
```

D.6.2. The Damage File for Damage-By-Power

Entity damage is configured through the system definition files and the damage files they reference. A system definition file configures power in the `damage-by-munition-power` block.

The system definition files provided with VR-Forces have a `damage-model` block. However none of the files provided use this block. It is available for customers who want to configure munitions using the damage-by-ammo-type method. It also enables backwards compatibility with old SMSs. For information about using the damage-model block, please see "Configuring Damage-by-Ammo-Type," on page D-22.

Configuring Damage-by-Power in a System Definition File

A system definition file can define any number of `damage-by-power-entry` items, each for a different power-type. This is the damage domain of the power. To add a new type of power, add new settings in *detonationParams.mtl* and in the system definition files for simulation objects that need to be damaged by this effect.

Each damage-by-power-entry specifies a damage file that defines how a power value maps to a damage probability. (Damage files are in *./data/simulationModelSets/EntityLevel/vrfSim/damage*.) The comments in the damage file indicate the unit of measurement for power.

The following example is from *./data/simulationModelSets/EntityLevel/vrfSim/systems/damage/ground-heavy-armor.sysdef*:

```
(damage-by-munition-power
  (damage-by-power-entry
    (power-type "explosive")
    (damage-file
      (filename  "$(damage-dir)\heavy-armor-explosive.dmg")
    )
  )
  (damage-by-power-entry
    (power-type "kinetic")
    (damage-file
      (filename  "$(damage-dir)\heavy-armor-kinetic.dmg")
    )
  )
  (damage-by-power-entry
    (power-type "armor-piercing")
    (damage-file
      (filename  "$(damage-dir)\heavy-armor-armor-piercing.dmg")
    )
  )
)
(damage-model )
```

It shows that simulation objects that use the *ground-heavy-armor.sysdef* system definition file can be damaged by explosive, kinetic, and armor piercing power.

Configuring the Damage File for Damage-By-Power

Damage files configure damage based on the side of the bounding volume being affected. The damage actuator delivered with VR-Forces uses surfaces of front, rear, left-side, right-side, bottom, and top (other components could model the entity's geometry differently, and therefore use different surface strings). A surface entry has a list of angle-of-incidence entries. An **angle-of-incidence** entry has an angle (in radians).

The `determinant` block specifies the damage. The determinant can be a `step-function` or a list of coefficients. Each determinant can specify probabilities for catastrophic-kill, mobility-kill, firepower-kill or some combination of them.

The *ground-heavy-armor.sysdef* file has a parameter called **round-down** that indicates what probabilities should be used when the input power value is between two steps. For power we want to round down to the next lower power (and lesser probability). Therefore **round-down** is set to 1 (true) for all damage files. If this parameter is not specified, the step function rounds up.

The following code, which uses the `step-function` determinant, is from *./data/simulationModelSets/EntityLevel/vrfSim/damage/heavy-armor-explosive.dmg*:

```
(damage-table
  (front
    (angle-of-incidence
      (angle  1.570800)
      (determinant
        (step-function (round-down 1)
          (power 7.5
            (catastrophic-kill  0.1)
          )
          (power 12.5
            (catastrophic-kill  0.5)
          )
          (power 20.0
            (catastrophic-kill  0.6)
          )
          (power 45.0
            (catastrophic-kill  0.75)
          )
          (power 70.0
            (catastrophic-kill  0.99)
          )
        )
      )
    )
  )
  ...
)
```

The following code, which uses the `coefficients` determinant, is from *./data/simulationModelSets/EntityLevel/vrfSim/damage/missile-explosive.dmg*:

```
(damage-table
  ...
  (left-side
    (angle-of-incidence
      (angle  1.5708)
      (determinant
        (coefficients
          (catastrophic-kill 0.02 -0.4)
          (firepower-kill 0.022 -0.44)
          (mobility-kill 0.025 -0.5)
        )
      )
    )
  )
  ...
)
```

D.6.3. Adding a New Power Type

To add a new type of power, you add it to the relevant munitions in *detonationParams.mtl* and then add it to the appropriate damage system. For example, suppose you want to model the effects of the heat generated by a munition. You could add thermal power. For the sake of this example, suppose that an 85mm HEAT round has thermal power. To add thermal power, you would have to do the following:

1. Add the new power to the `detonation-power-entry`, as follows:

```
(detonation-power-entry
  (munition
    ;; 85 mm HEAT
    (munition-type  2 2 222 2 8 1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (range-list
        (range 3.5
          (power 70.0)
        )
        ...
      )
    )
    (kinetic
      (range-from "shooter-to-impact")
      (range-coefficients -2.6600E-004   0.0000   5319.0)
    )
    (armor-piercing
      (range-from "shooter-to-impact")
      (range-coefficients -9.0000E-006   0.0000   180.0)
    )
```

```
      (thermal
        (range-from "impact-point")
        (range-list
          (range 3.5
            (power 70.0)
          )
          ...
        )
      )
    )
)
```

You would have to determine the unit of heat that you wanted to use to measure its effect and come up with realistic power levels at different distances.

2. In the damage systems for the simulation objects that can be affected by thermal power, add a `damage-by-power` entry. Suppose that you want lifeforms to be damaged by thermal power. You would update the *lifeform-default.sysdef* file as follows:

```
(damage-by-munition-power
  (damage-by-power-entry
    (power-type "explosive")
    (damage-file
      (filename  "$(damage-dir)\lifeform-explosive.dmg")
    )
  )
  (damage-by-power-entry
    (power-type "kinetic")
    (damage-file
      (filename  "$(damage-dir)\lifeform-kinetic.dmg")
    )
  )
  (damage-by-power-entry
    (power-type "thermal")
    (damage-file
      (filename  "$(damage-dir)\lifeform-thermal.dmg")
    )
  )
)
```

3. Create a damage file called *lifeform-thermal.dmg*. Configure it similarly to other damage files.

D.7. Configuring Damage-by-Ammo-Type

Damage-by-ammo-type is the damage configuration approach used by VR-Forces in release 4.4.2 and earlier. It is no longer the default approach. However, it is still supported both for backwards compatibility and for those cases where you want to configure a specific munition to affect a specific entity type.

To configure damage-by-ammo-type, you add an `ammo-entry` to the `damage-model` block of the damage system's system definition file (*.sysdef*) for each type of ammunition that you want a simulation object using that system definition to be vulnerable to. Then, you create a damage table (*.dmg*) for the type of damage that you can take.

The following code is from *vrforces4.4.2/data/simulationModelSets/EntityLevel/vrfSim/systems/damage/lifeform-default.sysdef*.

```
(damage-model
  (ammo-entry
    (ammo
      ;;Ballistic anti-air munitions-- 20-30mm HE cannon
      (munition-type  2 1 -1 2 -1 -1 -1)
      (warhead  -1)
      (guidance-mode  0)
    )
    (direct-damage-file
      (filename  "$(damage-dir)\light-armor-vs-explosive-round.dmg
    )
    (collateral-damage-file
      (filename  "$(damage-dir)\collateral-gnd-PG7.dmg")
    )
  )
  (ammo-entry
    (ammo
      (munition-type  2 2 222 1 8 -1 -1)
      (warhead  -1)
      (guidance-mode  0)
    )
    (direct-damage-file
      (filename  "")
    )
    (collateral-damage-file
      (filename  "$(damage-dir)\gnd-TOW.dmg")
    )
  )
  ...
)
```

These entries specify two munition types that lifeforms are vulnerable to (anti-air munitions and TOW missiles). In the case of ballistic anti-air munitions, they can take collateral damage and direct damage. In the case of TOW missiles they can take collateral damage.

D.7.1. Configuring Damage for Damage-by-Ammo-Type

As in damage-by-power, damage files for damage-by-ammo-type configure damage based on the side of the bounding volume being affected. The damage actuator delivered with VR-Forces uses surfaces of front, rear, left-side, right-side, bottom, and top (other components could model the entity's geometry differently, and therefore use different surface strings). A surface entry has a list of angle-of-incidence entries. An angle-of-incidence entry has an angle (in radians).

A surface entry has a range-determinant. The entry is intended for use with angles of incidence that go from the previous entry's specified angle to the current one (or in the case of the first entry, from 0.) The range determinant is exactly the same as the range determinant described in "Hit Probability Tables," on page D-7.

The damage actuator provided by MAK looks for a probability entry with the name catastrophic-kill, mobility-kill, or firepower-kill.

The following code is a portion of the damage table from *vrforces4.4.2/data/simulationModelSets/EntityLevel/vrfSim/damage/light-armor-vs-explosive-round.dmg*.

```
(damage-table
  (front
    (angle-of-incidence
      (angle  0.5236) ;; 30 degrees
      (range-determinant
        (coefficients
          (catastrophic-kill -5.0000E-008   0.0000   1.0000)
        )
      )
    )
    (angle-of-incidence
      (angle  1.0472) ;; 60 degrees
      (range-determinant
        (coefficients
          (catastrophic-kill -5.0000E-008   0.0000   1.0000)
        )
      )

  ...
  )
  ...
)
```

The following code is a portion of the damage table from *vrforces4.4.2/data/simulationModelSets/EntityLevel/vrfSim/damage/collateral-gnd-PG7.dmg*.

```
(damage-table
  (front
    (angle-of-incidence
      (angle  0.5236) ;; 30 degrees
      (range-determinant
        (range-list
          (range 1.000000
            (catastrophic-kill  0.900000)
          )
          (range 2.000000
            (catastrophic-kill  0.750000)
          )
        (range 5.000000
          (catastrophic-kill  0.600000)
        )
        (range 10.000000
          (catastrophic-kill  0.450000)
        )
        (range 15.000000
          (catastrophic-kill  0.050000)
        )
      )
    )
    ...
  )
  ...
)
```

D.8. Direct Fire Configuration Examples

This section has two examples of how to configure a direct fire weapon using the VR-Forces 4.6 and earlier approaches. The first uses the damage-by-power method. The second uses the damage-by-ammo-type method.

D.8.1. Direct Fire Weapon Damage-By-Power Example

This section describes the files that work together to define an M240 machine gun and allow it to fire on and damage entities using VR-Forces 4.6 and earlier configuration files. Figure D-3 illustrates the relationships between the files. (The lines of code are not in context. They are the minimum needed to show the connections between the files.)

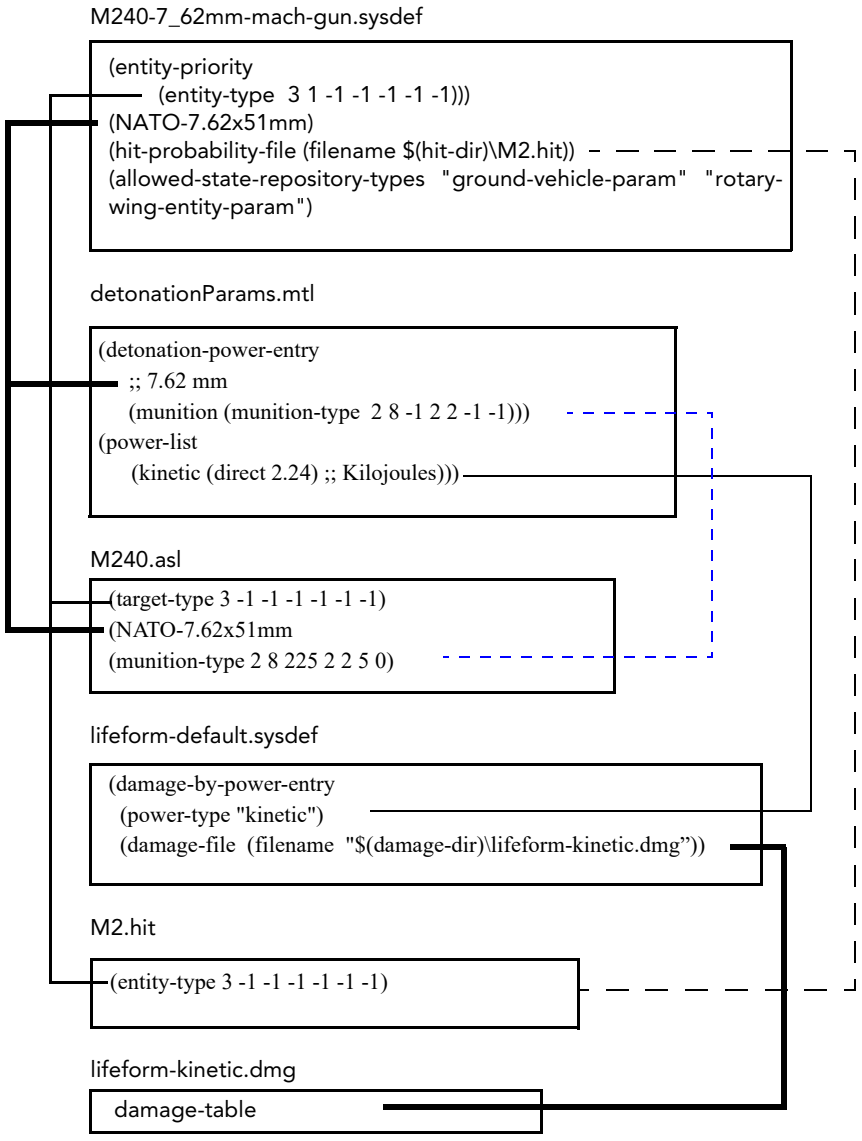


Figure D-3. Anatomy of a weapon system using damage-by-power

The Weapon System System Definition File

The M240 machine gun is configured in *./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/M240-7_62mm-mach-gun.sysdef*. Most of the file specifies the various components of the system that make it work. We are interested in the lines that help it connect to other files.

The meta-data section of a system definition file provides information the Simulation Object Editor needs to make the system available in the systems section.

In the following code snippet, the **allowed-state-repositories-types** parameter specifies that ground vehicles and rotary-wing entities can use this system.

```
(meta-data
  (system-name  "M240 Machine Gun")
  (system-description "Vehicle mounted M240 7.62mm Machine Gun. Typical
  secondary gun on tanks such as the M1A2. Targets lifeforms and unarmored
  vehicles.")
  (allowed-state-repository-types "ground-vehicle-param" "rotary-wing-
  entity-param")
  (system-categories "weapon")
```

The resources section specifies the type of ammunition this weapon uses.

```
(resources
  (NATO-7.62x51mm
  ...
)
```

The targeting control section specifies the entity types that this system can fire at —humans and several types of ground vehicles. Note that wildcards are used for entity types.

```
(targeting-control
  (target-priorities
    (entity-priority
      (entity-type  3 1 -1 1 1 -1 -1)
      (priority  2)
    )
    (entity-priority
      (entity-type  3 1 -1 -1 -1 -1 -1)
      (priority  2)
    )
    (entity-priority
      (entity-type  1 1 -1 7 -1 -1 -1); large trucks
      (priority  3)
    )
    ...
  )
)
```

Finally, a ballistic direct fire weapon system system definition file specifies a hit file for the munition it uses. The hit file specifies the probability that targets will get hit by the munition.

```
(main-gun-joy
  (component-descriptor-type "ballistic-gun-descriptor")
  ...
  (velocity-affects-hit-chance False)
  (hit-probability-file
    (filename  "$(hit-dir)\M2.hit")
  )
  (range-name  $display-name (default "M240 MG"))
  (start-with-tracers true)
)
```

The Ammo Select File

When an entity with an M240 machine gun detects a target, it must choose the correct ammunition for that target. This is specified in the ammo select file. The ammo select file for the M240 is *./data/simulationModelSets/EntityLevel/vrfSim/ammoselect\M240.asl*.

The file has entries like the following:

```
(lifeform ;; matches any lifeform
  (target-type  3 -1 -1 -1 -1 -1 -1)
  (ammo
    (NATO-7.62x51mm ;; these names need to match resource names
      (munition-type  2 8 225 2 2 5 0)
      (tracer-type    2 8 225 2 2 4 0) ; Ball/tracer mix, 4:1
      (warhead  0)
    )
  )
)
```

There is an entry for each of the target types in *M240-7_62mm-mach-gun.sysdef*. If you specify a target type in a weapon system definition file and do not have a corresponding entry in an ammo select file, the weapon will not know what kind of ammunition to use for that target type and will not fire on it.

For each target, it specifies the entity type for the munition and tracer bullet.



Weapon systems that fire autonomously, such as the main guns on tanks, machine guns, and missiles, need an ammo select file. Some weapons, such as artillery and naval guns, do not need an ammo select file because they only fire when tasked by users or in a plan, and the choice of munition is one of the task parameters.

The *detonationParams.mtl* File

As noted in previous sections, the M240 gun uses a 7.62 mm munition. The *detonationParams.mtl* file specifies that a 7.62 mm munition has kinetic power.

The Damage System System Definition File

Each simulation object has a damage system. The M240 can target lifeforms. Most lifeforms use the Default Armor damage system. It is configured in *./data/simulationModelSets/EntityLevel/vrfSim/systems/damage/lifeform-default.sysdef*. This file uses the damage-by-power configuration. It specifies that a lifeform can be damaged by explosive power and kinetic power.

```
(damage-by-munition-power
  (damage-by-power-entry
    (power-type "explosive")
    (damage-file
      (filename  "$(damage-dir)\lifeform-explosive.dmg")
    )
  )
  (damage-by-power-entry
    (power-type "kinetic")
    (damage-file
      (filename  "$(damage-dir)\lifeform-kinetic.dmg")
    )
  )
)
```

Since the M240 machine gun fires a munition that has kinetic power, if a lifeform is attacked by this munition, VR-Forces looks at the *lifeform-kinetic.dmg* damage file to calculate damage.

The Hit File

A hit file specifies the probability that a simulation object will be hit by a type of munition at various distances. The hit file specified by the M240 system definition is *M2.hit*. It has an entry for each simulation object type that can be struck by this munition type. The following code snippet is the entry for human entities.

```
(entity-range
  (entity-type  3 -1 -1 -1 -1 -1 -1)
  (range-determinant
    (range-list
      (range 10.000000
        (probability 1.00000)
      )
      (range 200.000000
        (probability 0.80000)
      )
      (range 500.000000
        (probability 0.4000)
      )
      (range 1000.0
        (probability 0.2)
      )
      (range 3600.000000
        (probability 0.100)
      )
    )
  )
)
```

For more information about hit files, please see "Hit Probability Tables," on page D-7.

The Damage File

Once a simulation object has been fired on, with an appropriate munition, for which it can take damage, and by which it can be hit, VR-Forces calculates the damage using the damage file specified in the damage system definition file. For the M240 machine gun, targeting a human entity, the damage file is *lifeform-kinetic.dmg*.

Based on the side of the entity that is hit, the angle it is hit at, and the range of the munition, the damage file specifies the type of damage caused.

For more information about damage files, please see "Damage Probability Tables," on page D-8.

D.8.2. Direct Fire Weapon Damage-by-Ammo-Type Example

This section describes the files that work together to define an M240 machine gun and allow it to fire on and damage entities using the damage-by-ammo-type method and VR-Forces 4.6 and earlier configuration files. Figure D-4 illustrates the relationships between the files. (The lines of code are not in context. They are the minimum needed to show the connections between the files.)

In the damage-by-ammo-type approach, the hit file, ammo select file, and weapon system file are the same as in the damage-by-power approach and will not be discussed in this section. The damage system system definition file is different, as is the damage file. The *detonationParams.mtl* file is not used.

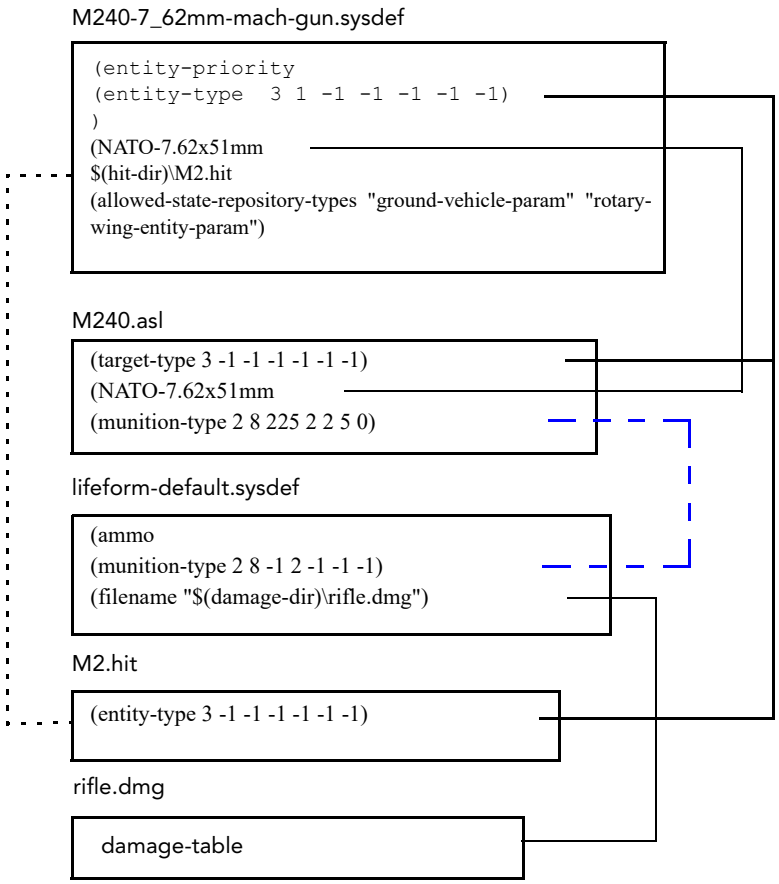


Figure D-4. Anatomy of a weapon system using damage-by-ammo-type

The Damage System System Definition File

Each simulation object has a damage system. The M240 can target lifeforms. Most lifeforms use the Default Armor damage system. For the damage-by-ammo-type approach, in VR-Forces 4.4.2 or earlier, it is configured in *./data/simulationModelSets/EntityLevel/vrfSim/systems/damage/lifeform-default.sysdef*. The damage system has a a `damage-model` block with an `ammo-entry` block for each type of ammunition that simulation objects that use the damage system are vulnerable to. The ammo entry can specify a damage file (*.dmg*) for direct hits and for collateral damage. The *lifeform-default.sysdef* includes the following ammo entry:

```
(damage-model
  ...
  (ammo-entry
    (ammo
      munition-type  2 8 -1 2 -1 -1 -1)
      (warhead  -1)
      (guidance-mode  0)
    )
    (direct-damage-file
      (filename  "$(damage-dir)\rifle.dmg")
    )
    (collateral-damage-file
      (filename  "")
    )
  )
)
```

The **munition-type** parameter uses wildcard values that match the munition type sent by the M240 machine gun (2 8 225 2 2 5 0). This means that a human entity that uses the Default Armor damage file can be damaged by a bullet fired by an M240 machine gun.

The Damage File

Once a simulation object has been fired on, with an appropriate munition, for which it can take damage, and by which it can be hit, VR-Forces calculates the damage using the damage file specified in the damage system definition file. For the M240 machine gun, targeting a human entity, the damage file is *rifle.dmg*.

Based on the side of the entity that is hit, the angle it is hit at, and the range of the munition, the damage file specifies the type of damage caused.

For more information about damage files, please see "Damage Probability Tables," on page D-8.



E. Systems and System Usage

The tables in this appendix provide information about the systems provided with VR-Forces and list the simulation objects that use them.

VR-Forces Systems E-2

E.1. VR-Forces Systems

To see a list of systems for each SMS with a description, a list of simulation objects that use it, and the types of ammunition used (if applicable), load the SMS in the Simulation Object Editor and view the General tab for each system.

Some scripted tasks are only available to simulation objects that have a particular system. Table E-1 lists systems and the scripted tasks configured in them for *EntityLevel.sms*. Table E-2 lists systems and the scripted tasks configured in them for *AggregateLevel.sms*.

Table E-1: Systems and configured scripted tasks, EntityLevel.sms

System	Scripted Tasks
Cargo Door	Open_Cargo_Door, Close_Cargo_Door
Sliding Door	Open_Sliding_Door, Close_Sliding_Door
Air Drop	Air_Drop
Anti-Submarine Missile (Vertically Launched)	Launch_ASW_Missile
Car Door	open_car_door, close_car_door
DDG Embedded Support Craft (LAMPS/RHIB)	Embedded_Sector_SAR
Fast Roping	Deploy_Ropes, FastRopeInsertion
Fighter Jet	Fixed_Wing_Takeoff, Fixed_Wing_Land, orbit_object
Heavy Plane	Fixed_Wing_Takeoff, Fixed_Wing_Land, orbit_object
High Maneuverability Fighter	Fixed_Wing_Takeoff, Fixed_Wing_Land, orbit_object
Folding Wing	Fold_Wings, Unfold_Wings
Tracks	move-to-location-path-plan, move-to-waypoint-path-plan
Tracks/Amphibious	move-to-location-path-plan, move-to-waypoint-path-plan
Wheels (off road)	move-to-location-path-plan, move-to-waypoint-path-plan
Wheels (road)	move-to-location-path-plan, move-to-waypoint-path-plan
Throw Grenades	ThrowGrenade, ThrowGrenadeTarget
Handheld UAV	orbit_object
Homing Torpedo Capability (Forward Launched)	Delayed_homing, Delayed_homing_ASW, Launch_Torpedo_Salvo
Human Disaggregated Movement	circle_around_location, find_cover_unit, find_firing_positions, move_between_cover_to_location_unit

Table E-1: Systems and configured scripted tasks, EntityLevel.sms

System	Scripted Tasks
Flee From Explosions	bhave-Flee_From_Explosion
IFF Code Changer	iff-code-changer
KN-02 Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
KN-08 Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Lifeform Suppression	Suppressed_Crouch, Suppressed_Prone
M203 Grenade Launcher	Fire_M203_at_Location, Fire_M203_at_Target
Naval Depth Charge Deployment	Drop_Naval_Depth_Charge, Drop_Naval_Depth_Charge_At_Location
Naval Mine Deployment	Drop_Naval_Mine, Drop_Naval_Mines_Along_Route
Naval Mine Sweep	Naval_Mine_Sweep
No Dong Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Nodong Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Pedestrian Crowd Movement	Crowd_Around, Crowd_Around_Location, Crowd_In_Front_Of, Crowd_Along_Line, Disperse_Crowd, Wander_In_Area, Protest, Protest_Along_Line
Pedestrian Traffic Generator	Create_Initial_Pedestrians, Create_Pedestrians, Delete_Created_Pedestrians
Periscope	Raise_Periscope, Lower_Periscope
Dipping SONAR Sensor	Sonar_Dip
Scud C Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Scud-ER Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Scud B Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Sonobuoy Deployer	Deploy_Sonobuoy, Deploy_Sonobuoys_Along_Route

Table E-1: Systems and configured scripted tasks, EntityLevel.sms

System	Scripted Tasks
SS-25 Sickle Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Tactical Radar Jammer	automatic_jamming
Tanker Refueling Boom	Deploy_Refueling_Boom, Stow_Refueling_Boom
Tactical Data Link Air Message Generator	Set_TDL_Messaging
Tactical Data Link Ground Message Generator	Set_TDL_Messaging
Tactical Data Link Subsurface Message Generator	Set_TDL_Messaging
Tactical Data Link Surface Message Generator	Set_TDL_Messaging
UGM-133 Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object

Table E-2: Systems and configured scripted tasks, AggregateLevel.sms

System	Scripted Tasks
Mortar, 120mm	Indirect_Fire, Limited_Munition_Attack
Aerial Refueling Boom	refueling_mission
Aerial Refueling Drogue	refueling_mission
122mm Howitzer	Indirect_Fire, Limited_Munition_Attack
152mm Howitzer Auto	Indirect_Fire, Limited_Munition_Attack
155mm Howitzer	Indirect_Fire, Limited_Munition_Attack
Bomb, 2000lb JDAM	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
Mk 45 5-inch Gun	Indirect_Fire, Limited_Munition_Attack

Table E-2: Systems and configured scripted tasks, AggregateLevel.sms

System	Scripted Tasks
9K330 Surface-to-Air Missile Launcher	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
9P140 MRL	Indirect_Fire, Limited_Munition_Attack
AIM-120 AMRAAM Missile	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, react_to_air_threats_air
Communications Jammer	EW_Attack, Activate_Jammer, Deactivate_Jammer
Bomb, GBU-24	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
Harpoon Antiship Missile	Limited_Munition_Attack, Attack_with_Antiship_Missile, React_To_Naval_Threats_Missile
M993 MLRS	Indirect_Fire, Limited_Munition_Attack
Torpedo, Mark 54	Limited_Munition_Attack, Attack_with_Torpedo, React_To_Naval_Threats_Torpedo
Maverick Missile	Limited_Munition_Attack, Attack_with_Antiship_Missile, Attack_with_Land-Attack_Missile
Patriot Anti-Air Missile	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
Radar Jammer	EW_Attack, Activate_Jammer, Deactivate_Jammer
Guided Bomb	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
RIM-66 SAM	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
Sidewinder Anti-Air Missile	Limited_Munition_Attack, Attack_with_AntiAir_Missile, react_to_air_threats_air
Stinger Anti-Air Missile	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
Tomahawk Cruise Missile	Limited_Munition_Attack, Attack_with_Land-Attack_Missile
Unguided Bomb	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
Torpedo, Type 40 (RPK-6 Vodopad SUBROC)	Limited_Munition_Attack, Attack_with_Torpedo, React_To_Naval_Threats_Torpedo
Air-Drop Supply	Provide_Supplies, Change_Supplying_Command
Aircraft Aggregated Movement	Maximum_Speed_Modifier, wait_for_escort, escort_air_unit, land_at_air_base
Biological Weapon	Biological_Attack, NBC_Attack

Table E-2: Systems and configured scripted tasks, AggregateLevel.sms

System	Scripted Tasks
Booby Trap Setter	Create_Booby_Trap, Improve_Booby_Trap
Boom Aerial Refueling Receiver	refuel_from_tanker
Bridge Builder	Construct_Bridge, Improve_Bridge
Chemical Weapon	Chemical_Attack, NBC_Attack
Ditch Digger	Create_Anti-Tank_Ditch, Improve_Ditch, Create_Dry_Gap
Drogue Aerial Refueling Receiver	refuel_from_tanker
Engineering Obstacle Destroyer	Destroy_Obstacle, Destroy_Fortification, Destroy_Ditch, Destroy_Bridge, Destroy_Explosive, Breach_Obstacles, Improve_Breach
Explosive Hazard and Simple Obstacle Destroyer	Destroy_Explosive_Hazard, Destroy_Obstacle, Breach_Obstacles, Improve_Breach
FASCAM	FASCAM
Fortification Builder	Construct_Barricade, Improve_Fortification, Construct_Fortified_Line, Construct_Fortified_Area, Construct_Strong_Point
Fuel Tank	drop_fuel_tanks
Ground Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path
Ground Combat Behaviors	Attack_Target
Infantry Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path
Mechanized Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path
Mine Dispenser	Create_Minefield
MOPP Gear	Change_MOPP_Level
Motorized Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path
Nuclear Weapon	Nuclear_Attack, NBC_Attack
Obstacle Builder	Construct_Abatis, Improve_Obstacle, Construct_Barbed_Wire, Construct_Berm
Ground Resupply	Provide_Supplies, Change_Supplying_Command
Riverine Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path

Table E-2: Systems and configured scripted tasks, AggregateLevel.sms

System	Scripted Tasks
Ship Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path
Simple Obstacle Destroyer	Destroy_Obstacle, Breach_Obstacles, Improve_Breach
Subsurface Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path
Tank Aggregated Movement	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path
Tactical Data Link Air Message Generator	Set_TDL_Messaging
Tactical Data Link Ground Message Generator	Set_TDL_Messaging
Tactical Data Link Subsurface Message Generator	Set_TDL_Messaging
Tactical Data Link Surface Message Generator	Set_TDL_Messaging



MAK Products Glossary

This glossary defines terms used in MAK product documentation.

absolute dead reckoning

A method of dead-reckoning in which an application uses the time stamps contained within state updates if the sender is using absolute time stamping. Contrast with relative dead reckoning.

aggregate

The combination of individual entities, aggregates, or both into a single object. For example, an organizational group such as a platoon.

API

Application Programming Interface.

articulated part

A part of an entity that is capable of movement relative to the entity, such as a turret or gun.

attached part

An object that is attached to another object, such as a rocket attached to a jet.

body coordinate frame

A coordinate framework centered on an entity. To represent the orientation of an entity, an application uses Euler angles or a rotation matrix that rotates from a reference frame to the body coordinate frame.

channel

A channel renders the view of an observer in a 3D visualization application.

clipping

A feature that prevents the display of terrain and objects or parts of objects, that are closer than or farther than specific distances from the observer.

clipping planes

The range of distances from the observer (near clipping and far clipping) in which an application clips objects.

culling

The process of discarding database objects which are not within view, and therefore need not be rendered and displayed.

Cartesian coordinates

A system for indicating location by means of three planes intersecting at right angles to one another at the origin.

Data Distribution Management

The set of HLA services used to specify the routing of data between publishers and subscribers. DDM adds greater control over the Declaration Management (DM) services that only specify filtering based on the class of the data (e.g., ground vehicle versus aircraft). DDM uses the data content to express regions of interest and regions of affect whose overlap establish a communication channel. An example is geographic filtering where a publisher indicates a region around the entity or effect and a subscriber indicates a region expressing its sensor range.

dead-reckoning

A process by which an application calculates the expected location of an entity during periods between state updates, based on velocity, acceleration, and rotational velocity.

DDM

Data Distribution Management.

Defense Modeling and Simulation Office

The former name of the executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. Renamed Modeling and Simulation Coordination Office. Currently the MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness.

DIS

Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

display engine

An object in an application that can render 3D data.

Distributed Interactive Simulation

The DIS protocol is a set of standards that govern how participating applications share information about the virtual world. The protocol specifies a set of packets, called protocol data units (PDUs), that communicate this information. Each PDU identifies the sender and contains other information depending on the PDU type. The DIS protocol also specifies when and how frequently PDUs are sent.

The DIS protocol has been superseded by the High-Level Architecture for use by the Department of Defense.

DMSO

See Defense Modeling and Simulation Office.

element definition

In VR-Forces and VR-Vantage, an element definition specifies all of the visualizers that control the various aspects of visualizing a scene element. For example, an element definition includes the main model or military symbology, trailing effects, cockpit displays, and so on.

emitter

A simulated device on an entity that emits electromagnetic radiation, for example, radar.

entity

An element in a simulation, such as a vehicle or a person, that is represented in the simulation through the issuance of state messages.

entity maintenance

A process by which the MAK Data Logger compensates for discontinuities following a time jump. The Logger sends out interim state messages for entities that were present before the time jump so that they do not time out before the next update message arrives.

entity-type

A list of seven components as specified by the DIS protocol and the HLA RPR FOM. If none of the seven components contains a wildcard (-1) value, then the entity type refers to a specific, narrowly-defined entity. If some components contain a wildcard, then the entity type refers to a class of entities. A larger number of wildcards indicates a broader, more general class.

The components of an entity-type are:

- Entity kind
- Domain
- Country
- Category
- Subcategory
- Specific
- Extra.

environmental process

According to the IEEE 1278.1a specification (DIS), environmental process PDUs communicate simple environment variables, small scale environmental updates, and embedded processes.

Euler angles

A set of three angles used to describe the orientation of an entity as a set of three successive rotations about three different orthogonal axes (x, y, and z). These angles specify successive rotations needed to transform from a reference coordinate system to the entity's body coordinate system.

event

An interaction between objects or between an object and the terrain, such as firing of a munition, or a collision of entities.

exercise

The DIS term for a one or more interacting simulation applications. Compare to federation execution in HLA.

exercise connection

The object (*DtExerciseConn*) through which a VR-Link application connects to the network. State messages are sent through the exercise connection and information about remote entities is received through the exercise connection. (All MAK products are VR-Link applications.)

exercise ID

A numeric identification for a DIS simulation exercise.

FED file

Federation Execution Data file. The Federation Execution Data is the subset of FOM data needed and read by the RTI. VR-Link applications also read the FED file.

federate

A connection to the RTI. Typically a single simulation application can be thought of as a federate.

federation

A group of HLA federates capable of playing in the same federation execution.

Federation Object Model

Defines the data content of a federation execution.

federation execution

The federation execution represents the actual operation, over time, of a subset of the federates and the RTI initialization data taken from a particular federation. A federation execution is the logical equivalent of a DIS simulation exercise.

field of view

Controls the perspective of the observer.

A wide field of view creates an effect like that of a wide-angle camera lens. Objects appear smaller and farther away from the observer, since the observer coverage spans a wider area. Depth become exaggerated.

A narrow field of view creates an effect like that of a telephoto lens. Objects appear larger and closer to the observer, and the overall scene depth appears flattened. The distances between objects appears compressed.

filter range

A setting that prevents distant entities from being processed while allowing distant terrain to appear normally.

FOM

See Federation Object Model.

FOM Module

In HLA Evolved, defines additional modular data content for a federation execution, usually extensions to an existing FOM. (Also supported by the HLA 1.3 and HLA 1516 versions of the MAK RTI. Per the HLA Evolved interface specification:

"A partial FOM (containing some or all OMT tables) that specifies a modular component of a FOM. A FOM module may contain classes not inherent to it but upon which the FOM module depends, i.e., superclasses to the modular components. These superclasses will be included in the FOM module either as complete or scaffolding definitions."

frame rate

The rate at which the application displays updated images.

ground clamping

A process by which the a 3D visualization application keeps an entity anchored to the surface of its terrain database, regardless of the altitude data contained in its entity state message.

geocentric coordinates

A coordinate system calculated with respect to the earth's center. The origin of the geocentric coordinate system is the center of the earth. The positive X-axis passes through the prime meridian at the equator; the positive Y-axis passes through 90 degrees east longitude at the equator; and the positive Z-axis passes though the north pole.

geodetic coordinates

A coordinate system in which position is determined relative to a reference ellipsoid, such as the surface of the earth at sea level. In MAK applications, geodetic coordinates consist of latitude and longitude in radians, and altitude in meters above the reference ellipsoid.

gridded data

Data that has been processed in a rectangular array of points, in X, Y or latitude/longitude, at which single data values define a two dimensional function. According to the IEEE 1278.1a specification, gridded data transmits information about large-scale or high-fidelity spatially and temporally varying ambient fields and about environmental processes and features.

guise

An alternative entity type used to display an object depending on the force ID. For example, a tank could look like an M1A1 to friendly forces and a T72 to hostile forces.

head-up display

A set of indicators and readouts superimposed onto a graphics display. Also called an overlay.

heartbeat

In DIS, the frequency with which current PDUs are sent to the network regardless of whether or not the entity's state has changed.

High-Level Architecture

The High Level Architecture (HLA) for simulations is a U. S. Department of Defense (DOD)-wide initiative to provide an architecture to support interoperability and reuse of simulations. The HLA supersedes DIS for the DOD.

HLA

See High-Level Architecture.

interaction

A message describing a simulation event. An interaction describes an event, it does not update an object's state.

Logger control PDUs

A set of DIS PDUs or HLA interactions that let you control the MAK Logger remotely.

logical range

A suite of TENA Resources, sharing a common object model, that work together for a given range event. Similar in concept to the HLA Federation.

Local RTI Component

The libraries on a computer that implement an RTI. A federate communicates with the HLA network through the LRC.

LRC

See Local RTI Component.

MAK Technologies Lisp

An adaptation of the Lisp language used in configuration files for MAK products.

message

A general term used to refer to DIS PDUs and HLA interactions and state updates. In TENA, a message is similar to an HLA interaction.

MIM

The MOM & Initialization Module (MIM) allows for extensions to be made to the HLA standard MOM. It is a subset of the FOM that contains OMT tables that describe the HLA MOM. The MIM also contains additional predefined HLA constructs such as object and interaction roots, data types, transportation types and dimensions. HLA specifies a standard MIM that is incorporated into all FDDs automatically by the RTI. The standard MIM can be replaced with a user-supplied MIM containing the standard MIM plus extensions.

model definition

In VR-Forces and VR-Vantage, a model definition specifies the 3D model and other basic attributes of a model. It is referenced by an element definition.

Modeling and Simulation Coordination Office

The executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. The MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness. (Formerly DMSO)

MSCO

Modeling and Simulation Coordination Office. (Formerly DMSO.)

MTL

See MAK Technologies Lisp.

orientation clamping

Adjusts the pitch and roll of an entity so that it appears properly seated when the terrain is inclined. For example, if a tank is moving horizontally across the face of a hill, orientation clamping prevents the tank from appearing level, and therefore, partially embedded into the hillside. Used with ground clamping.

object

An element in a simulation that has persistence, as opposed to an interaction, which is a transient element.

object handle

An integer that an application uses to identify a particular object in RTI service calls. An object handle is meaningful only to a particular federate. The same object can be known to different federates by different object handles.

object name

A character string that can be used to identify an object. In HLA, the object name is known to the RTI, and the RTI provides functions to find out an object's name, given its handle, and vice versa. Object names can be chosen by applications that register the objects with the RTI, however if you do not want to choose names for objects, the RTI will assign names for you.

observer

In MAK 3D applications, the point of view into the scene. (Called the eyepoint in MAK Stealth 6.x and earlier.)

PDU

See Protocol Data Unit.

Protocol Data Unit

A unit of data message (packet) that is passed on a network between DIS simulation applications.

radio transmitter

A simulated device on an entity, capable of transmitting radio communications.

radio receiver

A simulated device on an entity, capable of receiving radio communications.

Realtime Platform Reference FOM

An HLA reference FOM based on the DIS protocol.

recording

A Data Logger file that stores a history of the interactions in a simulation for playback.

reflected entity list

A list of entities simulated by remote applications (federates in HLA) and about which the local simulation has received information over the network.

reference FOM

A FOM designed to be used as a whole, or with modification, by a wide variety of similar federation executions.

relative dead reckoning

A method of dead reckoning in which an application approximates the location of an object based on the local time of receipt of the last state update message.

remote display engine

A display engine running in an application that is separate from the master application, and which is controlled by the master application.

RID file

See RTI Initialization Data.

RTI Initialization Data

The initialization data required by the RTI for operation.

Data required by an RTI during initialization, independent of the FOM being used. RID data is usually dependent on a specific implementation of the RTI.

RPR FOM

See Realtime Platform Reference FOM.

RTI

See Run-Time Infrastructure.

Run-Time Infrastructure

A library and other supporting software that implements the HLA interface specification. All federates communicate with one another in an HLA environment through RTI functions.

scene element

In VR-Vantage and VR-Forces, any object that is rendered in the scene, such as an entity model, interaction, or tactical graphic.

Simulation Interoperability Standards Organization

A group that seeks to promote modeling and simulation interoperability and reuse for the benefit of diverse M&S communities, including developers, procurers, and users, world-wide.

simulation time

In VR-Forces, simulation time is used in dead-reckoning of remote entities and thresholding of local entities. Typically, simulation time is set once during each iteration of the application's main simulation loop so that all entities are dead-reckoned based on the same value of current time.

SISO

See Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

A method of ensuring that transitions from an entity's dead-reckoned position to its actual position are not so abrupt as to be visually disconcerting.

state

The current status of an object, including location, direction of movement, extent of damage, and so on.

tape

An alternative term for a Logger recording. Not a physical magnetic tape.

terrain following

In a 3D visualization application, causes the observer (eyepoint) to maintain a constant distance above the terrain surface. The observer's height changes in tandem with the peaks and valleys as it passes over the geography.

timeout

The period of time in which an application continues to display an entity after that entity's update messages have stopped appearing on the network. Time-outs are usually not used in HLA because there is no set frequency (no heart-beat) for transmitting messages.

timescale

A factor by which time is accelerated or slowed during playback of a Data Logger recording.

topographic coordinates

A right-handed Cartesian coordinate system whose X-Y plane is tangent to the earth's surface at the origin, with the positive X axis pointing north, the positive Y axis pointing east, and the positive Z axis pointing down. There are an infinite number of topographic coordinate systems – one for each point on the earth's surface.

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

A method used by to smooth positional discontinuities that could occur when new state updates arrive.

UDP port

A network channel through which an application sends and receives data for DIS exercises.

Universal Transverse Mercator

In general, a non-cartesian coordinate system in which the X, Y, and Z correspond to easting (nearly east), northing (nearly north), and height above an ellipsoid that approximates the surface of the earth.

UTM

See Universal Transverse Mercator.

view control messages

A set of programmatic messages that let you control the view in VR-Vantage applications remotely.



Index

--enableVSync command-line option 6-24
--noAppDataEntityTypeMap command-line option 89-9
-- command-line option 5-2, 5-12, 5-16, 63-4
  Simulation Object Editor 67-4
--additionalSystemScriptsDirectory command-line option 5-9
--alphaBits command-line option 5-2
--antiAliasing command-line option 5-2, 6-16
--appDataDir command-line option 5-2, 5-9
  Simulation Object Editor 67-4
--appIdRange command-line option 5-8, 5-9
--appNumber command-line option 5-7, 5-8, 5-9
--attributePduStrategy command-line option 5-14
--autoConnect command-line option 5-2
--backend command-line option 5-16
--col command-line option 63-4
--config command-line option 5-16
--countryCodesMappingFile command-line option 5-10
--daemon command-line option 5-10
--dataDir command-line option 5-2, 5-10
  Simulation Object Editor 67-4
--defaultObjectTimeout command-line option 5-8, 5-10
--defaultSimulationModelSet command-line option 5-2
--depthBuffer command-line option 5-2
--destAddrString command-line option 5-8, 5-13
--deviceAddress command-line option 5-8, 5-14
--diGuyAnimationsFile command-line option 5-10
--diGuyCharacerDataFile command-line option 5-10
--dis command-line option 5-2
--disableDynamicTerrain command-line option 5-2
--disableKDTrees command-line option 5-4
--disableOcclusionCulling command-line option 5-2
--disableParallelTick command-line option 5-10
--discoveredModelPreloadFileName command-line option 5-2, 6-25
--disPort command-line option 5-9, 5-14
--dispSetting command-line option 5-2
--disVersion command-line option 5-8, 5-14, 5-22
--doNotCheckPluginVersions command-line option 5-3
  Simulation Object Editor 67-4
--doNotLoadVrfPlugins command-line option 5-3, 5-10
  Simulation Object Editor 67-4
--enableChannel command-line option 5-10
--enableLogFileTimestamps command-line option 5-3
--enableVSync command-line option 5-3

--encryptImages command-line option 63-4
--entDispSetting command-line option 5-3
--entityDefsDir command-line option 5-3, 88-17
--entTypeMap command-line option 5-3
--envSetting command-line option 5-3
--execname command-line option 5-8, 5-13
--exerciseId command-line option 5-9, 5-15
--extendLevelZero command-line option 63-4
--extraArgs command-line option 5-3
--factoryRootDir command-line option 5-3
--federateName command-line option 5-8, 5-13
--federateType command-line option 5-8, 5-13
--fedFileName command-line option 5-7, 5-12
--file command-line option 63-4
--fileNotifyLevel command-line option 5-10
--fileTransporterReceivePort command-line option 5-3, 5-8
--fomMapperInitData command-line option 5-7, 5-12
--fomMapperLib command-line option 5-7, 5-12
--fomModules command-line option 5-7, 5-12
--forceTextShaping command-line option 5-3
--forceVBO command-line option 5-3
--fowPerspective command-line option 5-3
--frameRate command-line option 5-3
--frontend command-line option 5-16
--frontEndPID command-line option 5-10
--fullScreen command-line option 5-3, 83-8
--fullyLoadTerrain command-line option 5-10
--generateFiles command-line option 63-5
--gui command-line option 5-4, 84-2
--guiArgs command-line option 5-16
--help command-line option 5-4, 63-5
  Simulation Object Editor 67-4
--highestPriority command-line option 5-4
--hla13 command-line option 5-4
--hla1516 command-line option 5-4
--hla1516e command-line option 5-4
--hostAddressString command-line option 5-8, 5-9, 5-10
--ignore_rest command-line option 5-2, 5-12, 63-4
  Simulation Object Editor 67-4
--ignoreNonBaseTex command-line option 90-36
--ignoreZoomForSpeedtreeLod command-line option 5-7
--launcherLocation command-line option, Simulation Object Editor 67-4
--layoutSettingsFile command-line option 5-4
  Simulation Object Editor 67-5
--level command-line option 63-5
--loadObservers command-line option 5-4
--loadPlugin command-line option 5-4, 5-10, 5-19
  Simulation Object Editor 67-5
--loadSimulationObjectGroup command-line option 5-4
--locale command-line option 5-3, 5-16
  Simulation Object Editor 67-4
--localSettingsDesignator command-line option 5-8, 5-13
--logFileName command-line option 5-4, 5-11
--mcastTtl command-line option 5-9, 5-14
--mergeEntityTypeMap command-line option 5-5
--mergeHierarchyEntityDef command-line option 5-5, 88-17
--mergeHierarchyTacticalGraphicsDef command-line option 5-5, 88-17
--mergeHierarchyUnitDef command-line option 5-5, 88-17
--mergeLeafEntityDef command-line option 5-5, 88-17
--mergeLeafTacticalGraphicsDef command-line option 5-5, 88-17
--mergeLeafUnitDef command-line option 5-5, 88-17
--mergeModelDefs command-line option 5-5, 87-12
--mergeTacticalGraphicsTypeMap command-line option 5-5
--mergeUnitTypeMap command-line option 5-5
--mimModule command-line option 5-12
--missing command-line option 63-5

--modelPreloadFileName command-line option 5-5, 6-25
--msdlSIDCMappingFile command-line option 5-11
--multicastAddresses command-line option 5-9, 5-14
--noAppDataEntityDefs command-line option 5-5, 88-17
--noAppDataEntityTypeMap command-line option 5-6
--noAppDataModelDefs command-line option 5-6, 87-12
--noAppDataTacticalGraphicsDefs command-line option 5-6, 88-17
--noAppDataTacticalGraphicsTypeMap command-line option 5-6
--noAppDataUnitDefs command-line option 5-6, 88-17
--nonVrfObjectsUUIDScheme command-line option 5-6, 5-11
--noRtiCompilerCheck command-line option 5-13
--notifyLevel command-line option 5-5, 5-11, 67-5
--numCallbackThreads command-line option 5-11
--outputLogFile command-line option 5-11
--plugin command-line option 5-6
--pseudoFullScreen command-line option 5-6, 83-8
--publishMftOnly command-line option 63-5
--pvd command-line option 5-6
--recvBufferSize command-line option 5-9, 5-14
--regenerateAll command-line option 63-4
--removeHighestLod command-line option 90-37
--reportedHostAddressString command-line option 5-6
--row command-line option 63-6
--rprFomRevision command-line option 5-8, 5-13
--rprFomVersion command-line option 5-8, 5-13
--scaleLod command-line option 90-37
--scenarioFile command-line option 5-7
--searchDir command-line option 63-5
--sendBufferSize command-line option 5-9, 5-14
--sessionId command-line option 5-8, 5-9
--setMainThreadToHighPriority command-line option 5-11
--settingsDirectory command-line option, Simulation Object Editor 67-5
--settingsFile command-line option, vrfSim 5-10
--showConsole command-line option 5-2
Simulation Object Editor 67-5
--simArgs command-line option 5-16
--simulationModelSet command-line option 5-11
--siteId command-line option 5-8, 5-9
--smsFile command-line option, Simulation Object Editor 67-5
--startMinimized command-line option 5-12
--stealth command-line option 5-7
--stencilBits command-line option 5-7
--subDir command-line option 63-5
--subnetMask command-line option 5-9, 5-14
--suppressDataAvailableException command-line option 5-12
--suppressHatIntersectOnAttach command-line option 5-7, 52-7
--suppressSelfReflect command-line option 5-14
--synchronizeOcean command-line option 5-6
--targetFrameRate command-line option 5-12, 6-22
--textureSize command-line option 63-6
--threads command-line option 63-5
--timeManagement command-line option 4-20, 5-13, 5-21
--unbatchedRendering command-line option 5-4
--unplug command-line option 5-7
--useAbsoluteTimeStamps command-line option 5-7, 5-12
--useAdvisories command-line option 5-8, 5-12
--useAsyncIO command-line option 5-9, 5-14, 5-22
--useFileCommChannel command-line option 5-7
--useGeographicDdm command-line option 5-8
--useIpv6 command-line option 5-9, 5-15

--userDataDir command-line option 5-7, 5-12
--useReverseZBuffer command-line option 5-7
--verbose command-line option 90-37
--version command-line option 5-7, 5-12, 5-16, 63-5
  Simulation Object Editor 67-5
--visualTerrain command-line option, Simulation Object Editor 67-5
--vtFile command-line option 63-5
--waitQueue command-line option 5-12
-b command-line option 90-36
-debug command-line option 5-16, B-11
-e command-line option 5-3
-G command-line option 5-16
  Simulation Object Editor 67-4
-h command-line option, Simulation Object Editor 67-4
-K command-line option 5-4
-r command-line option 90-37
-V command-line option 90-37
-v command-line option, Simulation Object Editor 67-5
-Z command-line option 5-7
-z command-line option 5-7
B-- command-line option 5-16, B-11
F-- command-line option 5-16, B-11

Symbols

@dis comment 59-28
@dis navigation_lights 59-28

Numerics

-1 (wildcard) 74-7
2525B icon 90-3
2D
  icon 14-11
    changing 68-16
    changing size 44-3
    color 16-15
    configuring 90-3
    editing 90-3
    label 16-10
    scaling 44-3
    using image for 68-17
  label, shadow text 16-16
  observer, attachment behavior 52-16
2D (continued)
  observer coordinate system 51-7
  projection, switching to 3D 50-2
  window 4-13
2D Icon model set 50-11
3D
  model 14-11
    changing 68-14
    DI-Guy 28-7
    licensing 1-21
  observer coordinate systems 51-7
  projection, switching to 2D 50-2
  window 4-13
3D Cartesian coordinate system 32-20
3D Colorized Models model set 50-11
  color, changing 16-19
3D Models model set 50-11
3DS
  ASCII Export 69-27
  ASE 26-4
  ASE file 69-24
  file 54-3
3DS ASCII Export format 26-4
900_ExtImagery.xml 60-18
9-Line briefing 27-7

A

-A command-line option 5-8, 5-13, 5-22, 63-4
-a command-line option 5-7, 5-8, 5-9, 5-18
Abandon Plan command 37-44
abandoning plan 24-1, 37-44
  confirmation prompt 25-5
abatis, constructing 30-12
absolute, pathname 12-7
Absolute mode 52-8
acceleration
  displaying 16-10
  specifying measurement unit 83-12
acceleration-to-rpm-factor parameter 9-25
accessibility check 19-6
ACO 7-20
action category 25-34, 31-8, 31-24
action table 73-4
actionCategories.tsk 25-34, 31-24
Activate Jammer task 30-4
activating
  jammer 30-4
  scenario event 8-10

activating (continued)
scenario event from plan 8-12
spawn points 18-4
activation, indirect fire 40-24
Activation set data request 41-2
active
sonar 9-24
stereo 82-21
tactical graphic 40-24, 41-2
Active Sonar Mode dialog box 34-4
Active Sonar Mode set data request 34-4
active-radar-sensor system 22-19
activity
adding 71-16
unit 35-3
Activity dialog box 35-3
Activity set data request 71-16
actuator 13-7, 13-8
indirect fire 76-17, D-9
missile-collision 76-16, 76-17
missile-maneuver 76-16, 76-17
warhead 76-20, D-12
warhead detonation 76-19, D-11
Add Cloud Layer dialog box 11-18
Add Conditional Expression dialog box 36-5
Add Group of Entities, dialog box 19-9
add image attribute 57-7
Add Jamming Target dialog box 34-4
Add Jamming Target set data request 34-4
Add Model Definition dialog box 87-5
Add Props page 57-22
Add Terrain Patch dialog box 57-4, 66-8
Add to Order of Battle dialog box 20-4
Add Visualizer Definition Attribute window 88-12
adding
activities 71-16
attribute, visual definition 88-12
category, Props Palette 90-14
channel 82-4
cockpit model 90-10
default overlays 68-9
DI-Guy animation 73-3
DI-Guy character 73-3
entity bitmap image 90-9
environment condition 11-7
extended label 16-18
feature layer 54-4, 57-11
adding (continued)
formation 70-9
global commands to plan 37-31
group of entities 19-9
image 68-18
key mapping 85-3
linked event 8-6
media to scenario event 8-5
model, visual 68-18
model definition 68-18, 68-20
model mapping 89-4
object, to Favorites list 14-36
object geometry 69-7
observer 50-6, 50-7
ocean layer 57-15, 57-16
parameter
model definition 87-8
schema 86-5
parameter properties 74-27
pedestrians, to crowd 28-4
plug-in 4-36
power type 77-10, D-20
prop 57-20
from feature layer 57-22
sensor component 75-13
sensor signature 75-5, 75-14
sensor type 75-11
simulation objectto unit 23-7
slot 72-2, 72-5, 72-11
state properties 74-27
subordinate to unit 70-4
system to simulation object 69-16
tag 67-27
terrain patch to terrain 57-4
terrain server 58-4
text, to title bar 83-11
text object 40-5
vertex 40-18
visual definition 88-8
window 82-2
additional-algorithm-config parameter 77-17
additionalDestinationAddresses parameter B-7
additional-opds parameter 12-6
additionalSystemScriptsDirectory parameter B-2
addMulticastAddr parameter B-7
advanced navigation 61-2
Advanced Visual Model Editing dialog box 68-22
advisory message, enabling and disabling B-7

Affects Controls check box 31-21
agent, contamination 22-9
aggregate*, See also* unit
Aggregate As
  dialog box 70-3
  specifying a unit as option 70-3
aggregate warfare model 22-3, 71-2
aggregate-comms-jammer, system 22-18
aggregated state 23-9
aggregated unit 21-2
  attack by fire 30-5
aggregate-formation-controller parameter 78-2
aggregate-level
  combat 22-9
  entity, configuring 71-2
  modeling 13-9
aggregate-level modeling 20-1
aggregate-level unit, parameter 71-3
AggregateLevel.sms 13-9, 22-3, 71-2
aggregate-object-param parameter 74-39
aggregate-object-param parameter 74-12
aggregate-param parameter C-8
aggregate-radar-jammer system 22-19
aggregateSpatialModelTickAsFastAsPossible-WhilePaused parameter B-2
aggregateSpatialModelTickPeriod parameter B-2
aggregateSpatialModelTickPeriodUsesRealTime parameter B-2
aggregateSpatialModelTickVariance parameter B-2
aggregating objects in Order of Battle 20-5
aggregation
  automatic 23-12
  manual 23-12
  state 23-6
agility, terrain 1-12
AI 34-5
AI Enabled dialog box 34-5
AI Enabled set data request 34-5
aiming, sensor 34-38
Aiming dialog box 34-6
Aiming set data request 34-6
air, temperature 11-6
air base
  assigning aircraft 35-4
  attacking 30-4
  facilities, setting 35-8
  landing at 30-25
  personnel, setting 35-13
air corridor 40-5
air crew, setting for air base 35-13
air defense 30-7
Air Drop task 26-3
air temperature, ambient 11-10
air traffic, generating 26-11
air unit
  attacking air base 30-4
  combat 71-24
  fuel tank, dropping 30-21
aircraft
  attacking with gun 27-3
  landing at air base 30-25
  landing at airport 26-11
  refueling 30-30
  See also Fixed-Wing entity and Rotary-Wing entity
  setting speed 34-30
  strafing target 27-22
aircraft assignment, deleting 35-6
airlift 26-26
  cargo 26-27
airport, landing aircraft at 26-11
airspace 40-5
airspace control order 7-20
Air-to-Air Attack dialog box 30-4
Air-to-Air Attack task 30-4
Air-to-Ground Air Base Attack dialog box 30-4
Air-to-Ground Air Base Attack task 30-4
Air-to-Ground Attack dialog box 30-5
Air-to-Ground Attack task 30-5
alias, attribute transformation 65-4
Allow Crossing dialog box 41-3
Allow Crossing set data request 41-3
allowed-state-repositories-types parameter D-26
all-tasks-exclusive parameter 25-33
alpha 90-34
alpha bits 5-2
alpha to coverage 6-17
  VRV_ALPHA_TO_COVERAGE_REN-DERBIN environment variable 6-17
Altitude
  condition 36-9
  dialog box 34-7
  set data request 34-7
altitude 16-9, 32-19
  entity, displaying 16-20
  fixed-wing entity 25-23
  flying to 26-8, 26-9
  intervisibility line 48-14

altitude 16-9, 32-19 (continued)
object, changing 14-25, 40-18
observer, filtering simulation objects by 6-11
ordering entity to 26-14
pasted object 14-30
route, setting 14-19
setting 14-18, 14-19, 34-7
fixed-wing entity 25-24
in dialog box 14-18
manually 14-18
specifying measurement unit 83-12
switching from 2D to 3D 50-2
testing entity for 36-9
vertex
changing 40-19
displaying 38-8
altitude point
changing color 55-5
deleting 55-4
inserting 55-4
Altitude set data request 25-24
Always Join Session with Open Database 4-18
Always Join Session with Session Database 4-18
ambient
color 45-4
occlusion map 1-15, 64-10
value, material 45-4
ambient air temperature 11-6, 11-10
ammo parameter D-6
ammo select file 76-8, D-2, D-27
ammo select table D-6
counter measure 9-26
ammo-entry 77-2, D-13, D-22
ammo-entry block D-31
ammunition 22-15
amount 35-3
pacing and tracking 35-4
specifying amount 34-34
status 16-6
unit 71-4, 71-7
Ammunition dialog box 35-3
Ammunition Pacing/Tracking dialog box 35-4
Ammunition Pacing/Tracking set data request 35-4
anaglyphic stereo 82-21
configuring 82-22
angle, damage 77-8, D-19, D-23
Angle of Attack 90-12
angle-of-incidence parameter 77-8, D-19, D-23
angleSegmentMax parameter 91-5
animated movement
configuring 69-24
movement file, configuring 69-29
Animated Movement dialog box 26-4, 69-29
Animated Movement task 26-4
animation
attack 43-10
DI-Guy 5-10, 28-7
adding 73-3
choosing 73-5
entity 26-4
fast roping 26-6
model 92-12
observer view transition 51-31
testing DI-Guy 36-10
Answer Questions button 32-25
anti-air
missile 30-6
resource 22-15
weapon 30-7
anti-aliasing 5-2, 6-16
anti-personnel 22-15
anti-ship
missile 30-6
resource 22-15
Anti-ship (Fixed Depth) task 27-16
anti-submarine, missile 27-13
anti-tank
ditch, improving 30-24
resource 22-15
anti-tank ditch, digging 30-12
Any, as condition parameter 37-23
Any Subordinate of Selected Aggregate check box 37-24
AOA updater 90-12
apparent signature 75-9, 75-16
appData directory 5-9
appearance
bit 45-6
DI-Guy 5-10
random 69-13
pasting 14-29
power plant 9-24
setting 34-8
specifying for DI-Guy model 69-11

appearance (continued)
testing DI-Guy 36-10
Appearance dialog box 34-8
Appearance set data request 34-8
appending, saved views to view list 51-34
appIdRange parameter B-7
application
data, specifying location of 5-2, 67-4
ID 3-4
number 5-7, 5-8, 5-9, 7-50, 12-9
configuring B-2
default 5-18
specifying at startup 5-18
remote, notifying of VR-Forces run and pause 7-50
application ID 5-8
Application Settings dialog box 7-28, 23-11
File Caching Settings page 6-13, 64-2, 64-6
General Application Settings page 4-14, 4-24
Key Mapping Editor page 85-3, 85-7
Mouse Settings page 51-14
Performance Settings page 6-19, 6-22
Script Settings page 6-6, 31-22, 31-44
Spot Report Settings page 9-3, 9-4, 9-5, 9-7, 9-8, 9-9, 34-41
applying
auto tags 67-30
platform update 67-25
appNumber parameter B-2
arc, segment 91-4
architecture, component 13-7
area 38-2, 38-5
creating for multiple entity creation 19-8
disaggregation 23-12, 38-2
drawing 40-2
dynamic terrain 62-3
extruded 36-11, 40-5
feature, list of 55-13
fill style 40-23
fortified, constructing 30-17
hazard 22-10
high fidelity damage 77-17
navigation 61-7
NBC 22-9
pedestrian 19-10
suppressive fire at 25-30, 27-19
terrain draw 55-18
area 38-2, 38-5 (continued)
terrain page-in 38-2, 58-6
types 40-2
vector feature draw 55-15
areal, feature 59-13
Arm Mine at Depth task 27-3
Armed dialog box 34-9
Armed set data request 34-9
arming
IED 34-9
mine, naval 27-3
armor-piercing power domain 77-3, D-14
arrow
adding to line 40-23
freehand line 40-3
sensor contact line 43-16
Arrowhead Style dialog box 41-3
Arrowhead Style set data request 41-3
articulated part 92-2, 92-8
attaching to 52-5
displaying 68-21
tasks for moving 29-16
artificial intelligence 34-5
artillery
Fire-for-Effect task 27-11
unit 30-25
ASCII Export file 69-24, 69-27
ASE file 69-24, 69-27
ASE format 26-4
Ask to Join Session 4-18
assembly 71-4, 71-10
adding to entity 71-13
creating 71-11
editing 71-12
removing 71-13
roll up rule 71-14
rolling up 71-15
assertOnBlockingTerrainCalls parameter 54-8, B-2
Assign Aircraft dialog box 35-5, 35-6
Assign Aircraft set data request 35-4
assigning
echelon ID 13-6
plan to multiple simulation objects 37-41
session ID 4-9
assigning aircraft, air base 35-4
Assume Unknown is Hostile dialog box 34-9
Assume Unknown is Hostile set data request 34-9

asynchronous I/O 5-22, B-8
  using for performance 6-14
atlas, texture 90-22
attach 52-8
  observer, articulated part 52-5
Attach Components to Remote Entities On parameter 7-5
attach mode 1-14, 51-3, 51-8, 52-8
  2D behavior 52-16
  Absolute 52-8
  Follow 52-9
  Mimic 52-10
  Mimic Location Track 52-11
  Mimic Track 52-11
  movement, attached context 51-8
  prop, affect of 52-7
  Space Follow 52-12
  Tether 52-13
  Tether Location Track 52-14
  Tether Track 52-14
  Track 52-15
Attach Object to Mouse check box 14-21
attached, context 51-8
Attached Near Clip Altitude, attribute 82-11
attaching observer
  to object, with keyboard 52-4
  to simulation object or prop 52-2
attachment
  list, filtering 52-6
  order 52-4
  preservation of view changes 52-4
Attachment Panel 52-3
  filtering attachment list 52-6
attack
  air-to-ground 30-5
  animation 43-10
  anti-air missile 30-6
  indicator 43-10
  limited munition 22-9
  missile 22-9
Attack Aircraft with Guns task 27-3
Attack By Fire dialog box 30-5
Attack By Fire task 30-5
attack parameter, edting 71-22
Attack with Anti-Air Missile dialog box 30-6
Attack with Anti-Air Missile task 30-6
attack with anti-ship missile 30-6
Attack with Anti-Ship Missile dialog box 30-6
Attack with Anti-Ship Missile task 30-6
Attack with Land-Attack Missile dialog box 30-6
Attack with Land-Attack Missile task 30-6
Attack with Torpedo dialog box 30-7
Attack with Torpedo task 30-7
attacking, air base 30-4
Attention Grabbing dialog box 34-9
Attention Grabbing set data request 34-9
attribute
  Attached Near Clip Altitude 82-11
  BOYSHP 59-23
  buoy 59-23
    building model definition 59-23
  cockpit updater 90-10
  COLOUR 59-23
  COLPAT 59-23
  DESIGNATOR 59-25
  destroyedsymbol 90-8
  Dynamic Near Clip Attached Policy 82-11
  Dynamic Water Visibility Enabled 82-16
  Dynamic Water Visibility Maximum 82-16
  earth file, differential processing 59-18
  feature 65-4
  Fixed Near Clip 82-11
  Fixed Near Clip When Attached 82-11
  image 57-7
  imagesymbolmap 90-7, 90-9
  itemname 84-2, 84-9
  light characteristic 59-30
  LITCHR 59-30
  MAK_EARTH_FILE 65-4
  MAK_LAYER 65-4
  MAK_SOURCE_FILE 65-4
  marker-scale 90-32
  modeldefinitionschema 90-7
  Near Far Clip Policy 82-10
  OBJNAM 59-25
  parent 90-6
  Reduce Z Fighting Ratio 82-10
  sensor 82-7
  setting simulation object 34-8
  SIGGRP 59-30
  signal group 59-30
  signal period 59-30

attribute (continued)
SIGPER 59-30
SIGSEQ 59-30
TOPSHP 59-30
transformation, alias 65-4
transforming 65-4
type 90-6
visual definition 88-9
adding 88-12
WidgetTheme 90-7
window 82-6
attrition
rate 22-4
unit 22-12
audio 49-2
adding to scenario event 8-5
Audio Settings dialog box 49-2, 49-3
authentication, proxy 58-5
Auto Generate 57-20
Auto Launch dialog box 34-12
auto tag, applying 67-30
automatic
aggregation 23-11
aggregation and disaggregation 23-12
attachment 40-16
counter measures 9-26
embarkation 17-6
laser 34-24
reorganization 34-33
running of scenarios 7-45
Automatic Air Defense task 30-7
Automatically Close Doors, set data request 34-10
Automatically Close Doors dialog box 34-10
Automatically Join Session 4-17
Automatically Open Session Database without Joining Session 4-17
autonomous lasing 9-19
auto-reorganize parameter 12-5
autoTagInformation.csv 67-30
Avoid Roads 34-28
avoidance, collision 34-11, 61-6
avoiding
collisions, obstacles, and features 13-11
objects 78-5

B

-B command-line option 5-9, 5-16, 7-48
back-end 3-2
balancing load 7-30
configuring feature data 57-12
console B-3
dropping from exercise 3-7
enabling time management 4-20, 5-21
feature representation 54-5
heartbeat B-6
late joining 3-7
mapping objects to 12-9
missing 3-8
multiple 3-2, 3-7, 12-9
reloading OPD B-3
remapping 3-8
in batch mode 7-48
sending messages 3-3
starting 4-3
as daemon 5-10
synchronizing 4-19
background
2D label 16-16
intervisibility information 48-14
background process 30-34, 31-5, 32-32
balancing, object load 7-30
ballistic gun, controller descriptor D-6
ballistic missile 10-8, 27-14
firing 10-9
movement, configuring 69-25
movement file, configuring 69-29
scripted movement 69-24
target event 10-11
Ballistic Missiles dialog box 69-25, 69-29
ballistic-missile-entity-state-repository 74-12
ballistic-missile-param parameter 74-12
barbed wire, constructing 30-13
barricade, constructing 30-13
batch file 7-45
command-line option 5-9
configuring B-2
editing 7-46
batch mode 7-45, 7-46
remapping 7-48
starting from command line 7-48
batchScenarioFileName parameter B-2
beacon 59-30
model definition 59-23
seasonal 59-29
streaming 59-22, 59-27
topmark 59-30

beacon 59-30 (continued)
wake 11-26
bead
DOF 92-8
switch 92-9
beam 34-32
emitter 9-24, 34-17
standard and tracking 34-17
bearing, intervisibility line 48-14
behavior
scripted 26-4
simulation object 13-8
behavior set 31-9
assigning to force 25-20
creating 25-19
script availability 31-25, 31-27
berm, constructing 30-14
best practice, model design 90-34
bin, render 6-17
binary key mapping 85-3
binding, system definition file variable 74-22
Biological Attack dialog box 30-7
Biological Attack task 30-7
biological contamination 22-9, 22-10, 30-7
bit
depth 5-2
depth and stencil 6-16
bitmap 54-4
adding new 90-9
adding to scenario event 8-5
displaying 57-6
icon 90-4
location of 2-4
blend image attribute 57-7
blending, LOD transition 64-15
block
ammo-entry D-31
damage-by-munition-power 77-8, D-18
damage-model 77-8, D-18, D-22, D-31
determinant 77-8, D-19
high-fidelity-terrain-damage-table 77-17
munition 77-3, D-14
power-list 77-3, D-14
standard-terrain-damage-table 77-16
blur 47-2, 47-4
BMP 54-4
body coordinates 51-3
bomb 76-20, D-12
dropping 27-20
laser guided 27-20
target 27-7
Bomb Location dialog box 30-8
Bomb Location task 30-8
bomb-bay weapon system 76-20, D-12
bomb-entity-state-repository 74-12
bomb-param parameter 74-12
booby trap
creating 30-14
destroying 30-20
improving 30-23
boolean, operator 36-13
boom, refueling 29-16, 29-18
border, icon 44-2
Boundary Generation Tool 59-15
running 59-16
using 59-16
boundary.txt 59-16
bounding box
selecting objects 15-5
unit 70-11
bounding volume 13-11, 68-20
displaying 68-20
drawing for terrain 64-22
simulation objects 16-22, 16-23
specifying 69-3
unit, color 24-10
box
drawing 40-4
extruded 40-5
resizing 40-20
BOYSHP, attribute 59-23
breach, improving 30-23
Breach Obstacles dialog box 30-9
Breach Obstacles task 30-9
breaching
fortification 30-9
obstacle 30-9
breaking, wave 11-27
bridge 14-33, 69-3
building 30-15
destroying 30-20
improving 30-23
brightness 47-4
broadcasting
state update 50-11
stealth position and orientation 50-12
buffer, size 5-9, 5-14

building 14-33, 69-3
  avoiding 13-11
  bridge 30-15
  conditional statement 37-13
  creating entity in 14-17
  ditch 30-16
  dry gap 30-16
  embedding in terrain 14-34
  extruded 59-14
  fortified area 30-17
  prop type 57-26
  setting avoidance 34-11
  terrain 57-4
bump map 1-15, 64-10
buoy
  attributes 59-23
  daymark 59-29
  designator 59-25
  model definition 59-22, 59-23
    using texture atlas 59-26
  seasonal 59-29
  signal sequence 59-30
  streaming 59-22
  topmark 59-30
  unique 59-25
  wake 11-26
buoyancy
  enabling or disabling 46-8
  LOD distance 46-8
burst
  length 18-9, 18-13
  period 18-9, 18-13
  spawn 18-9
  spawn pattern 18-13
button
  Answer Questions 32-25
  common for model definitions 83-10
  Joystick 80-7
  Key Mapping 80-8
  toolbars 4-11
buttonGroup, property 33-6

C

-C command-line option 5-2, 5-16
-c command-line option 5-10, 63-4
C++
  scripted task 31-30
  task 31-3, 32-14
cache
  clearing 64-6
  clearing model instancing 6-13
  model instance 6-13
  osgEarth Cache tool 64-3
  streaming terrain 64-7
caching
  earth file, offline 64-5
  earth file data 64-3
  file 64-2
  terrain 64-3
  terrain data 1-18
  terrain server data 64-3
calculating, support point 69-5
calculating damage 77-2, D-13
camera
  effects 47-2
  jitter 51-25
  moving 51-13
  position and orientation 82-20
  shaking 51-25
  ways to move 51-12
Camera Orientation Offset 82-7
Camera Position Offset 82-7, 82-20
Can Be Executed More Than Once at a Time check box 31-31
can-be-radar-jammed parameter 22-19
canceling, reactive task 25-17
cancelling, generation of navigation data 61-15
can-pivot parameter 26-29
Capabilities dialog box 34-10
Capabilities set data request 34-10
capability, setting 34-10
capture, screen 4-14
car
  displaying driver in 44-8
  getting in and out of 29-10
car bomb, detonating 34-14
car door, opening and closing 29-10
Car Door task 29-17
cargo
  carrying 26-26
  embarking 26-27
  unloading 26-27
cargo door
  closing 29-16
  opening 29-18
carrier, placing entity on 14-17
Cartesian, coordinate system 32-20, 32-22

CAS 27-7
  strafing target 27-22
catastrophic-kill parameter D-23
Categories, clearing list 68-10
category
  action 31-8, 31-24
  echelon 13-5
  Props Palette, adding 90-14
  undoing changes 67-29
  weapon and ammunition, adding 71-27
CDB
  configuration files 60-17
  earth file 60-2
  elevation sub-dataset 60-4
  lightmap 60-6
  lightpoint 60-9, 60-10
    properties 60-11
  osgEarth 60-2
CDB database 60-2
CDB raster driver 60-3
CEO. See combat engineering object
certainty level for spot reports 9-7
cgfDispatcherReceivePort parameter B-2
chaff 27-15
  launching 9-26
Change Hostility dialog box 9-15
Change MOPP Level dialog box 30-10
Change MOPP Level task 30-10, 35-12
Change Posture dialog box 30-11
Change Posture task 30-10, 35-13
changing
  2D icon 68-16
  2D icon image 68-17
  3D model 68-14
  altitude, vertex 40-19
  channel attributes 82-7
  color
    altitude point 55-5
    force 16-19
    tactical graphics 40-21
  colorized model 68-14
  control object 40-17
  entity
    enumeration 68-5
    parameter 69-6
    tag 68-4
  entity name 14-25
  extended label index 16-19
  field of view 82-14
  force hostility 9-13
changing (continued)
  frame rate mode 12-6
  gain 51-29
  grid spacing 70-14
  icon list 68-10
  icon size 44-3
  image display order 57-9
  intervisibility object 48-9
  key mapping 85-5
  line, direction 40-21
  marking 34-27
  model definition, prop 66-11
  model set 50-11, 68-21
  MOPP level 30-10
  name, tactical graphic 14-25
  object tag 68-4
  observer mode 50-3
  order of objects in OOB 20-7
  overlay name 39-5
  posture 30-11, 35-13
  prop
    model definition 57-26
    position or orientation 57-26
    type 57-26
  script ID 31-11
  simulation object, name 68-7
  Simulation Time Scale Toolbar range 7-35
  site ID 12-9
  spot report icon 68-18
  subordinate order 70-6
  terrain database for scenario 12-8
  unit of measurement 83-12
  unit state 23-11
    runtime 23-10
  viewport 82-15
  width of line 40-23
  window attributes 82-6
channel
  adding 82-4
  attribute, sensor 82-7
  changing attributes 82-7
  keyword 82-7
    compass 51-5
  keywords 82-17
  multiple 82-18
  observer 50-6
  projection resize policy 82-11
  property 82-7
  removing 82-5

Channel Name 82-7
ChannelKeyword parameter 90-12
character
  DI-Guy 5-10
    adding 73-3
  specifying for DI-Guy model 69-11
character_animations_table.mtl 73-3
check box
  Can Be Executed More Than Once at a Time 31-31
  Color by Force Type 16-15
  Enable Occlusion Queries 64-19
  Show Occupants 44-8
checking
  version, plug-in 5-3
checking waypoint accessibility 19-6
checkpoint 7-21, 7-38
  deleting 7-42
  disabling periodic 7-41
  periodic, scheduling 7-41
  rolling back to 7-36
  saving 7-39
  saving from snapshot 7-40
  scenario 7-39
Checkpoint Settings page 7-36, 7-41, 7-42, 7-43
Chemical Attack dialog box 30-11
Chemical Attack task 30-11
chemical contamination 22-9, 22-10, 30-11
child, element definition 88-2
choice
  list, in Qt Designer 33-6
choosing, window layout 83-5
chop, wave 11-22
choppiness, ocean 11-25
CID level 9-16
  for firing at target 9-18
  icon, color 9-16
circle, extruded 40-5
Circle Around Location task 28-3
circling, point 26-21
civilian, creating 19-2
Civilian Idle task 28-3
civilian visit point 19-10
clamp image attribute 57-8
Clamp Near Clip When Magnified, policy 53-10, 82-10
clamp to border image attribute 57-8
clamp to edge image attribute 57-8
clamping, entities 54-9
class
  DtSimComponentManager 74-9
  DtSimObjectNetInterface 74-9
  DtSimObjectStateRepository 74-9
  DtSimTaskManager 74-9
  DtVrfObject 74-9
  FeatureSet 32-2
  Location3D 32-19
  Lua 32-2
  SimObject 32-3
  Vector3D 32-19
  VectorGeoc3D 32-22
  VectorOffset3D 32-22
  vrf 32-3
  VR-Forces 74-3
Classified CID level 9-16
clearing
  default window layout 83-6
  file cache 64-6
  model instancing cache 6-13
  object console 16-27
  snapshot list 7-43
  view list 51-32
click to create 14-13
click to locate 14-13, 14-15
climb rate 26-9
clipboard 16-26
clipping plane 90-30
  adjusting when creating simulation objects 14-17
clock mode 3-12, 12-6
close air support 27-7, 30-5
  strafing 27-22
Close Car Door task 29-16
Close Cargo Door task 29-16
Close Door task 28-3
Close Window task 28-4
closing
  all windows 4-23
  application 4-23
  car door 29-10, 29-16
  cargo door 29-16
  Create Object dialog box 14-15
  door 28-3
  formation 21-5
  simulation 7-37
  simulation engine 4-23
  window 28-4

cloud
  cover 11-6, 11-9, 11-12, 11-17
    adding layer 11-20
    creating 11-17
    editing 11-20
    removing 11-21
    removing layer 11-20
  disabling 45-19
  layer 11-17, 40-14
    adding to cloud cover 11-20
    enabling and disabling 11-20
    parameters 11-21
    removing from cloud cover 11-20
  skybox cube map 45-15
  smoke 27-15
Cloud Cover Editor page 11-17
cockpit, enabling with keyword 82-17
cockpit display 53-6, 89-2, 90-10
  adding new 90-10
  element definition 88-2
  enabling or disabling 53-4
  installing 90-11
  model definition, creating 90-11
  updater 90-10
    attribute 90-10
Cockpit Display Settings page 53-5
cockpitOnly keyword 82-17
code
  country B-2
  laser 34-24
    assigning 9-18
coefficient probability 76-12, D-7
coefficients determinant 77-9, D-20
collapsing
  unit 24-2, 24-4
    to root 24-4
collateral damage D-31
  probability table 76-13, D-8
collision avoidance 13-11, 61-6
  configuring 78-4
  specifying for entity 34-11
Collision Avoidance Types dialog box 34-11
Collision Avoidance Types set data request 34-11
color
  2D icon 16-15
  2D label background 16-16
color (continued)
  altitude point
    changing 55-5
    deleting 55-4
    inserting 55-4
  ambient 45-4
  compass 51-5
  diffuse 45-4
  emitter volume, pulse rate 91-2
  entity label 16-9
  fog 11-6, 11-11
  force, changing 16-19, 67-33
  intervisibility, configuring 48-10
  partially identified simulation objects 9-16
  pattern 59-23
  radio communication line 43-28
  represents emitter volume frequency 91-2
  sensor contact line 43-16
  sensor field of view 34-39
  specular 45-4
  squawk indicator 43-28
  tactical graphic vertex 40-21
  tactical graphics, changing 40-21
  terrain, configuring 55-4
  unit bounding volume 24-10
Color by Force Type check box 16-15
Color dialog box 41-4
Color set data request 41-4
coloring
  draw call 6-23
  terrain 55-3
colorized model, changing 68-14
COLOUR attribute 59-23
COLPAT attribute 59-23
combat 22-3
  aggregate-level 22-9
  power 22-4
  unit 23-10
    graphics 43-10
combat engineering object 22-11
  as feature 65-10
  completion percentage, setting 35-12
  configuring 71-26
  creating 22-13
  creating new model 71-2
  partial completion 22-13
  rate of creation 22-13
  status 22-14

combat identification level 9-16
  CID 75-9
Combat parameters 71-24
combined mode, vrfLauncher 5-17
combining, key mapping 85-6
Come to Stop task 26-4
command
  Abandon Plan 37-44
  Delete Self 18-10
  Disembark 17-7
  Embark On 17-5
  global, adding to plan 37-31
  Issue Plan 37-36
  menu, configuring 84-2
  Restart Plan 37-43
  Send Text Message 37-34
command line
  loading scenario from 7-26
  loading terrain 5-7
  options 5-2
command-line, Simulation Object Editor 67-4
command-line option 5-1
  -- 5-2, 5-12, 5-16, 63-4
  -A 5-8, 5-13, 5-22, 63-4
  -a 5-7, 5-8, 5-9, 5-18
  --additionalSystemScriptsDirectory 5-9
  --alphaBits 5-2
  --antiAliasing 5-2, 6-16
  --appDataDir 5-2, 5-9
  --appIdRange 5-8, 5-9
  --appNumber 5-7, 5-8, 5-9
  --attributePduStrategy 5-14
  --autoConnect 5-2
  -B 5-9, 5-16
  B-- 5-16, B-11
  -b 90-36
  --backend 5-16
  -C 5-2, 5-16
  -c 5-10, 63-4
  --col 63-4
  --config 5-16
  --countryCodesMappingFile 5-10
  --daemon 5-10
  --dataDir 5-2, 5-10
  -debug 5-16, B-11
  --defaultObjectTimeout 5-8, 5-10
  --defaultSimulationModelSet 5-2
  --depthBuffer 5-2
  --destAddrString 5-8, 5-13
command-line option 5-1 (continued)
  --deviceAddress 5-8, 5-14
  --diGuyAnimationsFile 5-10
  --diGuyCharacerDataFile 5-10
  --dis 5-2
  --disableDynamicTerrain 5-2
  --disableKDTrees 5-4
  --disableOcclusionCulling 5-2
  --disableParallelTick 5-10
  --discoveredModelPreloadFileName 5-2, 6-25
  --disPort 5-9, 5-14
  --dispSetting 5-2
  --disVersion 5-8, 5-14, 5-22
  --doNotCheckPluginVersions 5-3
  --doNotLoadVrfPlugins 5-3, 5-10
  -E 5-3
  -e 5-3, 63-4
  --enableChannel 5-10
  --enableLogFileTimestamps 5-3
  --enableVSync 6-24
  --enableVSync 5-3
  --encryptImages 63-4
  --entDispSetting 5-3
  --entityDefsDir 5-3, 88-17
  --entTypeMap 5-3
  --envSetting 5-3
  --execname 5-8, 5-13
  --exerciseId 5-9, 5-15
  --extendLevelZero 63-4
  --extraArgs 5-3
  -F 5-3, 5-16
  F-- 5-16, B-11
  -f 5-7, 5-12, 5-20, 63-4
  --factoryRootDir 5-3
  --federateName 5-8, 5-13
  --federateType 5-8, 5-13
  --fedFileName 5-7, 5-12
  --file 63-4
  --fileNotifyLevel 5-10
  --fileTransporterReceivePort 5-3, 5-8
  --fomMapperInitData 5-7, 5-12
  --fomMapperLib 5-7, 5-12
  --fomModules 5-7, 5-12
  --forceTextShaping 5-3
  --forceVBO 5-3
  --fowPerspective 5-3
  --frameRate 5-3
  --frontend 5-16
  --frontEndPID 5-10

command-line option 5-1 (continued)
--fullScreen 5-3, 83-8
--fullyLoadTerrain 5-10
-G 5-3, 5-16, 5-18
-g 63-5
--generateFiles 63-5
--gui 5-4, 84-2
--guiArgs 5-16
-H 5-8, 5-9
-h 5-4, 5-10, 5-16, 63-5
--help 5-4, 63-5
--highestPriority 5-4
--hla13 5-4
--hla1516 5-4
--hla1516e 5-4
--hostAddressString 5-8, 5-9, 5-10
-I 5-4
-i 4-9, 5-8, 5-9, 5-10
--ignore_rest 5-2, 5-12, 63-4
--ignoreNonBaseTex 90-36
--ignoreZoomForSpeedtreeLod 5-7
-K 5-4
-L 5-4, 5-10
-l 5-4, 63-5
--layoutSettingsFile 5-4
--level 63-5
--loadObservers 5-4
--loadPlugin 5-4, 5-10, 5-19
--loadSimulationObjectGroup 5-4
--locale 5-3, 5-16
--localSettingsDesignator 5-8, 5-13
--logFileName 5-4, 5-11
-m 63-5
--mcastTtl 5-9, 5-14
--mergeEntityTypeMap 5-5
--mergeHierarchyEntityDef 5-5, 88-17
--mergeHierarchyTacticalGraphicsDef 5-5, 88-17
--mergeHierarchyUnitDef 5-5, 88-17
--mergeLeafEntityDef 5-5, 88-17
--mergeLeafTacticalGraphicsDef 5-5, 88-17
--mergeLeafUnitDef 5-5, 88-17
--mergeModelDefs 5-5, 87-12
--mergeTacticalGraphicsTypeMap 5-5
--mergeUnitTypeMap 5-5
--mimModule 5-12
--missing 63-5
--modelPreloadFileName 5-5, 6-25
--msdlSIDCMappingFile 5-11

command-line option 5-1 (continued)
--multicastAddresses 5-9, 5-14
-N 5-7, 5-12
-n 5-5, 5-11, 5-18, 67-5
--noAppDataEntityDefs 5-5, 88-17
--noAppDataEntityTypeMap 89-9
--noAppDataEntityTypeMap 5-6
--noAppDataModelDefs 5-6, 87-12
--noAppDataTacticalGraphicsDefs 5-6, 88-17
--noAppDataTacticalGraphics-TypeMap 5-6
--noAppDataUnitDefs 5-6, 88-17
--nonVrfObjectsUUIDScheme 5-6, 5-11
--noRtiCompilerCheck 5-13
--notifyLevel 5-5, 5-11, 67-5
--numCallbackThreads 5-11
-O 5-6
--outputLogFile 5-11
-P 5-9, 5-14, 5-21
-p 5-8, 5-13, 63-5
--plugin 5-6
--pseudoFullScreen 5-6, 83-8
--publishMftOnly 63-5
--pvd 5-6
-q 5-11
-r 5-11, 63-5, 90-37
--recvBufferSize 5-9, 5-14
--regenerateAll 63-4
--removeHighestLod 90-37
--reportedHostAddressString 5-6
--row 63-6
--rprFomRevision 5-8, 5-13
--rprFomVersion 5-8, 5-13
-S 5-7, 5-8, 5-13, 5-14, 5-22, 52-7
-s 5-11, 5-18, 63-5
--scaleLod 90-37
--scenarioFile 5-7
--searchDir 63-5
--sendBufferSize 5-9, 5-14
--sessionId 5-8, 5-9
--setMainThreadToHighPriority 5-11
--settingsFile 5-10
--showConsole 5-2
--simArgs 5-16

command-line option 5-1 (continued)
Simulation Object Editor
-- 67-4
--appDataDir 67-4
--dataDir 67-4
--doNotCheckPluginVersions 67-4
--doNotLoadVrfPlugins 67-4
-G 67-4
-h 67-4
--help 67-4
--ignore_rest 67-4
--launcherLocation 67-4
--layoutSettingsFile 67-5
--loadPlugin 67-5
--locale 67-4
--settingsDirectory 67-5
--showConsole 67-5
--smsFile 67-5
-v 67-5
--version 67-5
--visualTerrain 67-5
--simulationModelSet 5-11
--siteId 5-8, 5-9
--startMinimized 5-12
--stealth 5-7
--stencilBits 5-7
--subDir 63-5
--subnetMask 5-9, 5-14
--suppressDataAvailableException 5-12
--suppressHatIntersectOnAttach 5-7, 52-7
--suppressSelfReflect 5-14
--synchronizeOcean 5-6
-T 5-12
-t 63-5
--targetFrameRate 5-12, 6-22
--textureSize 63-6
--threads 63-5
--timeManagement 4-20, 5-13, 5-21
-u 63-5
--unbatchedRendering 5-4
--unplug 5-7
--useAbsoluteTimeStamps 5-7, 5-12
--useAdvisories 5-8, 5-12
--useAsyncIO 5-9, 5-14, 5-22
--useFileCommChannel 5-7
--useGeographicDdm 5-8
--useIpv6 5-9, 5-15
--userDataDir 5-7, 5-12
--useReverseZBuffer 5-7
command-line option 5-1 (continued)
-V 90-37
-v 5-7, 5-12, 5-16, 63-5
--verbose 90-37
--version 5-7, 5-12, 5-16, 63-5
--vtFile 63-5
-w 63-6
--waitQueue 5-12
-x 5-8, 5-9, 5-13, 5-15, 5-19, 63-6
DIS 5-22
-Z 5-7
-z 5-7
comma-separated values file 10-8
comment
@dis 59-28
MTL parameter A-3
comm-model-name parameter 79-4
commModelParams.mtl file 79-4
communication
component 13-7
line 43-25
line-of-sight 43-27
communication model 79-4
custom 13-6
simulation object 13-6
communications, jamming 22-18
company, pre-configured 23-5
comparison operator 36-12
compass
color 51-5
configuring 51-5
keyword 82-17
location 51-5
model definition 51-5
observer 51-4
compiling, incremental 6-14
component 74-8
actuator 13-8
architecture 13-7
attaching to remote entities 7-5
attaching to remote entity 81-2
communication among 13-7
controller 13-7, 13-8
descriptor, elements of 74-13
emitter 34-17
entity 74-13
sensor 13-7
simulation object 74-13
systems 74-17

component 74-8 (continued)
tuning 6-15
performance 6-15
component attachment table 81-2
Component Attachment Table parameter 7-6
Component Manager C-5
Component Visualization 64-12
Component-Attachment parameter 12-5
Component-Attachment-Table parameter 12-5
componentAttachmentTable.mtl 81-2
component-descriptor-type 74-13
component-manager parameter C-5, C-7
component-type 74-13
composing, terrain 54-2
compressing, model and image files 90-35
compression, texture 64-6
computed route, creating 40-9
Concealed dialog box 34-12
Concealed set data request 34-12
concealment, setting 34-12
concurrent task 24-1, 25-4, 31-24, 31-31
condition
considerations for using 36-9
Detect Entity 36-9
Engineering Object Breached 36-9
Entity Altitude 36-9
Entity Destroyed 36-9
Entity Di-Guy Animation Check 36-10
Entity Di-Guy Appearance Check 36-10
Entity Embarked 36-10
Entity Has Target 36-10
Entity In Area 36-11, 40-5
Entity Left of Line 36-11
Entity Under Fire 36-11
If, creating 37-5
Lifeform Surrendered 36-11
Missile Approach Warning 36-11
name 37-24
name condition, pattern 37-23
pattern 37-25
editing CSV file 37-25
Random 36-12
Receive Text Message 36-12
Resource 36-12
Scenario Event 36-12
Simulation Date/Time 36-13
Simulation Object Independently Tasked 36-13
Simulation Time 36-13
condition (continued)
Tactical Graphic Active 36-13, 40-24, 41-2
While 36-8
conditional expression dialog box 37-13
conditional statement 36-3
building 37-13
filtering simulation object list 25-9
trigger 36-6
*condition-entity-types.csv*file 37-25
cone
gimbaled sensor 53-9
sensor, color 34-39
configName parameter 90-22, 90-24
configuration
display engine 82-2
loading 82-5
saving 82-5
entity display 5-3
environment 5-3
file
condition-entity-types.csv 37-25
RTI 2-10
joystick, favorite 80-15
plug-in, deleting 4-38
simulation model set 7-9
configuration file
character_animations_table.mtl 73-3
commModelParams.mtl 79-4
feature 65-6
for object parameter database 67-6
navigation area 61-19
sysdef 74-17
vrfSim.mtl B-2
vrfSimSettings.xml 6-19
configuring
2D icon 90-3
aggregate-level entity 71-2
back-end
console B-3
heartbeat B-6
ballistic missile movement 69-25
batch file B-2
combat engineering objects 71-26
compass 51-5
condition pattern enumerations 37-25
context-sensitive menu 84-7
counter measures 9-26
Create Object dialog box 14-15
damage-by-power 77-3, D-14

configuring (continued)
depth of field 53-15
destination address B-8
detection tables 9-16
dialog box
information 84-5
pages 84-5
DI-Guy B-3, B-7
character and appearance 69-11, 73-3
direct fire weapon 76-8, D-2
DIS B-7, B-8
display 5-2
drainInput B-4
for DIS B-8
for HLA B-7
earth file 59-3
embarkation 72-2
slot 72-5, 72-11
embedded entity 17-9
emitter 79-2
emitter volume 91-2
segments 91-4
exercise ID B-8
feature avoidance 78-5
feature data 65-2
FED file name B-7
federation execution B-7
name B-7
fixed weapon system 76-19, D-11
FOM Mapper B-7
formations 78-2
frequency of data sends B-3
GDB soil type 55-19
grid lines 55-7
ground clamping 54-11
GUI elements 84-2
heartbeat threshold B-7
HLA B-7
hostility 9-13
IED 34-14
ingress and egress points 72-11
instrument panel 53-7
interest management 6-11
intervisibility objects 48-10
joystick 80-4
keyboard entity control 80-4
LOD 64-7
map datum B-6
menu 84-2
model definition 87-2
configuring (continued)
movement file 69-29
multicast address B-7
multichannel display 82-18
mutually exclusive tasks 25-33
object avoidance 78-5
obstruction avoidance 78-4
ocean planar reflection 45-10
orientation clamping 54-11
path, to OPD file B-3
port number B-8
power of detonation 77-4, D-15
projection resize policy 82-11
props 54-4
pulse rate 91-2
radar converage 9-23
radios 79-4
range rings 43-19
receive port B-2
right-click menu 84-7
scenario file to use B-5
schema 86-2
sensor contact line 43-16
sensor effects 47-4
sensors 75-2
session 4-17
ID B-5
shadows 45-23
ship wakes 90-18
signature propagator 75-11
simulation management message B-5
site ID B-6
slot exclusion 72-9
start mode B-6
stereo
anaglyphic 82-22
polarized 82-23
system for MAK RTI 2-10
TDL message 9-11
terrain color settings 55-4
terrain database
path B-3
to load B-6
terrain profile 56-3
time management 4-19
time stamp order B-7
toolbar 84-9
tracers 76-11, D-6
track history 43-6

configuring (continued)
unit, ammunition, weapons, equipment 71-4
visual quality slider 6-20
confirmation prompt
plan, enabling or disabling 25-5
task, enabling or disabling 25-5
connecting, to network automatically 5-2
connection
configuring in system definition file 74-18
information 4-31
sensor 75-3
simulation object group 68-29
Connection Information dialog box 4-31
console
configuring for back-end B-3
messages
sending 37-35
sent to 5-18
object 16-23
setting notification level 34-28
constrained, time 4-19
constraint 52-8
view 51-27
Construct Abatis dialog box 30-12
Construct Abatis task 30-12
Construct Anti-Tank Ditch dialog box 30-12
Construct Anti-Tank Ditch task 30-12
Construct Barbed Wire dialog box 30-13
Construct Barbed Wire task 30-13
Construct Barricade dialog box 30-13
Construct Barricade task 30-13
Construct Berm dialog box 30-14
Construct Berm task 30-14
Construct Booby Trap dialog box 30-14
Construct Booby Trap task 30-14
Construct Bridge dialog box 30-15
Construct Bridge task 30-15
Construct Dry Gap dialog box 30-16
Construct Dry Gap task 30-16
Construct Fortified Area dialog box 30-17
Construct Fortified Area task 30-17
Construct Fortified Line dialog box 30-18
Construct Fortified Line task 30-18
Construct Minefield dialog box 30-18
Construct Minefield task 30-18
Construct Strong Point dialog box 30-19
Construct Strong Point task 30-19
constructing
abatis 30-12
barbed wire 30-13
barricade 30-13
berm 30-14
fortified line 30-18
strong point 30-19
terrain 57-4
contamination
agent 22-9
biological 30-7
chemical 30-11
nuclear 30-29
nuclear, biological, chemical 22-9, 22-10
type, setting 35-6
Contamination dialog box 35-6
Contamination set data request 35-6
context-sensitive menu 83-10
configuring 84-2, 84-7
contrail 43-3
contrast 47-4
control, remote 1-15
control object 38-2, C-7
concepts 38-2
copying 14-28
count of 16-28
creating 14-3
editing 40-17
list of for scenario 12-10
name 38-2
parameter-type 74-39
pasting 14-28
pedestrian path 19-10, 38-2
selecting 15-4, 15-5
spot report 9-8
control panel, gimbaled sensor 53-10
controller 13-7
component 13-7, 13-8
detonation 76-19, D-11
dynamic-terrain-munition-damage 77-16
game 80-2
indirect fire 76-17, D-9
indirect fire weapon 76-17, D-9
joystick 80-4
lasing 76-16
launcher 76-16
making decisions and performing tasks 13-7
missile 76-16, 76-17

controller 13-7 (continued)
target selection 76-9, 76-15, 76-16, D-4
controlling
messages 5-18
remote 51-35
conventions, manual lxii
convoy 26-5
behavior 25-10
Convoy Along task 25-10, 26-5
Convoy To task 25-10, 26-5
coordinate
observer 51-3
point 55-16
UV 57-8
coordinate system 1-12
3D 51-7
3D Cartesian 32-20
Cartesian 32-20, 32-22
changing unit of 83-12
geocentric 54-6
Lambert Conformal Conic 51-7
LCC 51-7
supported 54-6
UTM 51-7
copying
formation 70-12
model definition 87-7
objects 14-28
performance and 14-28
plan statements 37-42
schema 86-4
scripted task 31-41
spawn pattern 18-14
corridor 40-5
count
object 16-28
simulation object 16-28
counter measure 27-15
ammo select table 9-26
launching 9-26
Counter Measures Auto Launch, set data request 9-26
Counter Measures Auto Launch set data request 34-12
Countries list 68-10
Country 68-9
country, marking 68-23
country code B-2
Country list 68-9
countryCodesMappingFile parameter B-2, B-4
cover, cloud 11-17
CPU time, giving up 6-22
crawling 34-31
Create Anti-Tank Ditch dialog box 30-12
Create Barbed Wire dialog box 30-13
Create Barricade dialog box 30-13
Create Berm dialog box 30-14
Create Booby Trap dialog box 30-14
Create Bridge dialog box 30-15
Create Dry Gap dialog box 30-16
Create Fortified Area dialog box 30-17
Create Fortified Line dialog box 30-18
Create global command 37-35
Create Indirect Fire dialog box 10-3
Create Intelligence Object dialog box 8-8
Create menu 14-3
assigning or removing simulation object 68-8
selecting object 14-7
Create Minefield (ADAM-RAAM) dialog box 30-22
Create Minefield (Volcano) dialog box 30-18
Create Multiple Entities, dialog box 19-7
Create New Entity dialog box 68-26
Create New Entity Type Model Mapping dialog box 89-4
Create Object
dialog box 14-3
Attach Mouse to Object 14-21
configuring 14-15
tab 14-13
Create Object dialog box 14-15
Create Pedestrians dialog box 19-3
Create Pedestrians task 28-4
Create Route dialog box 14-19
Create Scenario Event dialog box 8-3
Create Spawn Pattern dialog box 18-11, 18-14
Create Strong Point dialog box 30-19
Create_Global_Dynamic_Terrain parameter 12-7
Create_Global_Environment parameter 12-7
Created Object variable 36-14
create-script-id parameter 31-27
creating 14-11
aggregate-level entity 71-2
area for multiple entities 19-8
assembly 71-11
batch file 7-46
behavior set 25-19

creating 14-11 (continued)
booby trap 30-14
civilians 19-2
cloud cover 11-17
combat engineering object 22-13
computed route 40-9
control object 14-3
crowd 19-2
crowd unit 19-3
crowds 19-2
cultural feature 14-33, 69-3
demonstration recordings 37-37
element definition 88-7
embedded entity 29-19
entity 14-11
adjusting clipping plane 14-17
fixed-wing 25-23
from existing 68-27
in buildings 14-17
new 68-26
fortified area 30-17
fortified line 30-18
freehand line 40-3
global plan 37-27
If block 37-5
indirect fire event 10-3
inset view 53-3
joint engagement zone altitude limited 40-5
joystick configuration 80-5
key map 85-7
MEDF and MEIF files 90-35
minefield 30-18
missile engagement zone altitude limited 40-5
model, guidelines for 90-34
model definition 87-2, 87-5
cockpit display 90-11
navigation area 61-8
new scenario 7-3
object 14-3, 14-4, 14-7, 14-13
in plan 37-35
mouse with 14-21
setting properties 14-15
objects 14-7
observer mode 50-4
overlay 39-3
pedestrian area 19-2
plans 37-3
prop 14-11, 14-35, 90-13
creating 14-11 (continued)
reactive task 31-33
scenario event 8-3
schema 86-3
script 31-6
scripted set 31-31
selection group 15-8
selection group from menu 15-9
simulation model set 67-23
simulation object 14-3, 14-4, 14-7, 14-13
from existing 68-27
group 68-28
multiple 19-5
simulation objects, multiple 19-2
sink points 18-5
snapshot 7-42
spawn points 18-3
strong point 30-19
system definition file 74-17
tactical graphics 14-11, 40-2
terrain 57-2
text object 40-5
unit 23-2, 70-4
aggregation state 23-6
entity 23-3
user-defined formation 78-4
window layout 83-4
crepuscular ray 45-13
crew
air, air base 35-13
crosswalk 19-10
crouching 34-31
crowd
adding members 28-4
creating 19-2
dispersing 28-8
gathering around object 28-5
gathering at location 28-5
protesting 28-14
unit, creating 19-3
wander 28-16
Crowd Along Line dialog box 28-4
Crowd Along Line task 28-4
Crowd Around Location task 28-5
Crowd Around Object task 28-5
Crowd in Front of Entity dialog box 28-6
Crowd in Front of Entity task 28-6
cruise missile, firing 27-10
CSV, file 26-4

CSV file 10-8, 69-24, 69-27
  configuring for ballistic missile or animated movement 69-29
  importing objects from 7-18
cube map, recalculating 45-15
culling
  face front 46-9
  occlusion 5-2
cultural feature 14-33, 69-3
  aligning to gravity 69-10
  creating 14-3, 14-4, 14-7, 14-13
  embedding in terrain 14-34, 69-9
  ground clamping 69-10
  heading, setting 14-20
  information 16-3
cultural-feature-param parameter 74-12, C-8
cultural-feature-state-repository 74-12
Current Speed dialog box 34-13
Current Speed set data request 34-13
cursor 14-11, 14-12
curvature of the earth, display mode 43-27
custom, plan 18-2, 18-10, 18-13
cut-in site 59-15

D

-d command-line option, vrfSim 5-10
daemon
  running vrfSim as B-2
  starting back-end as 5-10
daemon parameter B-2
damage
  actuator 77-8, D-19, D-23
  calculating 77-2, D-13
  collateral D-31
  displaying 68-21
  dynamic terrain 77-16
  file 74-40
  model 69-15, 69-16
    specifying 69-17
  probability D-18
  probability table 76-13, D-8
damage file 76-8, 77-2, D-2, D-13, D-18, D-29, D-31
  damage-by-power 77-8, D-18
damage system 76-8, 76-13, 77-2, D-2, D-8, D-13, D-28, D-31
  migrating to new release 67-21
damage table D-22
damage-by-ammo-type 76-13, 77-2, D-8, D-13, D-22
  example D-30
damage-by-munition-power block 77-8, D-18
damage-by-power 76-13, 77-2, 77-11, D-8, D-13, D-21
  adding new 77-10, D-20
  configuring 77-3, D-14
  damage file 77-8, D-18
  terrain 77-16
damage-by-power-entry 77-8, D-18
damage-model block 77-8, D-18, D-22, D-31
dashed line 40-23
data
  caching terrain 64-3
  configuring frequency of sends B-3
  directory 2-4
    specifying location of 5-2, 67-4
  elevation 54-3
  feature, configuring 65-2
  input 74-13
  path 61-2
  port 13-7
  S-57 57-28
  streamed, processing 59-8
  synchronizing 3-16
data directory
  back-end 5-10
    user 5-12
data distribution management 6-10, 6-11
  See also DDM
data type, state property 74-28
database 54-6
  CDB 60-2
  large 54-7
  loading 54-7
  object parameter 74-2
  paging 54-8
  terrain 1-12
    loading 4-21
    provided 55-21
dataset
  CDB, geospecific 60-6
  elevation, CDB 60-3
  extended imagery, CDB 60-15
  geotypical, CDB 60-13
  imagery, CDB 60-5
  virtual and tiled 63-4

Datasets.xml 60-17
date 11-3
  setting 11-3, 11-4
  simulation 7-12
Date and Time page 7-12, 7-13, 11-4, 11-5
datumShiftX parameter B-3
datumShiftY parameter B-3
datumShiftZ parameter B-3
daylight, controlling light 45-7
daylight-illumination parameter 75-15
daymark, buoy 59-29
daymark_attr_map.csv 59-29
daymark_layer_list.csv 59-29
daymark.dtxc 59-30
day-night-transition-time is parameter 75-15
DDM 6-10, 6-11
  region size 6-11
DDS texture 64-6, 64-17
  flipping 57-5, 90-29
Deactivate Jammer task 30-19
deactivating
  jammer 30-19
  spawn points 18-4
dead reckoning, changing for fast forward 7-35
dead-reckoning 6-21, B-4
debug mode, running from vrfLauncher 5-17
debug overlay 6-27
debugging
  earth file 64-7
  Navigation Lab B-3
  shaders 64-12
  terrain 64-16
decal effects 43-3, 89-2
  enabling and disabling 43-3
  models 43-4
decal image attribute 57-7
decay rate for spot reports 9-7
decision making 13-7
decreasing speed of navigation 51-27
default
  formation 70-10
  joystick configuration 80-15
  name 14-13
  plan 18-2, 18-10, 18-13
  Set commands 33-1
  settings, restoring 3-13
  simulation model set 5-2
    configuration 7-9
default (continued)
  simulation model set configuration 7-11
  spawn pattern 18-10
  value, parameter 12-10
  variable binding value 74-23
  view 51-33
default overlay
  adding 68-9
  specifying 68-8
default_guiConfiguration.gui 74-29
default_guiConfiguration.gui 84-2, 84-3
default_guiSimpleScenarioPlayer.gui 84-2
default_ModelPreload.mpsx, file 6-24
DefaultEllipsoidArc, model definition 91-2
defaultHeartbeatThreshold parameter B-7
defaultObjectTimeout parameter B-8
DefaultObserverView parameter 12-7
defaultParameterDatabasePath parameter 12-8, B-3
defaultSpawnTemplates.mtl 18-10
default-task-rules.tsk 25-33, 31-24
defaultTerrainDatabasePath parameter 12-8, B-3
defense parameter, editing 71-23
defensive
  measure 27-15
  movement 30-28
definition
  element 88-2
  model, buoy 59-22, 59-23
degrees of freedom 92-8
delay parameter 77-17
Delete Checkpoints dialog box 7-42
Delete Object global command 37-35, 37-36
Delete Self, command 18-10
Delete Self global command 37-36
deleting
  aircraft assignment 35-6
  altitude point 55-4
  assembly 71-13
  channel 82-5
  checkpoints 7-42
  element definition 88-14
  entity 14-32
    from simulation model set 68-27
  environment condition 11-8
  extended label 16-19
  intervisibility object 48-16
  key map 85-7
  key mapping 85-5

deleting (continued)
messages from object console 16-27
model definition 87-6
model mapping 89-7
object 68-27
in plan 37-35
object from OOB 20-6
object from selection group 15-10
observer 50-9
mode 50-5
overlay 39-5
object 40-17
parameter
model definition 87-10
schema 86-6
plan statement 37-12
plug-in
configuration 4-38
XML file 4-39
scenario event 8-15
schema 86-4
script 31-41
selection group 15-10
simulation object 14-32
in plan 37-36
snapshots 7-43
spawn pattern 18-16
tag 67-27, 67-29
terrain, cached 64-6
trigger 37-10
unit 23-13
vertex 40-20
view 51-32
window 82-5
window layout 83-7
Deliberate-Attack, posture 22-6
Deliberate-Defense, posture 22-6
demonstration
creating 37-37
mode 83-8
Deploy Refueling Boom task 29-16
Deploy Sonobuoy task 29-22
Deploy Sonobuoys Along Route task 29-22
deploying
embedded entity 29-19
with task 29-21
sonobuoy 29-22
depth
bits 5-2
buffer 6-16
depth (continued)
focal 53-13
moving to 26-14
ocean 57-28
sonar 34-40
depth charge 27-4, 27-9
depth of field 53-13
configuring 53-15
enabling and disabling 53-13
descent rate 26-9
description
script 31-13
spawn pattern 18-11
deselecting
object 15-7
simulation object 15-7
DESIGNATOR, attribute 59-25
designator 13-5
buoy 59-25
laser beam 9-18
Designer 33-2
destAddrString parameter B-8
destination address 5-8, 5-13, B-8
Destroy Bridge dialog box 30-20
Destroy Bridge task 30-20
Destroy Ditch dialog box 30-20
Destroy Ditch task 30-16, 30-20
Destroy Explosive dialog box 30-20
Destroy Explosive task 30-20
Destroy Fortification dialog box 30-21
Destroy Fortification task 30-21
Destroy Obstacle dialog box 30-21
Destroy Obstacle task 30-21
destroyed, setting simulation object to 34-13
Destroyed set data request 34-13
destroyedsymbol attribute 90-8
destroyFedExec parameter B-7
destroying
booby trap 30-20
bridge 30-20
ditch 30-20
federation execution B-7
fortification 30-21
obstacle 30-21
unexploded ordnance 30-20
destroy-script-id parameter 31-27
detaching, observer from object 52-6
Detect Entity condition 36-9
Detected CID level 9-16

detecting, targets 9-16
detecting target 76-9, D-4
detection table 9-16, 75-16
detection types, sensor 75-11
detection-types parameter 76-9, D-4
determinant
  coefficients 77-9, D-20
  direct 77-6, D-17
  equivalent-tnt-mass 77-6, D-17
  power 77-4, D-15
  range-coefficients 77-5, D-16
  range-list 77-5, D-16
determinant block 77-8, D-19
detonating, IED and car bomb 34-14
detonation
  configuring power of 77-4, D-15
  effects 43-8, 89-2
  event
    mapping sound effects to 49-4
    sound effects 49-2
  line 43-9
  proximity 76-19, D-11
detonation controller 76-19, D-11
Detonation Fuse Type dialog box 34-14
Detonation Fuse Type set data request 34-14
detonation interaction 76-17
detonation result 77-18
detonation-munition parameter 77-17
detonation-on-destroy parameter 77-17
detonationParams.mtl 76-18, 77-3, 77-8, 77-10, 77-18, D-10, D-14, D-18, D-20, D-27, D-30
detonation-power-entry 77-3, D-14
Developer Debug Overlay 60-12
developer debug overlay 6-27, 60-9
Device address 5-8, 5-14
deviceAddress parameter B-8
dialog box 26-11
  26-11
  Active Sonar Mode 34-4
  Activity 35-3
  Add Cloud Layer 11-18
  Add Conditional Expression 36-5
  Add Group of Entities 19-9
  Add Jamming Target 34-4
  Add Model Definition 87-5
  Add Terrain Patch 57-4, 66-8
  Add to Order of Battle 20-4
  Advanced Visual Model Editing 68-22
dialog box 26-11 (continued)
  Aggregate As 70-3
  AI Enabled 34-5
  Aiming 34-6
  Air-to-Air Attack 30-4
  Air-to-Ground Air Base Attack 30-4
  Air-to-Ground Attack 30-5
  Allow Crossing 41-3
  Altitude 34-7
  Ammunition 35-3
  Ammunition Pacing/Tracking 35-4
  Animated Movement 26-4, 69-29
  Appearance 34-8
  Application Settings 7-28, 23-11
    File Caching Settings page 6-13, 64-2, 64-6
    General Application Settings page 4-14, 4-24
    Key Mapping Editor page 85-3, 85-7
    Mouse Mappings page 51-14
    Performance Settings page 6-19, 6-22
    Script Settings page 6-6, 31-22, 31-44
    Spot Report Settings page 9-3, 9-4, 9-5, 9-7, 9-8, 9-9, 34-41
  Armed 34-9
  Arrowhead Style 41-3
  Assign Aircraft 35-5, 35-6
  Assume Unknown is Hostile 34-9
  Attack By Fire 30-5
  Attack with Anti-Air Missile 30-6
  Attack with Anti-Ship Missile 30-6
  Attack with Land-Attack Missile 30-6
  Attack with Torpedo 30-7
  Attention Grabbing 34-9
  Audio Settings 49-2, 49-3
  Auto Launch 34-12
  Automatically Close Doors 34-10
  Ballistic Missiles 69-25, 69-29
  Biological Attack 30-7
  Bomb Location 30-8
  Breach Obstacles 30-9
  capabilities 34-10
  Change Hostility 9-15
  Change MOPP Level 30-10
  Change Posture 30-11
  Chemical Attack 30-11
  Collision Avoidance Types 34-11
  Color 41-4
  Concealed 34-12
  conditional statement 37-13, 37-24

dialog box 26-11 (continued)
configuring 84-5
Connection Information 4-31
Construct Abatis 30-12
Construct Anti-Tank Ditch 30-12
Construct Barbed Wire 30-13
Construct Barricade 30-13
Construct Berm 30-14
Construct Booby Trap 30-14
Construct Bridge 30-15
Construct Dry Gap 30-16
Construct Fortified Area 30-17
Construct Fortified Line 30-18
Construct Minefield 30-18
Construct Strong Point 30-19
Contamination 35-6
Create Anti-Tank Ditch 30-12
Create Barbed Wire 30-13
Create Barricade 30-13
Create Berm 30-14
Create Booby Trap 30-14
Create Bridge 30-15
Create Dry Gap 30-16
Create Fortified Area 30-17
Create Fortified Line 30-18
Create Indirect Fire 10-3
Create Intelligence Object 8-8
Create Minefield (ADAM-RAAM) 30-22
Create Minefield (Volcano) 30-18
Create Multiple Entities 19-7
Create New Entity 68-26
Create New Entity Type Model Mapping 89-4
Create Object 14-3
configuring 14-15
Create Pedestrians 19-3
Create Route 14-19
Create Scenario Event 8-3
Create Spawn Pattern 18-11, 18-14
Create Strong Point 30-19
Crowd Along Line 28-4
Crowd in Front of Entity 28-6
Current Speed 34-13
Delete Checkpoints 7-42
Destroy Bridge 30-20
Destroy Ditch 30-20
Destroy Explosive 30-20
Destroy Fortification 30-21
Destroy Obstacle 30-21
Detonation Fuse Type 34-14

dialog box 26-11 (continued)
DI-Guy Characteristics 34-15
DI-Guy Hand Item 34-15
Display Settings 3-13, 9-23, 11-17, 44-8, A-2
Cockpit Display Settings page 53-5
Display Units Settings page 83-14
Entity Display Settings page 16-12, 16-13, 16-19, 16-20, 16-27, 25-5, 43-13, 43-24, 44-3, 44-4, 46-6, 46-7, 46-9, 50-12, 54-10, 54-11
Grid Lines Settings page 55-7
Indirect Fire Settings page 10-7
Interest Management Settings page 6-11
Intervisibility Settings page 48-6, 48-8, 48-10, 48-13, 48-14
Lighting Render Settings page 45-4, 45-5
Loader Settings page 64-17
Observer Settings page 11-23, 16-22, 43-3, 43-5, 43-8, 43-14, 43-18, 43-23, 44-7, 44-8, 45-7, 46-3, 50-3, 50-4, 50-5, 50-6, 50-7, 50-9, 51-5, 51-14, 51-25, 51-26, 51-29, 53-9, 53-14, 55-6, 55-11
Ocean Settings page 11-26, 11-27, 45-9, 45-10, 45-15, 46-4, 46-6, 57-18, 57-19
Radio Communications Settings page 43-26, 43-27, 43-28
Range Rings Settings tab 43-19
Render Settings page 6-9, 6-10, 6-18, 6-23, 45-14, 53-15, 55-9, 60-9, 60-12, 64-12, 64-13, 64-15, 64-19, 82-12
Sensor Settings page 47-4, 47-5
Shadow Settings page 45-20, 45-23
SpeedTree Settings page 45-22, 90-30
Symbol Decoration Settings page 16-14, 16-15
Tactical Graphics Display Settings page 38-7, 38-8, 40-21, 42-2
Terrain Profile Settings page 48-15
Track History Settings page 43-6
Unit Display Settings page 24-5, 24-9, 24-10, 43-10, 43-23

dialog box 26-11 (continued)
Duplicate Model Definition 87-7
Edit Activities 71-16
Edit Behavior Sets 25-19, 25-20
Edit Cloud Layer 11-21
Edit Spawn Pattern 18-15
Edit Trigger Condition 37-7
Embark On 17-5, 29-9
Embarked 34-16
Emitter 34-18
Engagement Results 35-7
Engagement Zones 34-18
Environment Conditions 7-12, 7-13, 11-4, 11-5, 11-13
Environmental Object Information 16-3
Equipment Pacing/Tracking 35-8
Escort Air Unit 30-22
Execute Close Air Support 27-7
Experimental Features 11-28, 51-21, 51-22, 64-15
Export Object Information 67-26
Facilities 35-9
Fast Rope Insertion 26-6
Fill Style 41-4
Find Cover 28-9
Find Firing Position 28-10
Fire Cruise Missile 27-10
Fixed Wing Land 26-7
Fixed Wing Takeoff 26-7
Flee From Entities 28-11
Follow Entity 26-10
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant 35-9
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant Pacing/Tracking 35-10
Force 34-18
Formation 34-19
Has Drag Chute 35-10
Heading 34-20
Health 35-11
If/Else Expression 37-5
IFF 34-21
Import Object Information 67-26
Improve Booby Trap 30-23
Improve Breach 30-23
Improve Bridge 30-23
Improve Ditch 30-24
Improve Fortification 30-24

dialog box 26-11 (continued)
Improve Obstacle 30-24
Indirect Fire 30-25
Information 16-3, 16-21
displaying state properties 74-29
information, configuring 84-5
Initial Scenario Information 7-3, 7-11
Invulnerable 34-22
Jammer State 34-22
Jamming Mode 34-23
Land at Air Base 30-25
Lase Autonomous 34-24
Laser Code 34-24
Laser Power and Wavelength 34-25
Launch Flight 30-27
Launch QRA Aircraft 30-27
License Setup 2-7
Line Width 41-4
Load Batch File 7-48
Load Overlays 39-6
Loadout 35-11
Location 34-26
locking 84-10
Look At 34-26
Manage Reactive Task 25-16
Manage Reactive Tasks 19-10
Markings 34-27
MOPP Level 35-12
Morale 35-12
Move Along Route 26-12
Move Between Cover to Location 28-12
Move Into Formation 26-13
Move to Altitude 26-14
Move to Depth 26-14
Move to Location 26-15
Move to Waypoint 26-18
Munition Target Settings 10-3, 10-6, 10-9, 10-11
Navigation Enabled 34-27
Navigation Preferences 34-28
New Scenario 7-44, 12-3
Notify Level 34-28
Nuclear Attack 30-29
Object Visible to Sensor At 8-9
Open Car Door 29-17
Open Model Set 67-7, 67-22
Orbit Object 26-21
Ordered Speed 34-30
osgEarth Profile 64-7
Overlay Layers 68-9

dialog box 26-11 (continued)
pages, configuring 84-2
Patrol Between 26-23
Patrol Route 26-23
Pattern Hold (Location) 26-24
Pattern Hold (Waypoint) 26-24
Pen Style 41-5
Percent Complete 35-12
Performance Settings 6-19
configuring slider 6-20
Personnel 35-13
Place IED 27-18
Plug-ins 4-33
Posture 34-31, 35-13
Preferred Targets 35-14
Preferred Trajectory Height 34-32
Prepare Aircraft for QRA Status 30-30
Protest Along Line 28-14
Protest Around Location 28-14
Radar Mode 34-32
Refuel from Tanker 30-30
Refueling Mission 30-31
Register Trigger 37-9
Remove Jamming Target 34-33
Resources 34-34
Resources Pacing/Tracking 35-14
Resupply Mode 35-15
Role 35-15
Rotary Wing Land 26-26
Rules of Engagement 34-35
Sail Heading 26-28
Save Scenario 7-22
Scenario Event 8-12
Scenario Properties 7-29
Scenario Snapshots 7-36, 7-40, 7-42, 7-43
Scenario Startup 4-6, 4-8
Sector of Responsibility 34-37
Sector Search Operation 29-20
Send NBC Report 30-31
Send Obstacle Report 30-32
Sensor Enabled 34-38
Sensor Field of View Color 34-39
Simulation Connections Configuration 4-33
Simulation Object Type Mappings 89-2, 89-7
Sonar Depth 34-40
Spot Reports 34-41
Stow Hand Item 34-42
dialog box 26-11 (continued)
Strafe Ground Target 27-23
Superior 34-42
Supplying 35-16
Surrendered 34-42
Synchronize Laser Code 34-43
Target 34-43
TDL Messaging 34-44
Terrain Settings
Add Props page 57-22
Earth Layers page 55-10
Edit Existing Props page 57-24, 57-25, 57-26, 66-11
Extract Ocean page 57-15
Extract Props page 57-21, 66-9
Raster Maps page 57-6, 57-10
SpeedTree Randomization page 90-32
Terrain Contents page 57-4, 57-13, 57-16, 57-17, 57-28, 58-3, 64-8, 66-8
Visible Surfaces page 55-19
Throw Grenade at Location 27-24
Throw Grenade at Target 27-24
Tracer Use 34-44
Turn to Heading 26-29
Unit State 34-45
User Task 29-24
Visual Markings 68-23
Visual Model Editors 87-2, 89-7
Wait Duration 29-15
Wait Elapsed 29-15
Wait for Escort 30-33
Wait in Queue 28-15
Wait Until Expression 37-6
Weapons Pacing/Tracking 35-17
Weather 11-15
While Expression 37-6
Window Layout Manager 83-4
Zig Zag to Heading 26-29
diffuse, color 45-4
digging
anti-tank, ditch 30-12
DI-Guy
action table 73-4
adding new characters 73-3
animation
adding 73-3
testing for 36-10
appearance, testing for 36-10
character data file 5-10

DI-Guy (continued)
choosing animation 73-5
configuring B-3
animations and appearances 5-10
character and appearance 69-11, 73-3
use of B-7
displaying in vehicle 44-8
hand item, hiding 34-42
instancing 6-26
integration with VR-Forces 28-7
license 1-17
model definition 87-5
motion file 73-4
performance 6-26
DI-Guy Animation task 28-7
DI-Guy Characteristics dialog box 34-15
DI-Guy Characteristics set data request 34-15
DI-Guy Gesture task 28-7
DI-Guy Hand Item dialog box 34-15
DI-Guy Hand Item set data request 34-15
DI-Guy Settings page 6-26
diGuyAnimationsFile parameter B-3
dipping, sonar 29-23
direct determinant 77-6, D-17
direct fire weapon, configuring 76-8, D-2
direct fire weapon system
damage-by-ammo-type, example D-30
damage-by-power, example D-25
direction
line, changing 40-21
Lua 32-19
tidal current 11-22, 11-25
wind 11-9
directory
appData 5-9
back-end data 5-10
back-end user data 5-12
data 2-4
installed 2-4
DirectX 64-17
DIS
command-line options 5-21
configuring B-7, B-8
encoding information in OpenFlight format 92-2
Enumeration Document 3-8
exercise ID 5-9, 5-15
heartbeat, threshold B-7
IPV6 5-15
DIS (continued)
port 5-9, 5-14
port number 5-21
self reflection 5-14
specifying broadcast modes 5-22
starting in 5-2
startup tutorial 4-9
timeout 5-10
WRM specification 92-1
DIS Enumeration Document 74-4
DIS Enumerations Document 77-18
disable_if_checked 74-25
disable_if_multiple_selection 74-25
disabled-by-suppression parameter 25-32
disableParallelTick parameter B-3
disabling
advisory messages B-7
buoyancy 46-8
cloud layer 11-20
cockpit displays 53-4
creation of a simulation object type 68-8
decal effects 43-3
editing of overlay 39-4
entity labels 16-13
environment 45-19
frame rate statistics B-4
indirect fire event 41-2
indirect rendering 90-35
intervisibility 48-13
jamming 34-22
joystick 80-4
joystick use 80-2
periodic checkpointing 7-41
quit prompt 4-24
radio communication lines 43-26
range rings 43-16, 43-18
reactive task 25-14
rotor wash 46-6
scenario event 8-14
Scenario Startup dialog box 4-8
sensor contact line 43-14
shadows 45-20
speed scaling 51-27, 51-28
spot report 9-3
spot report for simulation object 34-41
squawk indicators 43-26
tactical smoke 43-23
texture compression 64-6
track histories for all simulation objects 43-5

disabling (continued)
trailing effects 43-3
view constraint 14-17
wireframe mode 55-8
disaggregated state, unit 23-9
disaggregated unit 21-2
disaggregation
area 23-12, 38-2
automatic 23-12
manual 23-12
disaggregation range 23-11
Disembark, command 17-7
disembark 17-8
all entities on parent 17-8
Disembark All task 29-3
Disembark task 29-3
Disembarked set data request 17-7, 34-16
disembarked status, testing for 36-10
disembarking
entities 17-2
entity 34-16
instantly 17-8
dispersal-angle parameter 27-22
Disperse Crowd task 28-8
dispersing, crowd 28-8
display
dragging 51-16
field of view 82-14
loading configuration 5-2
navigating 51-3
perspective 82-14
time, format 11-5
unit, changing 83-12
display engine 82-2
configuration 82-2
loading 82-5
saving 82-5
window, adding 82-2
Display Engine Configuration Editor Panel 50-6, 82-8
display mode, curvature of the earth 43-27
Display Settings dialog box 3-13, 9-23, 11-17, 44-8, A-2
Cockpit Display Settings page 53-5
Display Units Settings page 83-14
Entity Display Settings page 16-12, 16-13, 16-19, 16-20, 16-27, 25-5, 43-13, 43-24, 44-3, 44-4, 46-6, 46-7, 46-9, 50-12, 54-10, 54-11
Grid Lines Settings page 55-7
Display Settings dialog box 3-13, 9-23, 11-17, 44-8, A-2 (continued)
High Dynamic Range Settings page 45-17
Indirect Fire Settings page 10-7
Interest Management Settings page 6-11
Intervisibility Settings page 48-6, 48-8, 48-10, 48-13, 48-14
Lighting Render Settings page 45-4, 45-5
Loader Settings page 64-17
Observer Settings page 11-23, 16-22, 43-3, 43-5, 43-8, 43-14, 43-18, 43-23, 44-7, 44-8, 45-7, 46-3, 50-3, 50-4, 50-5, 50-6, 50-7, 50-9, 51-5, 51-14, 51-25, 51-26, 51-29, 53-9, 53-14, 55-6, 55-11
Ocean Settings page 11-26, 11-27, 45-9, 45-10, 45-15, 46-4, 46-6, 57-18, 57-19
Radio Communications Settings page 43-26, 43-27, 43-28
Range Rings Settings tab 43-19
Render Settings page 6-9, 6-10, 6-18, 6-23, 45-14, 53-15, 55-9, 60-9, 60-12, 64-12, 64-13, 64-15, 64-19, 82-12
Sensor Settings page 47-4, 47-5
Shadow Settings page 45-20, 45-23
SpeedTree Settings page 45-22, 90-30
Symbol Decoration Settings page 16-14, 16-15
Tactical Graphics Display Settings page 38-7, 38-8, 40-21, 42-2
Terrain Profile Settings page 48-15
Track History Settings page 43-6
Unit Display Settings page 24-5, 24-9, 24-10, 43-10, 43-23
Display Settings Toolbar 16-15
Display Terrain parameter 7-6
Display Units Settings page 16-20, 38-8, 83-14
displaying
articulated part 68-21
bounding volume 68-20
damage 68-21
entity information 16-9
environment setting 68-21
feature data 55-11

displaying (continued)
full screen 83-8
grid line 55-6
hand item 34-42
height-above-terrain lines 16-20, 38-8
intervisibility
lines 48-13
object 48-8
label
2D 16-14
entity 16-10
laser designator 43-22
markers and marker labels 68-21
navigation areas 61-11
navigation data 61-20
objects 38-6, 38-7
other MAK 3D GUI viewers 50-12
panel 83-3
performance statistics 6-7
range rings 43-16
globally 43-18
per entity 43-19
raster maps 57-6
resources 16-21
scenario information 7-26
Scenario Startup dialog box 4-8
segment outlines 91-5
stealth models 50-12
switch node 68-21
tactical graphics 38-7, 40-24, 41-2
terrain profile, creating line 40-7
toolbars 83-3
track histories 43-5
trailing effects 43-3
transient intervisibility 48-6
vertex labels 38-8
view 51-31
visual effect 68-22
disPort parameter B-8
distance, specifying unit 83-12
Distance To Attached, policy 82-11
Distributed Interactive Simulation 1-22
See also DIS
ditch
anti-tank, digging 30-12
building 30-16
destroying 30-20
improving 30-24
DLL 4-33
cockpit display 90-10
installing 90-11
plug-in, removing 4-37
specifying for plug-in 4-37
dockable panel 83-2
docking, panel 83-2
documentation, location of 2-4
DOF bead 92-8
domain, power 77-3, D-14
doNotUseConsole parameter B-3
door
car
closing 29-16
opening 29-17
cargo
closing 29-16
opening 29-18
closing 28-3
opening 28-12
dotted line 40-23
Douglas sea state 11-22, 11-25
drag and drop, plan statements 37-42
drag chute, setting for aggregate air unit 35-10
drag-and-drop, canceling 14-24
dragging
canceling move 14-24
object to new location 14-23
objects 34-26
simulation object 14-23
terrain 51-14
view 51-16
drainInput
configuring B-4
for DIS B-8
for HLA B-7
dr-allow-gui-overrides parameter 6-20
draw call, coloring 6-23
drawing
area 40-2
box 40-4
vector feature 55-15
driver
embaring as 29-10
OGR 59-10
osgEarth 59-7
simple 59-19
slot 29-10

driving
left or right side or road 25-22
on and off roads 25-21
on roads 25-22
Drop Fuel Tanks task 30-21
Drop Naval Depth Charge at Location task 27-4
Drop Naval Depth Charge task 27-4
Drop Naval Mine task 27-5
Drop Naval Mines Along Route task 27-6
dropping, mine 27-5, 27-6
dropping bomb 27-20
dropping from exercise 3-7
dry gap, building 30-16
DTED
loading 57-3
loading multiple 59-7
support for 57-2
DtSimComponentManager class 74-9
DtSimObjectNetInterface class 74-9
DtSimObjectStateRepository class 74-9
DtSimTaskManager class 74-9
DtVrfObject class 74-9
dtxc, file 90-24
duplicate, key mapping 85-6
Duplicate Model Definition dialog box 87-7
dust cloud 43-3
dynamic avoidance 61-6
dynamic feature 65-10
dynamic lighting 45-2, 45-6
dynamic linked library 4-33
Dynamic Near Clip Attached Policy, attribute 82-11
dynamic ocean 1-14, 11-22
enabling and disabling 11-23
layer 57-14
rotor wash, enabling and disabling 46-6
wake and spray, disabling 46-7
wakes 90-18
dynamic terrain 5-2
damage, high fidelity 77-17
global dynamic terrain processor 7-7
navigation data regeneration 61-16
switch type 62-4
dynamic terrain area 62-3
dynamic terrain damage 77-16
Dynamic Water Visibility Enabled, attribute 82-16
Dynamic Water Visibility Maximum, attribute 82-16
dynamic-terrain-munition-damage controller 77-16
dynamic-terrain-type parameter 77-17

E

-E command-line option 5-3
-e command-line option 63-4
earth file 1-18, 54-5, 59-3
adding mask 59-15
areal features 59-13
buoys and beacons 59-22
caching 64-3
offline 64-5
CDB 60-2
debugging 64-7
differential processing of attributes and elements 59-18
hiding layers 55-10
including files in 59-6
loading DTED 59-7
template 59-6, 64-5
earth gravitational model 59-18
Earth Layers page 55-10
echelon ID 13-2, 13-4, 13-5
assignment of 13-6
unit member 21-3
echelon level, ghosted icons by 24-6
echelon-level parameter 74-39
ECW 54-4
Edit Activities dialog box 71-16
Edit Behavior Sets dialog box 25-19, 25-20
Edit Cloud Layer dialog box 11-21
Edit Existing Props page 57-21, 57-24, 57-25, 57-26, 66-11
Edit Spawn Pattern dialog box 18-15
Edit Trigger Condition dialog box 37-7
editing 11-15
assembly 71-12
attack parameters 71-22
batch file 7-46
channel attributes 82-7
cloud cover 11-20
control object 40-17
defense parameters 71-23
electronic warfare parameters 71-24
element definition 88-8
ellipse 40-20
embarkation ingress and egress points 78-6

editing 11-15 (continued)
entity model 68-3
environment condition 11-8
equipment parameters 71-25
fill style 40-23
force hostility 9-13
formation 70-8
frustum 82-14
icon, 2D 90-3
indirect fire event 10-6
intervisibility object 48-9
key maps 85-2, 85-3
line 40-23
Lua scripts 31-44
model, visual 68-13
model definition 87-2, 87-8
model mapping 89-5
movement system properties 71-20
name, object 14-26
navigation area 61-11
object
in OOB 20-7
location, heading, altitude 14-25
scene 40-18
object map file 12-9
object parameters 68-3
observer mode 50-5
overlay and overlay object 39-4
parameter 69-6
model definition 87-10
schema 86-6
personnel parameters 71-25
plan 37-3
file 12-9
statements 37-11
plans 37-3
posture, transition time 71-21
scenario description 7-29
scenario file 12-3
scenario properties 7-26
script ID 31-11
scripted task 31-37
sensor 75-4
sensor parameters 71-21
sensor signature 75-5
Set menu 84-8
simulation object 67-3
simulation objects 14-25
spawn pattern 18-15
supplies parameters 71-25
editing 11-15 (continued)
system, properties 69-18
tactical graphic 40-17
property 68-32
visual model 68-33
Task menu 84-8
unit 70-4
unit subordinate 70-7
vertex 40-19
visual definition 88-9
parameter 88-11
weather zone, environment 11-15
window attributes 82-6
effect
fire and detonation 43-8
lighting 1-15, 64-10
mapping 89-2
model 85-5
ocean 11-22
trailing 43-3
EGM2008 59-18
EGM84 59-18
EGM96 59-18
egress, technique 72-10
egress point 72-2, 78-6
configuring 72-11
egress-points parameter 78-6
electromagnetic emissions 34-17, 43-24
segment angle 91-4
sensing 22-20
electronic
emission, sensing 22-18
warfare 22-18, 22-19, 22-20
electronic warfare 9-20, 22-18
electronic warfare parameter, editing 71-24
elem file 88-18
element
earth file, differential processing 59-18
informationitem 84-6
informationpagecollection 84-5
mainmenu 84-2
mainmenuconfiguration 84-2
mask 59-15
menuitem 84-7
model 59-9, 59-19, 90-32
option 59-17
pagecollection 84-5
pagecollections 84-5
parent 88-9

element (continued)
rightclickmenu 84-7
toolbar 84-9
toolbarrow 84-9
element definition 88-2
adding visual definition 88-8
changing type 68-22
common buttons 83-10
creating 88-7
deleting 88-14
editing 88-8
exporting 68-25, 88-17
exporting hierarchy and leaves 88-19
filtering list in Simulation Object Type Mappings dialog box 89-6
importing 68-24, 88-15, 88-16
interaction, exporting 88-18
jumping to from model definition 87-11
jumping to from object type mapping 89-7
jumping to model definition 88-14
pruning 68-25
saving 88-15
simulation object 88-2
spot report, regenerating 68-16
element_type 74-25
elevation
coloring terrain by 55-3
data 54-3
intervisibility point, configuring 48-10
Elevation Color page 55-3, 55-4, 55-5
elevation data, EGM 59-18
elevation dataset, CDB 60-3
elevation sub-dataset, CDB 60-4
ellipse
extruded 40-5
rotating 40-20
undoing and redoing rotation 40-20
ellipsoid, reference B-3, B-6
Embark On, command 17-5
Embark On dialog box 17-5, 29-9
Embark task 17-3, 29-4
embarkation
automatic 17-6
condition 36-10
configuring 72-2
slot exclusions 72-9
editing ingress and egress points 78-6
egress point 72-11
embedded entity alternative 17-8
embarkation (continued)
immediate 17-4
ingress point 72-11
instant 17-6
occupancy-director-controller 78-6
slot 78-6
configuring 72-5, 72-11
ingress and egress technique 72-10
Embarkation View 15-2, 17-2, 17-4, 17-8
embarking objects in 17-6
embarkation-slot-entry parameter 78-6
embarkation-slots parameter 78-6
Embarked dialog box 34-16
embarked object, information 16-4
Embarked set data request 17-5, 34-16
embarking
entities 14-17, 17-2
humans in cars 29-10
in Embarkation View 17-6
instantly 17-5
embedded entity 17-8, 29-20
assigning tasks to 17-11
configuring 17-9
deploying 29-19
deploying with task 29-21
recovering 29-19
restoring parent 17-11
Simulation Object Editor 17-9
system 17-9
embedded window 82-2
EM-Emission update message 91-3
emission sensor 22-20
emissions-sensor system 22-20
emitter 34-32
beam 9-24, 34-17
component 34-17
configuring 79-2
ID 34-18
information 16-4
setting 34-17
systems 43-24
Emitter dialog box 34-18
emitter ID, jammer 34-23
Emitter Power 91-3
Emitter set data request 30-4, 30-19, 34-17
emitter volume
color 91-2
configuring 91-2
outlines 91-5

emitter volume (continued)
pulse rate 91-2
radius 91-3
segment angle 91-4
segment size 91-4
Enable Occlusion Queries check box 64-19
enable_if_checked 74-25
enableLogFileTimestamps parameter B-3
enableNavigationLabDebugging parameter 61-12, B-3
enabling
advisory messages B-7
buoyancy 46-8
cloud layer 11-20
cockpit displays 53-4
creation of a simulation object type 68-8
decal effects 43-3
editing of overlay 39-4
entity labels 16-13
frame rate statistics B-4
intervisibility 48-13
joystick use 80-2
marine effects 11-23
periodic snapshot 7-43
radio communication lines 43-26
range rings 43-16, 43-18
reactive task 25-13
rotor wash 46-6
scenario event 8-14
Scenario Startup dialog box 4-8
sensor contact line 43-14
shadows 45-20
sound effects 49-2, 49-3
speed scaling 51-27, 51-28
spot report 9-3
simulation object 34-41
squawk indicator 43-26
tactical smoke 43-23
texture compression 64-6
time management 4-19
in back-end 4-20, 5-21
track histories for all simulation objects 43-5
trailing effects 43-3
wireframe mode 55-8
ending
scenario 7-37
scenario event 8-13
engagement, information 16-7
Engagement Results dialog box 35-7
Engagement Results set data request 35-7
engagement rules 34-35
Engagement Zone, set data request 34-18
engagement zone 34-9, 34-18
joint, creating 40-5
missile, creating 40-5
Engagement Zones dialog box 34-18
Engineering Object Breached condition 36-9
Engineering Object Information page 22-14
entity
See also simulation object
adding to Props Palette 14-35, 87-3
aggregate-level
creating 71-2
parameters 71-17
aircraft, attacking with gun 27-3
altitude
displaying 16-20
testing 36-9
animation 26-4
appearance 92-2
attaching observer to 52-2, 52-8
avoiding obstacles, features, and collisions 13-11
body frame coordinate system 51-7
bounding volume 16-22, 16-23
centering observer on 51-24
clamping to ground 54-9
component, attaching to remote 81-2
components 74-13
configuring
emitter 79-2
obstruction avoidance 78-4
console message 16-23
controlling by keyboard 80-4
controlling with joystick 80-2
copying 14-28
count 16-28
creating 14-3, 14-4, 14-7, 14-11, 14-13
from existing 68-27
in Simulation Object Editor 68-26
creating group of 19-5
creating in plan 37-35
cultural feature, embedding in terrain 14-34
deleting 14-32
from simulation model set 68-27
deleting in plan 37-35

entity (continued)
disembarking 17-2, 34-16
all 17-8
display settings 5-3
dragging 14-23
editing 14-25, 67-3
element definition 88-2
embarkation
configuring 72-2
testing 36-10
embarking 14-17, 17-2
embedded 17-8
enumeration
in condition 37-23
in plan 37-25
extents, zooming on scenario load 51-23
filtering 6-10, 6-11
attachment list 52-6
fixed-wing 25-23
following
entity 26-10
route 26-12
generating navigation data 61-17
ghosted 24-4
ground, road driving 25-21
ground clamping 54-11
heading, setting 14-20
icon
adding new image 90-9
changing size 44-3
image, adding 68-18
inset view 53-2
instrument panel, configuring 53-7
label 16-14
2D 16-10, 16-14
3D 16-9
color 16-9
enabling and disabling 16-13
pinning to display 16-13
level body frame coordinate system 51-7
mapping sound effects to 49-4
messages, viewing 16-25
model 67-6, 85-5
adding visual 68-18
editing 68-3
editing visual 68-13
movement, scripted 69-24
moving 14-23
entity (continued)
moving to
altitude 26-14
location 26-15
object or entity 26-18
name 16-9
changing 14-25
pasting 14-29
notification level, object console 16-24
object geometry 69-7
organization 13-5
pasting, control object 14-29
patrolling between 26-23
patrolling route 26-23
placing 14-13
new 14-17
on entity 14-17
platform, updating 67-25
primary 52-11, 52-14
properties, specifying at creation 14-15
random creation of 14-33
range ring 43-19
resource
at creation 14-22
monitoring 16-21
saving console messages 16-26
scaling, 3D model 44-5
secondary 52-11, 52-14
setting
collision avoidance types 34-11
target 34-43
shadows 45-20
sound effects 49-2
spawned 18-2
spawning, maximum simultaneous 18-12
specifying for multiple entity creation 19-9
surface
buoyancy 46-8
configuring wake 90-18
symbol, changing color 67-33
systems 69-15
tag, changing 68-4
task, visualizing 43-12, 43-13
tick rate 6-15
trailing effects 43-3
type, enumeration 16-9, 89-4
unit 23-3
velocity, publishing B-4

entity (continued)
visual markings 68-23
VR-Forces 3-9
wake, enabling and disabling 46-7
ways to add to scenario 7-14
zooming to extents 51-23
Entity Altitude condition 36-9
Entity Behavior Settings page 23-6, 23-11, 23-12, 25-22
Entity Definition Editor page 90-3, 90-4, 90-18, 91-6
Entity Destroyed condition 36-9
Entity Di-Guy Animation Check condition 36-10
Entity Di-Guy Appearance Check condition 36-10
Entity Display Settings page 16-12, 16-13, 16-19, 16-20, 16-27, 25-5, 43-13, 43-24, 44-3, 44-4, 44-8, 46-6, 46-7, 46-9, 50-12, 54-10, 54-11
Entity Editor. See Simulation Object Editor
Entity Embarked condition 36-10
entity enumeration, changing 68-5
entity file, importing 67-25
Entity Has Target condition 36-10
Entity Image Symbol, visual definition 90-3, 90-4, 90-9
Entity In Area, condition 36-11
Entity In Area condition 40-5
Entity Left of Line condition 36-11
Entity List, filtering 15-6
Entity Mapping Settings page 89-4, 89-5, 89-7
entity plan 36-2
Entity Resource Manager D-6
entity state repository 76-10, D-5
entity type, changing 68-5
Entity Under Fire condition 36-11
entity-level, modeling 13-9
entity-level modeling 20-1
EntityLevel.sms 13-9
entity-range parameter 76-12, D-7
entity-type parameter 76-12, D-7
entrance, embarkation 72-11
entry point, scripted task 32-6
entVrfDataTimeThreshold parameter B-3
enumeration 74-4, 89-4
entity, changing 68-5
for condition patterns 37-25
enumeration 74-4, 89-4 (continued)
in condition 37-23
object type C-9
published and matching 74-6
environment
disabling 45-19
global 7-7
loading configuration 5-3
weather zone 11-15
editing 11-15
environment condition 11-6
adding 11-7
deleting 11-8
editing 11-8
marine, setting 11-24
Environment Conditions dialog box 7-12, 7-13, 11-4, 11-5, 11-13
Environment Conditions Settings page 90-20
environment setting, displaying 68-21
Environment Settings Toolbar 11-5
Environment Settings toolbar 7-12, 7-13, 11-4, 11-5
environment variable
GDAL_CACHEMAX 64-7
MAK_GEOID_GRID 57-3
MAKLMGRD_LICENSE_FILE 2-8
OpenSceneGraph, anaglyphic stereo 82-22
OSG_CURL_PROXY 58-5
OSG_CURL_PROXYPORT 58-5
OSG_ICO 6-14
OSG_MAX_PAGEDLOD 64-7
OSGEARTH_CACHE_PATH 64-3
OSGEARTH_CURL_PROXYAUTH 58-5
RTI_CONFIG 2-10
VRFSIM_OSGEARTH_CACHE_PATH 64-3
VRV_ALPHA_TO_COVERAGE_RENDERBIN 6-17
Environmental Object Information dialog box 16-3
ephemeris 7-7
model 11-3
equipment
pacing and tracking 35-8
unit 71-4, 71-7
Equipment Pacing/Tracking dialog box 35-8

Equipment Pacing/Tracking set data request 35-8
equipment parameter, editing 71-25
equivalent task 31-29
equivalent-tnt-mass, determinant 77-6, D-17
escaping
  plan 37-44
  task assignment 25-6
escort, waiting for 30-33
Escort Air Unit dialog box 30-22
Escort Air Unit task 30-22
event 13-8
  linked, adding 8-6
  list, numbering 8-15
  scenario 8-2
Event window 8-10, 8-13
EW-Comms-Degradation-Percentage 22-18
EW-Comms-Dependence 22-18
EW-Defense-Strength 22-18
exaggerated reality 50-10
exaggerating, 3D models 44-5
exclusion, slot 72-9
execName parameter B-7
exectool, module 32-5
executable, installation locations 2-4
Execute Close Air Support dialog box 27-7
Execute Close Air Support task 27-7
execution name, HLA, specifying 5-19
exercise
  clock 3-10
  starting 7-34
exercise ID B-8
  DIS 5-9, 5-15
exerciseId parameter B-8
exercise-start--time parameter 12-6
exit, embarkation 72-11
exiting
  application 4-23
  full screen window 83-8
  plan 37-44
  prompt 4-24
  scenario 7-37
  simulation engine 4-23
expanded label 16-9
  pinning to display 16-13
expanding
  unit 24-2, 24-4
    all 24-4
Experimental Features dialog box 11-28, 51-21, 51-22, 64-15
Explode Charge at Depth task 27-9
explosion 43-9
explosive device, arming 34-9
explosive device system 76-19, D-12
explosive power 76-18, D-10
explosive power domain 77-3, D-14
Export Object Information dialog box 67-26
exporting
  element definition 88-17
  element definitions 68-25
  hierarchy 88-17
  interaction element definition 88-18
  model definition 87-13
  model definitions 68-25
  object information 67-26
  object type mappings 68-25
  Order of Battle 20-7
  overlays to shapefiles 39-7
  props to shapefile 57-27
  scripted task 31-39
  settings 3-13
  tactical smoke element definition 88-18
  tags 67-30
  visual models 68-25
extended imagery dataset, CDB 60-15
extended imagery sub-dataset, CDB 60-16
extended label 16-17, 38-3
  adding 16-18
  deleting 16-19
  index, changing 16-19
extended name 31-12
extent, zooming to 51-23
external communication model 79-4
external reference 90-34
external script, editing 31-44
Extract Ocean page 57-15
Extract Props page 57-21, 66-9
extracting
  prop
    from terrain 54-4
    from terrain patch 57-20
  water texture 57-15
extras file 12-5
extruded
  building 59-14
  tactical graphic 40-5
extruded area 36-11

extruded geometry 54-5
eyepoint. *See* observer

F

-F command-line option 5-3, 5-16
-f command-line option 5-7, 5-12, 5-20, 63-4
face front culling 46-9
Facilities dialog box 35-9
Facilities set data request 35-8
factory
  window layout, restoring 83-7
factory settings, restoring 3-13
fan, intervisibility 48-4
far clipping plane 82-9
FASCAM task 30-22
Fast Rope Insertion dialog box 26-6
Fast Rope Insertion task 26-6
favorite 14-7
  adding or removing from list 14-36
  joystick configuration 80-15
  observer view 51-33
Favorites, clearing list 68-10
favorites.mst 14-36
FDD file 2-10
feature 1-13
  areal 59-13
  attribute 65-4
  avoiding 13-11, 78-4
  configuration file 65-6
  configuring avoidance of 78-5
  coordinate 55-16
  data
    buoys and beacons 59-22
    configuring 65-2
    configuring in front-end and back-end 57-12
    displaying 55-11
    extracting from GDB 65-7
    layer 65-7
    ocean 57-28
    processing 59-8
  dynamic 65-10
  extracting as prop 57-20
  geometry 54-5
  layer, adding 54-4
  query 65-6
  querying 65-3
  soil type 55-16
feature 1-13 (continued)
  SwitchStateMapper 90-23
  UvAtlasMapper 90-23
  vector
    draw area 55-15
    list of 55-13
feature layer 59-22
  adding 57-11
  adding prop from 57-22
  name, specifying 65-2
Feature Settings page 55-14
Feature_Data_Dictionary.xml 60-17
featureconfig.txt
  mapping, features 57-12
featureconfig.txt 78-5
featureDataBroker 59-8
FeatureSet class 32-2
FED file 2-10
  name 5-7, 5-12, B-7
  specifying 5-21
federate
  name 5-8, 5-13, B-7
  time managed 4-19
federateName parameter B-7
federateType parameter B-7
federation execution
  destroying B-7
  name B-7
  specifying name 5-19
federation time 4-19
fedFileName parameter B-7
Feet Above Ground Level 90-12
Feet Above Mean Sea Level 90-13
FeetAGL updater 90-12
FeetMSL updater 90-13
FeetPerMinute updater 90-13
field of view 51-19, 51-20, 82-14, 82-15, 91-3
  changing 82-14
file
  ammo select D-27
  batch 7-45, 7-46
  cache, clearing 64-6
  caching 64-2
  compressed 90-35
  damage 77-2, D-13, D-18, D-29, D-31
  default_ModelPreload.mpsx 6-24
  dtxc 90-24
  FED 2-10, 5-21
  formation 78-2

file (continued)
geometry 90-22
hit D-28
hostility relationships 12-10
installed 2-4
log 4-40, B-3
MTL A-3
object map 12-9
OPE 76-2
physicalWorldParams.mtl 75-11
plan 12-9
platform 74-16
RID 2-10, 5-19
rid.mtl 5-19
scenario 6-1, 12-2, 12-3, 12-7
scenario extras 18-10
scenario objects 12-2, 12-10
scnx 6-1, 12-2
simulation model set 67-6
system definition 76-2
transporter 5-3
vrfGui.log 4-40
vrfSim.log 4-40
vrfSim.mtl B-2
vrfSimSettings.xml 6-19
File Caching Settings page 6-13, 64-2, 64-6
Filename parameter 90-12
filename_shp_map.txt configuration file 57-11
fill style 40-23
Fill Style dialog box 41-4
Fill Style set data request 41-4
filtering
earth file layers 55-10
element definition list 89-6
function list 85-7
model definition list 87-8
object attachment list 52-6
object creation palette 14-8
object lists 15-6
scripted task list 31-37
simulation object list 25-9
simulation objects
by observer altitude 6-11
in scene 6-10
Find Cover dialog box 28-9
Find Cover task 28-8
Find Firing Position dialog box 28-10
Find Firing Position task 28-9
finite state machine 32-5
fire
effects 43-8
event
mapping sound effects to 49-4
sound effects 49-2
indirect 10-2
line 43-9
suppressive 25-30, 27-19
at location 25-30
at point, line, or area 25-30, 27-19
target 27-9
Fire at Target task 27-9
Fire Cruise Missile dialog box 27-10
Fire Cruise Missile task 27-10
fire interaction 76-15, 76-16
Fire-for-Effect task 27-11
firepower-kill parameter D-23
fire-when-fired-upon 34-35
firing
at target 27-9
ballistic missile 10-9
cruise missile 27-10
effects 89-2
target at 9-16
spot report 9-18
Fixed Near Clip attribute 82-11
Fixed Near Clip When Attached, attribute 82-11
fixed trajectory missile 10-8
fixed weapon system, configuring 76-19, D-11
Fixed Wing Land dialog box 26-7
Fixed Wing Land task 25-27, 26-7
Fixed Wing Takeoff dialog box 26-7
Fixed Wing Takeoff task 25-26, 26-7
fixed-frame 4-20
fixed-frame best-effort 3-12, 12-6
fixed-frame keyword 12-6
fixed-frame-run-to-complete 3-12, 4-20
fixed-frame-run-to-complete keyword 12-6
fixed-wing
altitude 26-9
holding pattern 26-24
landing at airport 26-11
strafing target 27-22
fixed-wing entity
altitude 26-8
attacking with gun 27-3
circling a point 26-21
counter measures 27-15

fixed-wing entity (continued)
creating 25-23
dropping bombs 27-20
follow entity behavior 26-10
heading 26-8, 26-9
landing 25-27, 26-7
setting
altitude 25-24
IFF transponder 34-21
speed 34-30
takeoff 25-26
takeoff and landing 25-25
taking off 26-7
fixed-wing-entity-param parameter 74-12, C-8
fixed-wing-entity-state-repository 74-12
flag, movement in wind 90-21
flaming effects 43-8, 89-2
flare 27-15
launching 9-26
flat earth, coordinate system 51-7
Flee From Entities dialog box 28-11
Flee From Entities task 28-11
FLEXlm 2-6
flight plan, generating air traffic from 26-11
flipping, DDS textures 57-5, 90-29
FLIR 47-2
floating, panel 83-2
FLT file 54-3, 85-5
Fly Altitude task 26-8
Fly Heading and Altitude task 26-9
Fly Heading task 26-8
flying
altitude to 26-8, 26-9
heading to 26-8, 26-9
focal depth 53-13
focal range 53-13
focus, scene 53-13
fog 11-6
denseness of 11-11
height and color 11-11
fog of war 5-3
fog-of-war 9-2
same side 9-6
folder
adding scripted task to 31-38
removing scripted task from 31-38
folder (continued)
scripted task 31-38
adding 31-38
deleting 31-38
renaming 31-38
follow entity, fixed-wing entity behavior 26-10
Follow Entity dialog box 26-10
Follow Entity task 26-10
in plan 36-17
Follow mode 51-8, 52-8, 52-9
Follow Route task 26-12
following
paths 61-5
route 26-12
simulation objects 26-10
FOM 2-9
module 5-7, 5-12
FOM Mapper
configuring B-7
initialization data 5-7, 5-12
initialization string B-7
library name 5-7, 5-12
specifying at startup 5-20
fomMapperInitData parameter B-7
fomMapperLib parameter B-7
food 22-15
pacing and tracking 35-10
setting 35-9
Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking set data request 35-10
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant dialog box 35-9
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant Pacing/Tracking dialog box 35-10
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant set data request 35-9
footprint 43-3
affect of posture 22-6
sensor 22-10
unit 22-5
force 13-5, 16-9
adding 67-31
behavior set, assigning to 25-20

force 13-5, 16-9 (continued)
color
2D icon 16-15
changing 67-33
editing 67-33
hostility
changing 67-33
changing in plan 9-15
matrix 9-13
name, changing 67-33
removing 67-33
setting at runtime 34-18
Force dialog box 34-18
Force set data request 34-18
tactical graphic 41-4
forceHostilty.mtl 9-13
forceParameterDbReload parameter B-3
format
OpenFlight 92-2
time display 11-5
formation 21-4
adding 70-9
bounding box 70-11
closing 21-5
configuring 78-2
copying 70-12
default 70-10
editing 70-8
expanding and collapsing in Formation Editor 70-10
file 78-2
layout
automatic 70-13
manual 70-13
moving to 26-13
removing 70-10
renaming 70-10
setting 34-19
in plan 36-17
snap to grid 70-14
supported 34-19
user-defined 78-4
Formation dialog box 34-19
Formation Editor 70-9, 70-10, 70-12, 70-13
Formation set data request 34-19
in plan 36-17
fortification
breaching 30-9
destroying 30-21
improving 30-24
fortified area, constructing 30-17
fortified line, constructing 30-18
fragmentation power domain 77-3
frame mode, changing for fast forward 7-35
Frame Mode parameter 7-7
frame rate 5-12, 6-4, 6-22, 6-24
exercise clock 3-12
improving 64-19
mode, specifying 12-6
script 6-6
setting 5-3
statistics, enabling and disabling B-4
Frame Rate Monitor Panel 6-6
frame time, changing for fast forward 7-35
Frame Time parameter 7-7
frame-mode parameter 4-20, 12-6
time management 4-20
framentation power domain D-14
frame-time parameter 12-6
free-fly mode 52-8
freehand line 40-3
sampling rate 40-3
frequency
color of 91-2
radio communication line 43-28
sampling for terrain profile 56-3
front-end 3-2
coordinating multiple 3-6
feature representation 54-5
joining session at startup 4-17
multiple 3-2
process ID 5-10
selecting database 7-3
sending messages 3-3
starting 4-3
frustum 82-7, 82-20
editing 82-14
fsm, module 32-5
fuel 22-15, 34-10
pacing and tracking 35-10
setting 35-9
specifying amount 34-34
status 16-6
fuel tank, dropping 30-21
Full Color Mode 47-5
Full Knowledge CID level 9-16
full screen 5-6
hiding mouse 83-9
mode 5-3

full screen 5-6 (continued)
window 82-2, 83-8
function
list, filtering 85-7
movement 85-2
profiler 6-10
setCheckpointMode 32-13

G

-G command-line option 5-3, 5-18
-g command-line option 63-5
gain 47-2
changing 51-29
sensor effects 47-4
game pad 80-2
Gamepad, installing on Linux 80-15
Gameware Navigation Lab 61-20
gamewareMemorySize parameter B-3
gamewareQueryTimeBudget parameter B-3
gasoline 22-15
GDAL 65-7
configuring cache 64-7
GDAL_CACHEMAX, environment variable 64-7
GDB, extracting feature data 65-7
GDB terrain 54-3
configuring soil type 55-19
shapefile 54-4, 57-11
General Application Settings page 4-14, 4-24
Generate Air Traffic From Flight Plans task 26-11
generating
air traffic from flight plan 26-11
navigation data 61-13, 61-14
entity for 61-17
traffic 18-2
geocentric
converting to geodetic B-3, B-6
coordinate system 51-7, 54-6
vector 32-22
terrain 54-6, 58-2
geodetic, converting from geocentric B-3, B-6
geographic DDM 6-10, 6-11
geoid grid file 57-3
geometric, objects 32-19
geometry 90-34
extruded 54-5
geometry 90-34 (continued)
file 90-22
object, adding to entity 69-7
sharing 6-13
Geometry Grid Dataset 63-3, 63-7
geospecific airport lightpoint sub-dataset, CDB 60-9
geospecific dataset 60-6
geospecific environmental lightpoint sub-dataset, CDB 60-10
geospecific sub-dataset, CDB 60-8
GeoTIFF 54-4
image 57-8
geotypical dataset, CDB 60-13
geotypical sub-dataset, CDB 60-15
gesture, task 28-7
GGDS 63-7
ghosted
icon, by echelon level 24-6
unit 24-4
ghosted units 24-4
GIF 54-4
gimbaled sensor 34-38, 34-40, 53-8
control panel 53-10
field of view 53-9
setting azimuth 53-12
GL Studio 1-19, 53-4, 90-10
instrument panel 53-6, 53-7
license 1-20
global
environment 7-7
weather 7-7
global command
adding to plan 37-31
Create 37-35
Delete Object 37-35, 37-36
Delete Self 37-36
Issue Plan 37-36
global dynamic terrain processor 7-7
global plan 11-3, 36-2
creating 37-27
file 6-1
global setting 3-16
God ray 45-13
graphic
adding to scenario event 8-5
buffer, depth and stencil 6-16
depth 5-2
fire and detonation 43-9
mode, wireframe 55-8

graphic (continued)
object, showing and hiding 38-6, 38-7
unit combat 43-10
graphical object 38-2
changing, line width 40-23
deleting 40-17
spot report 9-8
graphical user interface 3-2
configuring 84-2
gravity, aligning cultural feature 69-10
grenade, hand 27-24
grid line
configuring 55-7
displaying 55-6
keyword 82-17
Grid Lines Settings page 55-7
grid spacing, changing 70-14
GridLines keyword 82-17
ground, attacking from air 30-5
ground clamping 54-9
configuring 54-11
cultural feature 69-10
Ground Clamping Cutoff Distance Scale 54-11
ground truth 9-2, 9-4
ground vehicle
configuring obstruction avoidance 78-4
road driving 25-21
ground-entity-state-repository 74-12
ground-vehicle-param parameter 74-12, C-8
ground-vehicle-param parameter 74-3
group
entities, adding 19-9
selection 15-8
simulation object 14-32
GUI 3-2
configuring 84-2
layout, saving 83-3
layout settings 5-4
localizing 2-11
locking 84-10
settings
file 6-1
scenario 12-2
Simulation Object Editor, layout settings 67-5
GUI Settings page 4-8, 4-23, 7-28, 7-50, 68-12, 83-11, 84-3
guidance-mode parameter 77-3, D-14
guidelines 90-34
creating models 90-34
GuiObserverViews parameter 12-7
GuiScenarioSettings parameter 12-7
Gui-Terrain-Database parameter 12-5
gumball 16-6
gun
aircraft, attacking aircraft with 27-3

H

-H command-line option 5-8, 5-9
-h command-line option 5-4, 5-10, 5-16, 63-5
Halt Movement task 30-22
hand grenade 27-24
hand item, hiding 34-42
Has Drag Chute dialog box 35-10
Has Drag Chute set data request 35-10
Hasty-Attack, posture 22-6
Hasty-Defense, posture 22-6
HAT, *See* height-above-terrain line
hazard, area 22-10
HDR 45-17
header file, *objTypeEnums.h* C-9
heading 16-9, 51-3
flying to 26-8, 26-9
indicator 14-20, 16-10
intervisibility line 48-14
object, changing 14-25, 40-18
observer 51-4
pasting 14-29
rotating icon to 44-4
setting 14-3, 34-20
manually 14-20
specifying measurement unit 83-12
turning to 26-29
zig zag to 26-29
Heading dialog box 34-20
Heading set data request 34-20
headlight 45-2, 45-6
Headlight updater 90-13
health 22-3, 22-4
restoring simulation object 34-34
Health dialog box 35-11
heartbeat
back-end, configuring B-6
DIS B-7
relationship to thresholds 6-19
varying B-8

heartbeatVariance parameter B-8
heavy lift system 26-26
height
  fog 11-11
  ocean 57-16
height map
  ocean
    regenerating 57-19
    resolution 57-17, 57-18
    texture size 57-17, 57-19
height-above-terrain line 16-20, 38-6, 51-27
  displaying 38-8
helicopter
  fast rope insertion 26-6
  rotor wash, high quality 46-6
  terrain avoidance 25-29
help 5-10, 5-16
heuristic, path planning 61-4
HiddenTerrainViews parameter 12-7
Hide Occupants 29-10
Hide Occupants set data request 34-20
hide_by_default 74-25
hiding
  3D models 44-8
  hand item 34-42
  intervisibility lines 48-13
  mouse 83-9
  objects 38-6, 38-7
  overlay objects 39-4
  panel 83-3
  simulation objects 9-2
  symbol decorations 16-15
  tactical graphics 40-24, 41-2
  toolbar 83-3
  vertex labels 38-8
hierarchy
  element definition 88-2
  entity 13-5
  exporting 88-17, 88-19
  importing 88-15
hierarchy file, importing 88-16
high dynamic range 45-17
High Dynamic Range Settings page 45-12, 45-17
high fidelity damage area 77-17
high-explosive, resource 22-15
high-fidelity-damage-table 77-17
high-fidelity-terrain-damage-table block 77-17
hit file 76-8, D-2, D-28
hit probability
  file 74-40
  table 76-12, D-7
HLA 6-10, 6-11
  1516
    troubleshooting 7-50
  command-line options 5-19
  federation execution name, specifying 5-19
  parameters B-7
  specifying federation execution 5-19
  starting in 5-4
  startup tutorial 4-9
  time management 4-19
holding pattern
  fixed-wing 26-24
  rotary-wing 26-24
horizontal navigation 85-2
host address 5-8, 5-9, B-7, B-8
hostAddressString parameter B-7, B-8
hostile-fire-only parameter 25-32
hostility
  changing, plan in 9-15
  matrix, force 9-13
hostility relationships file 6-1, 12-10
hostname, license server 2-7
HUD. *See* cockpit display
human
  getting in and out of car 29-10
  idling 28-3
  releasing from queue 29-23
  sliding down rope 26-6
  waiting in line 28-15
human-param parameter 74-12, C-8
humans, adding to crowd 28-4
human-state-repository 74-12

I

-I command-line option 5-4
-i command-line option 4-9, 5-8, 5-9, 5-10
icon
  2525B 90-3
  2D 14-11, 90-4
    bitmap 90-4
    changing 68-16
    displaying label 16-14
    editing 90-3
    image 68-17

icon (continued)
bitmap, adding new 90-9
borders 44-2
changing on objects list panel 68-10
common 83-10
configuring 2D 90-3
heading, rotating to 44-4
image 90-4
adding new 90-9
level of nesting 24-4
list, changing 68-10
location of 2-4
notification 16-27
scaling, 2D 44-3
Set menu 31-12
settings 3-13
simulation object 44-3
size, changing 44-3
spot report
changing 68-18
updating 68-16
tactical graphic 68-30
Task menu 31-12
toolbar, setting 84-10
triangle 16-27
warning 16-27
yellow 9-16
icon list
creating, editing, and removing 68-10
specifying entity icons 68-11
using 68-12
ID
application 5-8
echelon 13-4, 13-5
emitter 34-32
marking, license plate number 68-23
object 13-4
script 31-8, 31-11, 31-27, 32-14
editing 31-11
session 4-3, 5-8, 5-9
site 5-11, B-6
task 32-14, 32-15
unique 13-3
identification
simulation object, UUID 13-2
Identification Friend from Foe 34-21
identifier, site and application 3-4
identifying, simulation objects 13-2
idling, human 28-3
IED 76-19, D-12
arming 34-9
detonating 34-14
placing 27-18
If block, creating 37-5
If statement 36-3, 36-5
building 37-13
If/Else Expression dialog box 37-5
IFF 34-21
information 16-4
IFF dialog box 34-21
IFF set data request 34-21
ignore-sets parameter 84-8
illumination 75-14
model 7-7, 11-5
setting 11-3
image
2D icon 68-17
adding 68-18, 90-9
attributes 57-7
compressing 90-35
display order 57-9
displaying 57-6
format, supported 54-4
GeoTIFF 57-8
icon 90-4
list in script 31-20
texture coordinate system 57-8
imagery, filtering 55-10
imagery dataset, CDB 60-5
imagery lightmap sub-dataset, CDB 60-6
imagery sub-dataset, CDB 60-5
imagesymbolmap attribute 90-7, 90-9
ImGui'overlay, debug 6-27
immediate embarkation 17-4
impact-point range 77-4, D-15
Import Object Information dialog box 67-26
importing
element definition 88-15, 88-16
element definitions 68-24
entity file 67-25
hierarchy 88-15
hierarchy file 88-16
leaf file 88-16
leaf files 88-15
model definition 87-12
model definitions 68-24
MSDL 7-14, 7-16, B-4
navigation area 61-19

importing (continued)
navigation data 61-19
object from CSV files 7-18
object information 67-26
object type mappings 68-24
order of battle 20-8
props from shapefiles 57-27
scenarios 7-14
scripted task 31-39, 31-40
settings 3-13
shapefiles to overlay 39-7
SpeedTrees from shapefile 57-27
tags 67-31
visual models 68-24
Improve Booby Trap dialog box 30-23
Improve Booby Trap task 30-23
Improve Breach dialog box 30-23
Improve Breach task 30-23
Improve Bridge dialog box 30-23
Improve Bridge task 30-23
Improve Ditch dialog box 30-24
Improve Ditch task 30-16, 30-24
Improve Fortification dialog box 30-24
Improve Fortification task 30-24
Improve Obstacle dialog box 30-24
Improve Obstacle task 30-24
improve-script-id parameter 31-27
improving
anti-tank ditch 30-24
booby trap 30-23
breach 30-23
bridge 30-23
ditch 30-24
fortification 30-24
obstacle 30-24
strong point 30-24
improvised explosive device, placing 27-18
inactive, tactical graphic 40-24, 41-2
including XML files in earth files 59-6
increasing speed of navigation 51-27
incremental compiling 6-14
independent mode, tutorial 4-9
independent setting, global and observer 3-16
independent task 24-1, 25-3
index
extended label, changing 16-19
indicator 16-13
heading 16-10
squawk 43-25
indirect fire 10-2
activation 40-24
creating 10-3
editing 10-6
trajectory 34-32
Indirect Fire dialog box 30-25
indirect fire event, disabling 41-2
Indirect Fire page 10-3, 10-6
Indirect Fire Settings page 10-7
Indirect Fire task 30-25
indirect fire weapon system 76-17, D-9
indirect rendering 60-14, 64-14
disabling 90-35
LOD distance scale 64-15
LOD transition blending 64-15
indirect-fire-actuator 76-17, D-9
indirect-fire-controller 76-17, D-9
indirect-fire-weapon-controller 76-17, D-9
individual, plan 11-3
influenceName parameter 32-7
influenceParameters parameter 32-7
influencer parameter 32-7
information
connection 4-31
entity label 16-9
multiple objects 16-8
object 16-3
scenario, displaying 7-26
Information dialog box 16-3, 16-21
centering observer from 51-24
display user-created state properties 74-29
information dialog box, configuring 84-5
information message B-4
informationitem, element 84-6
informationpagecollection, element 84-5
infrared 47-2
ingress, technique 72-10
ingress point 72-2, 78-6
configuring 72-11
ingress-points parameter 78-6
inheritance, element definition 88-2
Initial Scenario Information dialog box 7-3, 7-11
Initial Unit State 23-6
initialization string, FOM Mapper B-7
initial-road-preference parameter 34-28
initialSetMenuConfiguration.xml 84-8
initialTaskMenuConfiguration.xml 84-8

input
of data 74-13
port and port group 74-14
inserting, altitude point 55-4
inset view 53-2, 53-3
installing, RTI 2-10
instancing 5-4
DI-Guy 6-26
instant embarkation 17-4
instrument panel 53-6
channel keyword 82-17
configuring 53-7
GL Studio 53-6, 53-7
instrumentPanelCategory parameter 53-7
Intel Collection Area 40-11
intelligence, artificial 34-5
intelligence object 8-8
intensity, rain and snow 11-12
interaction 3-13
detonation 76-17
element definition, exporting 88-18
fire 76-15, 76-16
signal 43-25
interaction element definition, exporting 88-18
interactive script 32-25
interest management 6-10, 6-11
configuring 6-11
Interest Management Settings page 6-11
interface
Lua 32-2
weapon 76-9, D-4
intersection, KD trees 5-4
intersector line, terrain 64-16
Intersector Settings page 64-16
interval
spawn 18-7, 18-9, 18-13
timeout 5-8
intervisibility 48-2
enabling and disabling 48-13
entity 48-3
fan 48-3, 48-4
moving 48-9
grid 48-3
line 38-2
hiding 48-13
information 48-14
intervisibility 48-2 (continued)
object
configuring 48-10
deleting 48-16
displaying 48-8
editing 48-9
transient and permanent 48-5
pinning object to simulation object 48-12
point-to-point 48-3
terrain profile 48-15
terrain profile line 56-5
types 48-3
intervisibility fan, simulation object 48-4
Intervisibility Settings page 48-6, 48-8, 48-10, 48-13, 48-14
invalid model definition 87-10
invert pattern 37-23
Invert Selection check box 37-24
inverting condition parameter 37-23
Invisible set data request 34-22
Invulnerable dialog box 34-22
Invulnerable set data request 34-22
IP address 5-6, B-5
IPV6 5-9, B-8
DIS 5-15
Issue Plan, command 37-36
itemname, attribute 84-2, 84-9
IVE 58-6

J

jammer
activating 30-4
deactivating 30-19
emitter ID 34-23
mode 34-23
Jammer State dialog box 34-22
Jammer State set data request 34-22
jamming
communications 22-18
enabling and disabling
enabling, jamming 34-22
radar 9-20, 22-18, 22-19
target 34-4
Jamming Mode dialog box 34-23
Jamming Mode set data request 34-23
jamming-defense-factor parameter 22-19
jam-strength-factor parameter 22-20
jitter, camera 51-25

joining exercise, late 3-7
joint engagement zone
  altitude limited, creating 40-5
joystick 80-2
  configuration
    creating 80-5
    favorite 80-15
  configuring 80-4
  control, mapping 80-7
  controller 80-4
  disabling 80-4
  Gamepad 80-15
  keyboard contorl 80-4
Joystick button 80-7
Joystick Configuration page 80-4, 80-5, 80-7, 80-8, 80-11, 80-15
JPEG 54-4
JPG 54-4
JRM Technologies 47-2
jumping 34-31
  observer 51-17

K

KD tree 5-4
key map 85-2
  creating 85-7
  deleting 85-7
  editing 85-3
key mapping 51-11
  adding 85-3
  binary 85-3
  changing 85-5
  combining 85-6
  deleting 85-5
Key Mapping button 80-8
Key Mapping Editor 85-2
  adding mapping 85-3
  binary mapping 85-3
  changing mapping 85-5
  deleting mapping 85-5
  filtering function list 85-7
Key Mapping Editor page 85-3, 85-7
key value pair, reader writer 71-27
keyboard 85-2
  attaching to objects from 52-4
  changing speed 51-29
  control 83-8
    mapping 80-8
  controlling entities 80-4
keyboard 85-2 (continued)
  creating selection group from 15-9
  entity control, configuring 80-4
  inset views 53-3
  mappings 51-6, 51-10, 85-2
  movement functions 51-6
  moving observer 51-10
  navigation 85-2
key-names, location of configuration files 81-3
keyword
  channel 82-7, 82-17
  cockpitOnly 82-17
  GridLines 82-17
  largeCompass 51-5, 82-17
  NoExternalInput 82-17
  showCockpits 82-17
  showRemoteGraphics 82-17
  smallCompass 51-5, 82-17
  tactical graphic property 68-32
KilometersPerHour updater 90-13
kinetic power domain 77-3, D-14
kneeling 34-31
KnotsPerHour updater 90-13

L

-L command-line option 5-4, 5-10, 7-26
-l command-line option 5-4, 63-5
label 13-2
  2D
    background 16-16
    displaying 16-14
    showing and hiding 16-15
  entity 16-14
    2D 16-10
    3D 16-9
    enabling and disabling 16-13
  extended 38-3
  object 13-4, 16-14, 38-3, 38-8
  overlay 39-4
  pinning to display 16-13
  simulation object 13-4, 16-14
    extended 16-17
  spot report 9-8
  tactical graphics 38-3, 38-8
Lambert Conformal Conic, coordinate system 51-7
lamppost, prop type 57-26
Land at Air Base dialog box 30-25

Land at Air Base task 30-25
Land at Airport Near Location task 26-11
land attack, missile 30-6
landing
  air base 30-25
  fixed-wing entity 25-25, 25-27, 26-7
  rotary-wing entity 26-26
language
  scripted task 31-30
  specifying 5-3
    for GUI 5-18
largeCompass keyword 51-5, 82-17
Lase Autonomous dialog box 34-24
Lase Autonomous set data request 9-19, 34-24
Lase Target task 9-19, 27-13
laser
  autonomous 34-24
  bomb 27-20
  code 34-24
    range 9-18
    synchronizing between entities 9-19, 34-43
    using 9-18
  designator, displaying 43-22
  power 34-25
  targeting 27-13
  wavelength 34-25
Laser Code dialog box 34-24
Laser Code set data request 9-18, 9-19, 34-24
laser guided missile 76-16
Laser Power and Wavelength dialog box 34-25
Laser Power and Wavelength set data request 34-25
laser-guided missile 76-17
lasing, autonomous 9-19
lasing controller 76-16
Last Clicked Location Panel 55-16
Last Clicked Object Panel, centering observer from 51-24
late joiner 3-7
lateral-clearance parameter 78-4
latitude 16-9, 32-19
Launch Anti-Submarine Missile (Vertical) task 27-13
Launch Ballistic Missile at Location task 27-14
Launch Ballistic Missile at Target Object task 27-14
Launch Counter Measures, task 9-26
Launch Flight dialog box 30-27
Launch Flight task 30-26
Launch QRA Aircraft dialog box 30-27
Launch QRA Aircraft task 30-27
Launch Smoke task 27-15
Launcher, VR-Forces 4-4
launcher controller 76-16
launching
  counter measures 9-26
  torpedo 27-16
layer
  cloud 11-17, 40-14
  earth file
    debugging 64-7
    hiding 55-10
  feature 59-22
    adding 54-4, 57-11
    adding prop from 57-22
  feature data 65-7
  name 59-10
    shapefile 59-10
  ocean 57-14
    adding 57-16
layout
  default, clearing 83-6
  formation
    automatic 70-13
    manual 70-13
  GUI 5-4, 67-5
    choosing 83-5
    creating 83-4
    default 83-5
    deleting 83-7
    renaming 83-6
    restoring factory 83-7
    reverting 83-6
    saving 83-3
    updating 83-6
LCC, coordinate system 51-7
leader-promotion-id parameter 78-2
leaf, exporting 88-19
leaf file, importing 88-16
leaf files, importing 88-15
length
  intervisibility object 48-9
  simulation object name 13-3
  track history 43-6

LettersNumbers.ctxc 59-26
level, notification for messages 5-18
level observer 51-7
level-of-detail 51-19, 51-20, 90-30
  See also LOD
library, dynamic 4-33
license
  3D models 1-21
  DI-Guy 1-17
  GL Studio 1-20
  OpenSceneGraph 1-22
  osgEarth 1-22
  SilverLining 1-19
  SpeedTree 1-21
License Manager 2-6
license server, hostname 2-7
License Setup dialog box 2-7
lifeform 36-11
  appearance, random 69-13
  configuring DI-Guy appearance and character 73-3
  setting, posture 34-31
Lifeform Surrendered condition 36-11
lifetime
  radio communication line 43-27
  squawk indicator and communication line 43-27
light
  daylight control 45-7
  navigation 59-28
  point 45-7
  source 45-7
light characteristic attribute 59-30
light point
  CDB, properties 60-11
Light_DefaultValues_3_0.xml 60-18
Light_DefaultValues_3_2.xml 60-17
light_layer_list.csv 59-28
lighting
  disabling 45-19
  dynamic 45-2, 45-6
  effects 1-14, 1-15, 64-10
  high dynamic range 45-17
Lighting Render Settings page 45-3, 45-4, 45-5
lightmap, CDB 60-6
lightpoint, CDB 60-9, 60-10
Lights_xxx.xml 60-18
limitation, terrain 57-2
limited munition attack 22-9
line
  arrow, editing 40-23
  changing width 40-23
  creating
    displaying terrain profile 40-7
    opening terrain profile 56-3
  direction, changing 40-21
  dotted and dashed 40-23
  editing 40-23
    opening terrain profile 56-3
  fire 43-9
  fortified, constructing 30-18
  freehand 40-3
  height-above-terrain 16-20, 38-6
  intervisibility 38-2
    information 48-14
  radio communication 43-25
  sensor contact 43-16
  style 40-23
  supply 43-23
  suppressive fire at 25-30, 27-19
  task visualization 43-12, 43-13
  terrain profile 38-2, 56-5
    for intervisibility 48-15
  vertex, showing in terrain profile 56-3
  waiting in 28-15
Line Width dialog box 41-4
Line Width set data request 41-4
linear feature, list of 55-13
linear obstacle 38-2, 38-5
linear scaling 44-5
line-of-sight
  See also intervisibility
  communication line 43-27
  terrain profile 48-15
  unit versus entity 22-10
linked event, adding 8-6
Linux, installing Gamepad 80-15
lisp C-2
list 68-9
  clearing Categories, Used by Countries, and Favorites 68-10
  Countries 68-10
  image in script 31-20
  props 57-24
  spot report 9-8
  vector feature 55-13
list box, multiple 25-7
listing, components in OPD 74-16
LITCHR attribute 59-30

Little Pond terrain 55-21
load balancing 7-30
  scenario
    existing 7-30
    new 7-30
Load Batch File dialog box 7-48
Load Overlays dialog box 39-6
Load Scenario dialog box 7-25, 7-30
load-balanced scenario
  saving 7-23
  starting 7-32
load-entry parameter 78-6
load-entry-point parameter 78-6
Loader Settings page 64-17
loading
  cargo 26-27
  database 54-7
  display engine configuration 5-2, 82-5
  DTED 57-3
  entities on entities 17-2
  entity display settings 5-3
  environment configuration 5-3
  MetaFlight 58-5
  models 64-2
  multiple DTED files 59-7
  object type mapping 5-3
  plug-ins 4-33, 4-34, 5-3, 5-4, 5-6, 5-10, 5-19
  preventing 5-7
  saved views 51-34
  scenario 5-7, 5-10, 7-24, 7-25
    at startup 7-26
    recently used 7-26
  settings 3-13
  tactical graphics 39-6
  terrain 4-21, 64-2
    from command line 5-7
    paged 5-10
Loadout dialog box 35-11
Loadout set data request 35-11
loadPluginIfVersionMismatch parameter B-4
local
  object 3-9, 74-9
    network interface 74-12
  simulation object 3-9
Local Weather Zones page 11-13
locale, specifying 5-3
localization, specifying 5-3
localizing, GUI 2-11
local-objects parameter 74-9, C-4
localSettingsDesignator parameter B-7
location
  compass 51-5
  fixed-wing at creation 25-23
  jumping to 51-17
  moving to 26-15
  new entity 14-17
  object, changing 14-25, 40-18
  observer 51-3
  ordering entity to 26-15
  ordering simulation object to 26-15
  setting 14-3, 34-26
    by dragging 14-23
  suppressive fire at 25-30
  window 82-6
Location dialog box 34-26
location image attribute 57-8
Location set data request 34-26
Location3D class 32-19
locking
  GUI element 84-10
  overlay 39-4
LOD 51-19, 51-20, 90-30, 90-34
  configuring 64-7
  ocean elevation 57-17
  paged, scale factor 51-21
  scale, setting 45-10
  scaling, indirect rendering 64-15
  terrain, zooming in 51-22
  transition blending 64-15
LOD distance, buoyancy 46-8
log file 4-40, 5-11, 67-5, B-3
  messages sent to 5-18
  sending entity messages 16-26
  specifying 5-4
  vrfSim 5-11
logFrameRateStatistics parameter B-4
Logger
  recording to 7-50
  recording view control messages 37-37
Logger Control message 7-49
logger-files-path parameter 7-47
logical, operator 36-13
logistics 22-15
  resupply, starting or stopping 35-16
  resupply modes 35-15
logo, marking on entity 68-23
longitude 16-9, 32-19
Look At dialog box 34-26
Look At set data request 34-26

Lower Periscope task 29-17
Lua 31-3, 31-5
  background process 32-32
  classes 32-2
  direction vector 32-19
  interface 32-2
  locations and vectors 32-19
  method 32-2
  module 32-5
  object 32-2
    location of 32-19
  parameter 32-16
    as table 32-14
  require 32-5
  script, extended name 31-12
  table 32-14
  tasks and subtasks 32-14
  utility functions 32-5
  vector
    geocentric 32-22
    offset 32-22
lua
  global variables, saving 32-13
Lua API Documentation 32-2, 32-14
lubricant 22-15
  setting 35-9
lubricants, pacing and tracking 35-10
LWO file 54-3

M

-m command-line option 63-5
magnification, performance 51-22
mainmenu, element 84-2
mainmenuconfiguration, element 84-2
MAK Encrypted Data Format 63-7
MAK RTI, configuring 2-10
MAK Technologies LISP A-3
MAK_EARTH_FILE attribute 65-4
MAK_GEOID_GRID, environment variable 57-3
MAK_LAYER attribute 65-4
MAK_SOURCE_FILE attribute 65-4
making decisions 13-7
Makland terrain 55-22
MAKLMGRD_LICENSE_FILE, environment variable 2-8
Manage Reactive Task dialog box 25-16
Manage Reactive Tasks dialog box 19-10
managing, plug-ins 4-33
manual
  aggregation 23-11
  aggregation and disaggregation 23-12
  conventions lxii
  location of 2-4
map
  cube, recalculating 45-15
  texture 1-15, 64-10
map datum, configuring B-6
map scale overlay 55-17
Map Scale Toolbar 55-17
mapping
  joystick controls 80-7
  joystick keyboard controls 80-8
  keyboard 51-11
  keys 85-2
  model
    to object type 89-4
    to remote observer 50-12
  movement to keyboard 51-10
  object type 89-4
  objects, back-ends to 7-32
  shapefiles to features 57-11
  simulation objects 7-23
  soil types 64-20
  sound effects 49-4
marine
  conditions, setting 11-24
  effects, enabling and disabling 11-23
marker
  displaying 68-21
  label, displaying 68-21
  system 69-18
marker-scale attribute 90-32
marking
  changing 34-27
  remote simulation object 36-18
  visual 68-23
marking text 16-9, 68-7
Markings dialog box 34-27
mask
  adding to earth file 59-15
  element 59-15
matching object type 74-6
material, ambient value 45-4
Max Sheaf Radius 30-25
max-detonation-distance-for-passing-rounds parameter 25-32
maxDrainInputReads parameter B-8
maxDrainInputTime parameter B-4

Maximum Disaggregation Range 23-11
maximum sheaf parameter 71-12
maxNavigationRegenerationThreads parameter B-4
max-range parameter 27-21
max-range-altitude parameter 27-21
max-slope parameter 25-26
max-speed parameter 34-30
max-thrust parameter 25-26
mcastTtl parameter B-8
MEDF 63-7
  creating 90-35
  file 54-3
medfTool 63-7
media, adding to scenario event 8-5
MEIF, creating 90-35
memory 6-4
  improving use of 6-13
menu
  configuring 84-2
  context-sensitive 83-10
  locking 84-10
  separator 84-2
  set, editing 84-8
  Settings 3-13
  shortcut 84-7
  sorting, Task and Set 84-3
  Task 31-4
  tearing off 83-8
menuitem, element 84-7
merge-with-sub-systems parameter 76-5
merging
  model definition 87-12
  scenarios 7-14
mesh, navigation 40-12
message 5-11, 67-5
  console 5-11, 67-5
  entity 16-23
    saving 16-26
  frequency of B-3
  information and warning B-4
  notification level 5-5
  notification of 16-27
  object, viewing 16-25
  object console, sending 37-35
  radio 29-11, 29-13
  receiving 79-4
  sending 79-4
    between front-end and back-end 3-3
    text (command) 37-34
    text (task) 29-14
message 5-11, 67-5 (continued)
  sent 13-8
  session, configuring 4-17
  setting levels 5-18
  Start/Resume 7-50
  state update 50-11
  Stop/Freeze 7-50
  tactical data link 9-10
  TDL 9-10
  time stamp ordered, TSO 4-19
  view control 51-35
  warning icon 16-27
metadata
  in system definition file 74-17
  script 31-5
metafile, texture skinning 90-22, 90-23
metaFileName parameter 90-22, 90-23, 90-24
MetaFlight
  building efficient terrains 63-2
  loading 58-5
  processing 63-2
method, Lua 32-2
mftTool 63-4, 63-6
middle-click menu 83-10
migrating, SMS 67-20
MilesPerHour updater 90-13
Military Scenario Definition Language, importing 7-16
Military Symbol Icon, visual definition 90-3
MIL-STD 2525B color 16-19
  unit 24-10
Mimic Location Track mode 52-11
Mimic mode 51-8, 52-8, 52-10
Mimic Track mode 52-11
mimModule parameter B-7
minDrainInputTime parameter B-7
mine
  dropping 27-5, 27-6
  naval, arming 27-3
  sweeping for 26-28
minefield, creating 30-18
minimum lift speed 25-26
minimum tick period 6-15
minimum tick period variance 6-15
min-tick-period parameter 6-15
min-tick-period-variance parameter 6-15
mirror image attribute 57-8
missile 76-16
  anti-air 30-6

missile 76-16 (continued)
anti-ship 30-6
anti-submarine 27-13
ballistic 10-8, 27-14
firing 10-9
target event 10-11
cruise 27-10
fixed trajectory 10-8
land attack 30-6
laser guided 76-16
laser-guided 76-17
strategic 10-8
types 76-14
Missile Approach Warning condition 36-11
missile attack 22-9
missile controller 76-16, 76-17
missile engagement zone
altitude limited, creating 40-5
missile launcher 76-15
laser guided 76-16
missile launcher system 76-14
Missile Target, page 10-9, 10-11
missile target event 10-11
missile-collision actuator 76-16, 76-17
missile-entity-state-repository 74-12
missile-maneuver actuator 76-16, 76-17
missile-param parameter 74-12, C-8
Mission Oriented Protective Posture. See MOPP
mobility, modifier 22-11
mobility-kill parameter D-23
mode
Absolute 52-8
batch 7-45
C and 5 in IFF 34-21
Follow 52-9
jammer 34-23
Mimic 52-10
Mimic Track 52-11
observer 50-2
Space Follow 52-12
specifying 5-22
speed scaling 51-28
Tether 52-13
Tether Track 52-14
Track 52-15
view 51-8, 52-8
model
3D 14-11, 85-5
changing 68-14
DI-Guy 28-7
exaggerating entity 44-5
hiding 44-8
adding
as prop 90-13
visual 68-18
aggregate-level, new 71-2
animation 92-12
calculating support point from 69-5
colorized, changing 68-14
compressing 90-35
creating 90-34
decal effects 43-4
definition, buoy 59-23
DI-Guy, choosing 73-5
editing model mapping 89-5
element 59-8, 59-9, 90-32
entity 67-6
editing 67-3, 68-3
file caching 64-2
flipping DDS textures 90-29
illumination 11-5
instance, caching 6-13
instancing 5-4
layer, filtering 55-10
mapping
adding 89-4
common buttons 83-10
deleting 89-7
editing 89-5
to remote observer 50-12
name 16-9
object 67-6
physical 13-8
preloading 6-24
schema 87-2
set 50-11, 88-8
changing 50-11
showing 68-20
sharing resources 6-13
simulation 67-6
stealth 50-12
tactical graphic 68-33
texture 90-22
texture atlas 90-22
topological 3-13
trailing effects 43-4

model (continued)
vehicle 1-6
visual
editing 68-13
refreshing 68-25
model definition
adding 68-18, 68-20
buoy 59-22, 59-23
building from attributes 59-23
texture atlas 59-26
compass 51-5
copying 87-7
creating 87-2, 87-5
cockpit display 90-11
DefaultEllipsoidArc 91-2
deleting 87-6
DI-Guy character 87-5
editing 87-8
exporting 68-25, 87-13
filtering list of 87-8
importing 68-24, 87-12
importing from command line 87-12
invalid 87-10
jumping to element definition 87-11
jumping to from element definition 88-14
merging 87-12
parameter
adding 87-8
deleting 87-10
editing 87-10
prop, changing 57-26, 66-11
pruning 68-25
saving 87-13
schema 86-2
SpeedTree 87-5
tactical graphic texture 68-35
texture atlasing 90-22
wake 11-26
Model Definition Editor page 87-2, 87-5, 87-6, 87-7, 87-8, 87-10, 87-11, 87-12, 87-13, 90-9, 90-22, 90-29, 91-4, 91-6
model element 59-19
model instancing 6-13
cache, clearing 6-13
model mapping, jumping to element definition 89-7
Model Preload Configuration page 6-24
model set 50-11
changing 68-21
Model Specification for DIS Interoperability 92-2
Model_PagingDistance.xml 60-17
modeldefinitionschema, attribute 90-7
modeling
aggregate-level 13-9, 20-1
entity-level 13-9, 20-1
modifier 22-4
combat engineering object 22-11
mobility 22-11
visibility 22-12
vulnerability 22-11
modifiers 71-17
modifying
control object 40-17
sensor signature 40-11
module
exectool 32-5
fsm 32-5
location of 32-5
Lua 32-5
monitor, multiple 82-18
monitoring, entity resources 16-21
MOPP level 22-10
changing 30-10
setting 35-12
MOPP Level dialog box 35-12
morale, setting 35-12
Morale dialog box 35-12
motion, file 73-4
mouse
changing speed 51-29
drag terrain 51-14
drag view 51-16
hiding in full screen mode 83-9
locking to object 14-21
moving observer 51-14, 51-16, 51-19
orbiting terrain 51-16, 51-19
teleporting observer 51-17
Mouse Settings page 51-14, 51-19, 83-9
Move Along Route dialog box 26-12
Move Along Route Retrograde task 30-28
Move Along Route task 26-12
Move Between Cover to Location dialog box 28-12
Move Between Cover to Location task 28-11
Move Into Formation dialog box 26-13

Move Into Formation task 26-13
Move to Altitude dialog box 26-14
Move to Altitude task 26-14
Move to Depth dialog box 26-14
Move to Depth task 26-14
Move to Location (Plan Along Roads) task 25-21, 26-16
Move to Location dialog box 26-15
Move to Location Retrograde task 30-28
Move to Location task 26-15
  in plan 26-16
Move to Waypoint (Plan Along Roads) task 26-19
Move to Waypoint dialog box 26-18
Move to Waypoint Retrograde task 30-28
Move to Waypoint task 25-21, 26-18
movement
  DI-Guy 28-7
  lifeform 34-31
  model 69-15
    specifying 69-17
  retreat 30-28
  scripted 26-4
  speed, affect of posture 22-6
  speed scaling 51-27
  stopping 30-22
  system 69-16, 74-17
  task, affect of aggregation and disaggregation 23-10
  unit 21-4, 22-8
    restriction 65-9
movement file, configuring 69-29
movement parameter 71-19
movement system
  properties, editing 71-20
moving
  along a route 26-12
  articulated parts 29-16
  camera 51-13
  in formation 36-17
  intervisibility fan 48-9
  object 14-23
  objects 14-23, 34-26
    between overlays 40-6
  observer 51-3, 51-12
    mouse with 51-14, 51-16, 51-19
  options 51-12
  to altitude 26-14
  to location 26-15
  to points or simulation objects 26-18
moving (continued)
  vertex 40-19
moving object parameter C-8
moving-object-param parameter 74-12, C-8
moving-object-param parameter type C-8
MSDL, importing 7-14, 7-16, B-4
MTL file A-3
  vrfSim.mtl B-2
multicast address, configuring B-7
multicast mode, specifying 5-22
multicast time-to-live 5-22, B-8
multichannel, configuring 82-18
MultiGen-Paradigm, OpenFlight format 92-2
multimedia, adding to scenario event 8-5
multiple
  back-ends 3-2, 3-7, 12-9
  front-ends 3-2
    coordinating 3-6
  object, information 16-8
  parameter values 88-11
  simulation model set 7-8
  VR-Forces processes 3-2
multiple entity creation 19-5
  specifying entity type 19-9
multiple list box 25-7
Multiple Object Information window 16-8
munition, display of path and detonation 43-9
munition block 77-3, D-14
Munition Target Settings dialog box 10-3, 10-6, 10-9, 10-11
munition-type parameter 76-11, D-6
munition-type parameter D-31
mutually exclusive task 24-1, 25-4, 36-6
  configuring 25-33
mutually-exclusive-task-groups parameter 25-33
muzzle flashes 43-8

N

-N command-line option 5-7, 5-12
-n command-line option 5-5, 5-11, 5-18, 67-5
name 13-2
  changing
    simulation object 68-7
    tactical graphic 14-25
  changing overlay 39-5
  component 74-13

name 13-2 (continued)
control object 38-2
default 14-13
displaying 16-10
entity 16-9
extended 31-12
feature layer, specifying 65-2
FED file B-7
federate B-7
in condition 37-23, 37-24
layer 59-10
object 68-7
editing 14-26
setting 14-3
short 68-7
simulation object 13-3, 16-9, 68-7
slot 29-9, 72-2
spawn pattern 18-11
trigger 36-7
named query 65-3
nav edge 40-12
Naval Depth Charge Deployment, system 27-4
Naval Mine Deployment, system 27-5, 27-6
naval mine explosive device system 76-19, D-12
nav-edges 61-13
navigating, view 51-3
navigation
advanced 61-2
attached context 51-8
function, editing key map 85-3
key map 85-2
keyboard 85-2
light 59-28
streaming 59-28
mesh 40-12
options to 51-12
speed scaling 51-27
navigation area 61-7
cancelling generation of navigation data 61-15
configuration file 61-19
creating 61-8
disabling use of 34-27
displaying 61-11
editing 61-11
reverting 61-11
generating navigation data for 61-14
navigation area 61-7 (continued)
importing 61-19
information 61-12
nav edge 40-12
pedestrian path 19-10, 38-2
removing 61-11
sector 61-7
status 61-15
Navigation Areas page 61-8, 61-11, 61-14, 61-19
navigation data 61-2
displaying 61-20
generating 61-13, 61-14
canceling 61-15
removing from queue 61-15
generating at runtime 61-7
generating for entity 61-17
importing 61-19
profiles 61-13
regenerating 62-2, B-4, B-5
regenerating automatically 61-16
using 61-16
Navigation Enabled dialog box 34-27
Navigation Enabled set data request 34-27
Navigation Lab 61-20
debugging B-3
viewing simulation objects in 61-22
Navigation Preferences dialog box 34-28
Navigation Preferences set data request 34-28
navigation profile 62-2
navigational aid light 59-30
navigation-preference parameter 34-28
navigationProfiles.mtl 62-2
regenerating navigation data 61-16
navigationRegenerationDelay parameter B-4
NBC
area 22-9, 22-10
report 30-31
sending 30-31
near clipping plane 82-9
Near Far Clip Policy, attribute 82-10
net-interface-min-tick-period parameter 6-16, C-5, C-7
net-interface-min-tick-period-variance parameter 6-16, C-5, C-7
network
connecting to, automatically 5-2
vector 25-21

network interface C-5
  hierarchy 74-12
network-interface parameter C-5, C-7
new
  spawn pattern 18-11
  terrain 57-2
  window 4-13
New Scenario dialog box 7-44, 12-3
night vision 47-2
night-illumination parameter 75-15
no dialog box 5-17
node, switch 90-21, 90-23
NoExternalInput keyword 82-17
noise 47-4
  propulsion, RPM 9-25
non-VR-Forces, simulation object 36-18
normal map 1-15, 64-10
notification, icon 16-27
notification level 5-11, 67-5, B-4
  setting 5-18, 34-28
  setting for entity console 16-24
  specifying 5-5
Notify Level dialog box 34-28
Notify Level set data request 34-28
  tactical graphics 41-5
notifyLevel parameter B-4
Nuclear Attack dialog box 30-29
Nuclear Attack task 30-29
nuclear contamination 22-9, 22-10, 30-29
numbering, events 8-15
number-of-runs parameter 7-47
NVG 47-2

O

-O command-line option 5-6
OBJ, file 54-3
objec creation palette, changing icons 68-10
object
  adding
    to Props Palette 14-35, 87-3
    to selection group 15-9, 15-10
  adding to Order of Battle 20-3, 20-4, 20-5
  altitude, changing 14-25, 40-18
  attaching observer to 52-2
  changing
    line width 40-23
    name 14-25
object (continued)
  combat engineering 22-11
  concepts 3-8
  configuring, avoidance of 78-5
  control 38-2
  copying 14-28
  count 6-4, 16-28
  creating 14-3, 14-4, 14-7, 14-11, 14-13
    from OOB 20-6
    in plan 37-35
  creating new in Simulation Object Editor 68-26
  creation, mouse locking 14-21
  cultural feature 14-33, 69-3
  deleting 68-27
    from plan 37-35, 37-36
  deleting from simulation model set 68-27
  detaching observer from 52-6
  disembarking 34-16
  dragging 14-23
    to new location 34-26
  editing 40-17
  editing name 14-26
  embarking in Embarkation View 17-6
  enabling and disabling creation of 68-8
  entity enumeration, changing 68-5
  exaggerating models 44-5
  fill style 40-23
  filtering list of 15-6
  geometric 32-19
  heading, changing 14-25, 40-18
  hiding 9-2, 39-4
  ID 13-2, 13-4
  intelligence 8-8
  intervisibility 48-5
    pinning to simulation object 48-12
  label 13-4, 16-14, 38-3, 38-8
  list 12-10
  list of 67-26
  local 3-9
  location, changing 14-25, 40-18
  locking mouse to 14-21
  Lua 32-2
  messages, viewing 16-25
  model 67-6, 85-5
  moving 14-23
  moving to 26-18
  multiple, information 16-8
  name 68-7

object (continued)
object type, changing 68-5
orbiting 51-13
order, changing in OOB 20-7
overlay 38-2
palette 14-4
specfiying 68-8
parameter C-3
default value 12-10
parameters 74-3
editing 68-3
pasting 14-28, 14-29
altitude 14-30
placing 14-13
properties
setting 14-3
specifying 14-15
remapping back-ends 7-32
remote 3-9
removing from OOB 20-6
selecting 15-4, 68-3
for creation 14-4
in Objects List Panel 15-6
selecting in multiple list box 25-7
setting icon for icon list 68-11
showing and hiding 38-7
subcomponents 74-9
super-type 74-4
tag, changing 68-4
timeout threshold B-8
type 74-4
unselecting 15-7
vertex
adding 40-18
deleting 40-20
editing 40-19
views of 15-2
VR-Forces 3-9
object console 16-23
clearing 16-27
notification of messages 16-27
setting notify level 16-24
warning icon 16-27
Object Console Summary Panel 15-4, 16-25, 32-25, 51-24
answering questions 16-26
selecting simulation object 15-6
object creation palette, filtering 14-8
object geometry 14-34
adding 69-7
Object Group tag 14-32
object information
exporting 67-26
exporting and importing 67-26
importing 67-26
object information file 67-26, 67-31
object map file 6-1, 12-2, 12-9
missing back-ends 3-8
object parameter database 3-8, 67-6, 74-2
component list order 74-16
configuring, sensors 75-2
configuring path to B-3
pathname interpretation 74-40
reloading B-3
system definition file 74-17
object type
mapping 89-4
loading 5-3
published and matching 74-6
object type enumeration C-9
object type mapping
exporting 68-25
importing 68-24
jumping to element definition 89-7
pruning 68-25
Object Visible to Sensor At dialog box 8-9
objectConsoleNotifyLevel parameter 16-24
objectConsoleNotifyLevel parameter B-4
Object-Map parameter 12-5
objects 14-11, 14-13
Objects Console Summary Panel 16-3
Objects List Panel 15-2, 15-4, 15-6, 17-2, 17-6, 23-6
changing icons 68-10
Embarkation View 17-2
selecting object in 15-6
Objects List View
adding object to unit 23-7
removing object from unit 23-8
Objects View 15-2
object-type parameter C-3
OBJNAM, attribute 59-25
objTypeEnums.h header file C-9
observer 51-3
adding 50-6, 50-7
altitude, filtering simulation objects by 6-11

observer 51-3 (continued)
attaching
articulated part 52-5
to simulation object or prop 52-2
to simulation objects 52-7
attachment
in 2D 52-16
order 52-4
camera 51-13
changing 82-20
camera shake 51-25
centering on object from Information dialog box 51-24
centering on object from Last Click Object Panel 51-24
centering on object from Object Console Summary Panel 51-24
compass 51-4
coordinate system
2D 51-7
3D 51-7
detaching from object 52-6
dragging
terrain 51-14
view 51-16
heading 51-4
inset view, in 53-3
location and orientation 51-3
LOD, ocean 57-17
mode 50-2
changing 50-3
creating 50-4
deleting 50-5
editing 50-5
renaming 50-6
setting 3-16
switching projection 50-2
view control message 37-37
XR 50-10
movement 52-8
functions 51-6, 85-2
speed 51-29
moving
keyboard with 51-10
mouse with 51-14, 51-19
orbiting 51-13
orientation 51-16
rain and wave effects 46-2
removing 50-9
saving view 51-30
observer 51-3 (continued)
teleporting 51-17
view
attachment change 52-4
saving 51-30
transitioning between 51-31
visibility 11-6
ways to move 51-12
zooming 51-19
2D 51-21
3D 51-20
entity extents 51-23
zooming to extents 51-23
Observer Control Panel 51-27
Observer Information Panel 51-3
Observer Name 82-7
Observer Settings page 9-22, 11-23, 16-22, 43-3, 43-5, 43-8, 43-14, 43-18, 43-23, 44-7, 44-8, 45-7, 46-3, 47-4, 50-3, 50-4, 50-5, 50-6, 50-7, 50-9, 51-5, 51-14, 51-25, 51-26, 51-28, 51-29, 53-9, 53-14, 55-6, 55-11
Observer Views Panel 51-30
loading saved views 51-34
saving views 51-34
obstacle 38-2, 38-5
avoidance 13-11
breaching 30-9
configuring avoidance of 78-4
destroying 30-21
fill style 40-23
improving 30-24
linear 38-2, 38-5
report 30-32
sending 30-32
obstructed terrain, viewing in terrain profile 48-15
obstruction, configuring avoidance of 78-4
occlusion culling 5-2
occlusion query 64-19
occupancy-director-controller, role in embarkation 78-6
ocean 1-20
choppiness 11-25
configuration, Triton.config 51-23
depth 57-28
dynamic, enabling and disabling 11-23
effect 11-22
height 57-16

ocean 1-20 (continued)
height map
regenerating 57-19
resolution 57-17
texture size 57-17, 57-19
layer 57-14
adding 57-15, 57-16
planar reflection 45-8
configuring 45-10
performance 45-10
SpeedTrees 45-10
texture height 45-10
texture width 45-10
quality 11-27
swell 11-25
synchronized 5-6
zoom, display problems 51-23
Ocean Height Map Regeneration Threshold 57-19
ocean height map resolution 57-18
Ocean LOD Elevation 57-17, 82-16
Ocean Settings page 11-26, 11-27, 45-9, 45-10, 45-15, 46-4, 46-6, 57-18, 57-19
off road driving 25-21
offset
burst 18-9
UV 90-24
offset start time 18-13
OffsetX parameter 90-12
OffsetY parameter 90-12
OffsetZ parameter 90-12
OGR 65-7
driver 59-10
ogr2ogr 65-7
oil 22-15
pacing and tracking 35-10
setting 35-9
on-ground-tolerance parameter 25-27
OOB 20-2
OOB. See Order of Battle
opacity
2D label background 16-16
prop 57-25
setting 57-25
OPD, reloading B-3
OPE file 74-40, 76-2
adding state properties 74-27
Open Car Door dialog box 29-17
Open Cargo Door task 29-18
Open Door task 28-12
Open Model Set dialog box 67-7, 67-22
Open Window task 28-12
OpenFlight 85-5
converting texture data 64-20
disabling indirect rendering for 90-35
file 54-3
format 1-6, 92-2
OpenGL 64-17
opening
car door 29-10, 29-17
cargo door 29-18
Create Object dialog box 14-15
door 28-12
inset view 53-3
scenario 7-38
simulation model set 67-22
terrain 54-7
window 28-12
OpenSceneGraph 1-3, 1-19
environment variables for anaglyphic stereo 82-22
license 1-22
operator
boolean 36-13
comparison 36-12
logical 36-13
optimizing, performance 6-4
option
command-line 5-2
element 59-17
navigation 51-12
Orbat parameter 12-5
Orbit, set data request 34-29
orbit, observer 51-13
Orbit Object dialog box 26-21
Orbit Object task 26-21
Orbit task 26-21
orbiting
terrain, with mouse 51-16, 51-19
Order of Battle 20-2
adding simulation objects to 20-3, 20-4, 20-5
changing order of objects 20-7
creating units 20-5
exporting 20-7
object, editing 20-7
removing objects from 20-6
reorganizing 20-7

order of battle 6-1
  importing 20-8
  instantiating members in scenario 20-6
Order of Battle Palette 20-6
Order of Battle Panel 20-3
Order of Battle View 15-2
ordered speed, setting 34-30
Ordered Speed dialog box 34-30
Ordered Speed set data request 34-30
  fixed-wing for 34-30
Order-Of-Battle parameter 12-5
ordnance
  unexploded, destroying 30-20
organization
  entity of 13-5
  unit 21-3
orientation 51-3
  camera 82-20
  clamping 54-11
  displaying 16-10
  icon 44-4
  observer 51-3, 51-14, 51-16, 51-19, 52-8
  prop, changing 57-26
  specifying measurement unit 83-12
OSG file 54-3
OSG_CURL_PROXY, environment variable 58-5
OSG_CURL_PROXYPORT, environment variable 58-5
OSG_ICO, environment variable 6-14
OSG_MAX_PAGEDLOD, environment variable 64-7
osgEarth 1-19, 58-2, 59-3, 64-7
  adding terrain servers 58-4
  CDB 60-2
  driver 59-7
  license 1-22
osgEarth Boundary Generation Tool 59-15, 59-16
osgEarth Cache Tool 64-3
osgEarth Profile dialog box 64-7
osgearth_cache 1-18, 64-5
osgearth_cache tool 64-3
OSGEARTH_CACHE_PATH, environment variable 64-3
OSGEARTH_CURL_PROXYAUTH, environment variable 58-5
outline
  emitter volume segment 91-5
  segment 91-5
output, port group 74-14
Out-the-Window, observer mode 3-16
out-the-window view 82-18
overlay 1-13
  changing name 39-5
  concepts 38-2
  creating 39-3
  default
    adding 68-9
    specifying 68-8
  deleting 39-5
  developer debug 60-9
  editing, enabling and disabling 39-4
  export to shapefile 39-7
  file 12-2, 39-6, 40-8
  hiding 39-4
    by default 39-4
  hiding graphics on 38-7
  importing shapefile to 39-7
  introduction to 39-2
  label 39-4
  locking and unlocking 39-4
  map scale 55-17
  moving objects between 40-6
  object 38-2
    selecting 15-4
  setting 14-3
  tab 39-4
  text, adding 40-5
Overlay Layers dialog box 68-9
overlay object
  copying 14-28
  deleting 40-17
  disaggregation area 23-12
  moving, to different overlay 40-6
  pasting 14-28, 14-29
  spot report 9-8
Overlay parameter 12-5
Overlays tab 38-7
Overlays View 15-2
Override Position 17-5
overriding
  default scenario parameters 7-5
  plan 37-44

P

-P command-line option 5-9, 5-14, 5-21
-p command-line option 5-8, 5-13, 63-5

pacing
ammunition 35-4
equipment 35-8
resource 35-14
weapon 35-17
package, scripted task 31-39
page
Checkpoint Settings 7-36, 7-41, 7-42, 7-43
Cloud Cover Editor 11-17
Date and Time 7-12, 7-13, 11-4, 11-5
dialog box, configuring 84-5
DI-Guy Settings 6-26
Display Units Settings 16-20, 38-8
Earth Layers 55-10
Edit Existing Props 57-21
Elevation Color 55-3, 55-4, 55-5
Engineering Object Information 22-14
Entity Behavior Settings 23-6, 23-11, 23-12, 25-22
Entity Definition Editor 90-3, 90-4, 90-18, 91-6
Entity Display Settings 44-8
Environment Conditions Settings 90-20
Feature Settings 55-14
General Application Settings 4-14, 4-24
GUI Settings 4-8, 4-23, 7-28, 7-50, 68-12, 83-11, 84-3
High Dynamic Range Settings 45-12, 45-17
Indirect Fire 10-3, 10-6
Intersector Settings 64-16
Joystick Configuration 80-4, 80-5, 80-7, 80-8, 80-11, 80-15
Lighting Render Settings 45-3
Local Weather Zones 11-13
Missile Target 10-9, 10-11
Model Definition Editor 87-12, 90-9, 90-29
Model Preload Configuration 6-24
Mouse Settings 51-19, 83-9
Navigation Areas 61-8, 61-11, 61-14, 61-19
Observer Settings 9-22, 47-4, 50-6, 51-28
Radar Coverage Settings 9-23
Raster Maps 64-11
Render Settings 46-5, 53-15
Resource Information 16-21
Scenario Event Settings 8-16
page (continued)
Schema Editor 87-5
Sensor Settings 47-4
Shadow Settings 45-20, 45-23
Simulation Ocean Settings 57-28
Simulation Settings 16-18, 16-19, 37-11
Supplies 22-14
Symbol Decoration Settings 16-16
Tactical Data Link Settings 9-10, 9-11, 9-12
Tactical Graphics Display Settings 40-25, 41-2
Tactical Graphics Settings 40-24
Terrain Contents 57-13, 57-15, 64-3
Terrain Profile Settings 56-3
Thermoclines 11-29
Unit Display Settings 16-19
Weather 11-6, 11-7, 11-8, 11-9, 11-10, 11-11, 11-12, 11-24, 82-16
pagecollection, element 84-5
pagecollections, element 84-5
Paged LOD Magnification LOD Angle 51-22
Paged LOD Magnification Scale Factor 51-21
paged terrain
flipping DDS textures 64-17
loading 5-10
preprocessing 58-6
paging
terrain 54-8
manually 58-6
palette
Entity 14-3
object creation 14-4
changing icon 68-10
specifying 68-8
Order of Battle 20-6
Props 14-3
Tactical Graphics 14-4
panel
displaying and hiding 83-3
dockable 83-2
docking 83-2
floating 83-2
Frame Rate Monitor 6-6
Last Clicked Location 55-16
Object Console Summary 16-25
Terrain Information 64-22
undocking 83-2

panel (continued)
Visual Models 68-13
parachute 28-13
using Air Drop task 26-3
Parachute to Target task 28-13
parachuting 34-31
Parallel Pattern task 26-22
parameter
acceleration-to-rpm-factor 9-25
adding, to model definition 87-8
adding in Qt Designer 33-2
additional-algorithm-config 77-17
additionalDestinationAddresses B-7
additional-opds 12-6
additionalSystemScriptsDirectory B-2
addMulticastAddr B-7
aggregate-formation-controller 78-2
aggregate-level entity 71-17
aggregate-level unit 71-3
aggregate-object-param 74-12
aggregate-object-param 74-39
aggregate-param C-8
aggregateSpatialModelTickAsFastAsPossible-WhilePaused B-2
aggregateSpatialModelTickPeriod B-2
aggregateSpatialModelTickPeriodUsesRealTime B-2
aggregateSpatialModelTickVariance B-2
allowed-state-repositories-types D-26
all-tasks-exclusive 25-33
ammo D-6
angle-of-incidence 77-8, D-19, D-23
angleSegmentMax 91-5
appIdRange B-7
appNumber B-2
assertOnBlockingTerrainCalls 54-8, B-2
Attach Components to Remote Entities On 7-5
attack, editing 71-22
auto-reorganize 12-5
ballistic-missile-param 74-12
batchScenarioFileName B-2
bomb-param 74-12
can-be-radar-jammed 22-19
can-pivot 26-29
catastrophic-kill D-23
cgfDispactherReceivePort B-2
ChannelKeyword 90-12
commenting out A-3
comm-model-name 79-4
Component Attachment Table 7-6
parameter (continued)
component list order 74-16
Component-Attachment 12-5
Component-Attachment-Table 12-5
component-manager C-5, C-7
configName 90-22, 90-24
countryCodesMappingFile B-2, B-4
Create_Global_Dynamic_Terrain 12-7
Create_Global_Environment 12-7
create-script-id 31-27
cultural-feature-param 74-12, C-8
daemon B-2
datumShiftX B-3
datumShiftY B-3
datumShiftZ B-3
daylight-illumination 75-15
day-night-transition-time is 75-15
defaultHeartbeatThreshold B-7
defaultObjectTimeout B-8
DefaultObserverView 12-7
defaultParameterDatabasePath 12-8, B-3
defaultTerrainDatabasePath 12-8, B-3
defense, editing 71-23
delay 77-17
deleting, from model definition 87-10
destAddrString B-8
destroyFedExec B-7
destroy-script-id 31-27
detection-types 76-9, D-4
detonation-munition 77-17
detonation-on-destroy 77-17
deviceAddress B-8
diGuyAnimationsFile B-3
DIS B-7
disabled-by-suppression 25-32
disableParallelTick B-3
dispersal-angle 27-22
Display Terrain 7-6
disPort B-8
doNotUseConsole B-3
dr-allow-gui-overrides 6-20
dynamic-terrain-type 77-17
echelon-level 74-39
editing in Simulation Object Editor 69-6
egress-points 78-6
electronic warfare, editing 71-24
embarkation-slot-entry 78-6
embarkation-slots 78-6
enableLogFileTimestamps B-3
enableNavigationLabDebugging 61-12, B-3

parameter (continued)
entity-range 76-12, D-7
entity-type 76-12, D-7
entry C-2
entVrfDataTimeThreshold B-3
equipment, editing 71-25
execName B-7
exerciseId B-8
exercise-start--time 12-6
federateName B-7
federateType B-7
fedFileName B-7
Filename 90-12
firepower-kill D-23
fixed-wing-entity-param C-8
fomMapperInitData B-7
fomMapperLib B-7
forceParameterDbReload B-3
Frame Mode 7-7
Frame Time 7-7
frame-mode 4-20, 12-6
time management 4-20
frame-time 12-6
gamewareMemorySize B-3
gamewareQueryTimeBudget B-3
ground-vehicle-param 74-3
ground-vehicle-param C-8
guidance-mode 77-3, D-14
GuiObserverViews 12-7
GuiScenarioSettings 12-7
Gui-Terrain-Database 12-5
heartbeatVariance B-8
HiddenTerrainViews 12-7
HLA B-7
hostAddressString B-7, B-8
hostile-fire-only 25-32
human-param 74-12, C-8
ignore-sets 84-8
improve-script-id 31-27
influenceName 32-7
influenceParameters 32-7
influencer 32-7
ingress-points 78-6
initial-road-preference 34-28
instrumentPanelCategory 53-7
jamming-defense-factor 22-19
jam-strength-factor 22-20
lateral-clearance 78-4
leader-promotion-id 78-2
load-entry 78-6

parameter (continued)
load-entry-point 78-6
loadPluginIfVersionMismatch B-4
local-objects 74-9, C-4
localSettingsDesignator B-7
logFrameRateStatistics B-4
logger-files-path 7-47
Lua, table 32-14
max-detonation-distance-for-passing-rounds 25-32
maxDrainInputReads B-8
maxDrainInputTime B-4
maximum sheaf 71-12
maxNavigationRegenerationThreads B-4
max-range 27-21
max-range-altitude 27-21
max-slope 25-26
max-speed 34-30
max-thrust 25-26
mcastTtl B-8
merge-with-sub-systems 76-5
metaFileName 90-22, 90-23, 90-24
mimModule B-7
minDrainInputTime B-7
min-tick-period 6-15
min-tick-period-variance 6-15
missile-param C-8
mobility-kill D-23
model definition, editing 87-8, 87-10
movement 71-19
moving-object-param C-8
munition-type D-31
mutually-exclusive-task-groups 25-33
navigation-preference 34-28
navigationRegenerationDelay B-4
net-interface-min-tick-period 6-16, C-5, C-7
net-interface-min-tick-period-variance 6-16, C-5, C-7
network-interface C-5, C-7
night-illumination 75-15
notifyLevel B-4
number-of-runs 7-47
object 74-3, C-3
editing 68-3
objectConsoleNotifyLevel 16-24
objectConsoleNotifyLevel B-4
Object-Map 12-5
object-type C-3
OffsetX 90-12
OffsetY 90-12
OffsetZ 90-12

parameter (continued)
on-ground-tolerance 25-27
Orbat 12-5
Order-Of-Battle 12-5
Overlay 12-5
parameter-type 74-3, 74-8, 74-12, 74-39, C-2, C-3
passing-rounds-suppress 25-32
personnel, editing 71-25
Perspective 90-12
PerspectiveHeight 90-12
PerspectiveWidth 90-12
physical 71-18
Plan 12-5
plan-manager C-5, C-7
pluginsDirectory B-4
position-offset 78-2
power-plant-active 9-25
power-type 77-17
promotion-id 78-2
providing information to components 13-7
publish-transmitter 79-4
publish-transmitters 79-4
publishZeroVelocityWhenPaused B-4
Radio Button Choice 31-20, 33-6
rain-to-attenuation-coefficient 75-15
Random Number Seed 7-7
random-number-seed 12-6
range-determinant 76-12, D-7, D-23
range-from 77-4, D-15
range-list 76-12, D-7
recvBufferSize B-8
regenerateNavigationDataAtRuntime B-5
rejectMismatchedScenarioMessages B-5
relative-heading 17-9
relative-position 17-9
remote-objects 74-9, C-6
reportedHostAddressString B-5
reuseEntityIdentifiersOnScenarioLoad B-5
rotary-wing-entity-param C-8
RotDir 90-21
RotOffsetDeg 90-21
round-down 77-8, D-19
rprFomVersion B-7
RSOName 90-12
RTI_timeMgmt 4-20
RTI_useRtiExec 4-20
run-duration 7-47
run-duration-time 12-6
parameter (continued)
runInTimeManagementMode 4-20, B-7
scenario
file 12-4
setting 7-5
Scenario Name 7-5
scenario-data 12-7
ScenarioDescription 12-7
ScenarioExtentInformation 12-7
Scenario-Extras 12-5
scenarioFileName B-5
scenario-filename 7-47
scenario-name 12-6
Scenario-Scripts 12-5
schema
adding 86-5
deleting 86-6
editing 86-6
script 31-14
scripted task, getting list of 32-18
script-ids 31-27
SelectionGroups 12-5
sendBackendLogToNetwork B-5
sendBufferSize B-8
sendFedTime B-7
send-spot-reports-on-own-force 9-6
sensor, editing 71-21
separation-distance 25-10
separation-tolerance 25-10
sessionId B-5
setDeadReckoningAlgorithmToStaticOnPause B-5
setMainThreadToHighPriority B-5
setting simulation object 33-1
showLines 91-6
simMgmtPduProcessLevel B-5
sim-time 12-6
Simulation Model Sets 7-6
Simulation Object 31-20, 31-21, 33-7
Simulation Terrain 7-5
simulationModelSet B-6
Simulation-Model-Set-Files 12-6
simulation-run-duration 7-47
siteId B-6
slot 72-7
sms-filename 7-47
soil-list 13-12
specifying for task 25-6
speed-to-rpm-factor 9-25
spheroidSemiMajor B-6
spheroidSemiMinor B-6

parameter (continued)
standard-power-type 77-16
standard-terrain-damage-type 77-16
startInRunMode B-6
startMinimized B-6
state-repository C-4, C-6
state-repository-min-tick-period 6-16
state-repository-min-tick-period-variance 6-16
statusUpdateHeartbeat B-6
stopping-distance-factor 78-4
structure of entries C-2
subnetMask B-8
subsurface-entity-param C-8
sunrise-time 75-15
sunset-time 75-15
supplies, editing 71-25
suppressDataAvailableException B-6
suppression-degrades-sensor-performance-by 25-33
suppression-detonation-insult 25-32
suppression-detonation-range 25-32
suppression-insult-level 25-32
suppression-recovery-time 25-32
suppressSelfReflect B-8
surface-entity-param C-8
tactical-smoke-param 74-12
targetFrameRate B-6
targeting-capable 34-17
targeting-control 76-9, D-4
task, Lua 32-16
task-manager C-5, C-7
taskVisualizationPublicationMode B-6
terrain-check 25-29
Terrain-Database 12-5
terrainDatabase B-6
Time Multiplier 7-6
time-multiplier 12-5
tracer-type 76-11, D-6
underFireDistance 34-35
under-fire-parameter 36-11
underFireTime 34-35
useAbsoluteTimeStamps B-7, B-8
useAdvisories B-7
useAsyncIO B-8
useCustomMapDatum B-6
useDIGuy B-7
useDrawControlForTaskVisualizations B-6
useIpv6 B-8
use-logger-control 7-47
using remote entities as 36-18
visual definition, editing 88-11
parameter (continued)
vrf-base-object-param 74-39, C-3
wakeLength 11-26
warhead 76-11
warhead 77-3
warhead D-6
warhead D-14
WindSpeed 90-21
WindState 90-21
parameter property 74-26
adding 74-27
parameters, Combat 71-24
parameter-type, control object 74-39
parameter-type parameter 74-3, 74-8, 74-12, 74-39, C-2, C-3
paratrooper 26-3, 28-13
parent
disembarking from 17-8
element 88-9
element definition 88-2
parent, attribute 90-6
particle effect, sea spray 46-4
passenger
embarking as 29-10
slot 29-10
passing-rounds-suppress parameter 25-32
passive sonar 9-24
paste
special, altitude 14-30
Paste Special 14-29
pasting
appearance 14-29
objects 14-28, 14-29
performance and 14-28
plans 14-29
simulation object 14-29
patch, terrain 54-3
path
data 61-2
disabling use of navigation data 34-27
finding 61-4
following 61-5
planning 26-16, 26-19, 61-4
using road data 34-28
specifying for RTI 2-10
pathname
interpreting by object parameter database 74-40
scenario files 12-7
Patrol Between dialog box 26-23

Patrol Between task 26-23
Patrol Route dialog box 26-23
Patrol Route task 26-23
patrolling
  between objects 26-23
  route 26-23
pattern
  color 59-23
  editing condition 37-25
  in condition 37-23, 37-25
Pattern Hold (Location) dialog box 26-24
Pattern Hold (Location) task 26-24
Pattern Hold (Waypoint) dialog box 26-24
Pattern Hold (Waypoint) task 26-24
Pattern tab 37-25
pause
  scenario, using PDU 7-50
pause mode B-6
pausing
  scenario, automatically 7-7, 7-44
  simulation 7-36
PDU
  sending 50-11
  siman, configuring B-5
  Start/Freeze 7-50
  Start/Resume 7-50
  Stop/Freeze 7-50
  Underwater Acoustics 9-24
  view control 51-35
pedestrian, creating 19-2
pedestrian area 19-10
  adding entities to 19-3
  adding pedestrians 28-4
  creating 19-2
pedestrian path 26-12
  control object 19-10, 38-2
pedestrians, adding to crowd 28-4
pen style 40-23
Pen Style dialog box 41-5
Pen Style set data request 41-5
Percent Complete dialog box 35-12
Percent Complete set data request 35-12
performance 6-10, 6-11, 6-16, 51-20, 57-17
  affected by copy and paste 14-28
  alpha to coverage 6-17
  anti-aliasing 5-2
  asynchronous I/O 5-22, 6-14
  attached observer 52-7
  coloring draw calls 6-23
  compressing texture 64-6
performance 6-10, 6-11, 6-16, 51-20, 57-17 (continued)
  cube map calculation 45-15
  DI-Guy 6-26
  factors affecting 6-3
  frame rate 5-12, 6-22, 64-19
  function profiler 6-10
  graphics, depth 5-2
  HAT lines 16-20, 38-8
  improving 6-13
  indirect rendering 64-14
  instancing 5-4
  interest management 6-10
  load balancing 7-30
  main thread 5-11
  model, best practices 90-34
  network, tuning 6-19
  occlusion culling 5-2
  ocean planar reflections 45-10
  optimizing 6-3, 6-4
  preloading models 6-24
  setting application priority 5-4
  shadows 45-23
  statistics
    displaying 6-7
    VR-Vantage 6-9
  stencil bits 5-7
  streamed data 6-14
  terrain 57-2
  tuning
    network interface 6-16
    state repository 6-16
    visual for 3D application 6-19
  visual 6-24
  zooming in 51-22
Performance Settings dialog box 6-19, 6-20
Performance Settings page 6-19, 6-22
period, burst 18-9
periodic checkpointing
  disabling 7-41
  scheduling 7-41
periodic snapshot, enabling 7-43
periscope
  lowering 29-17
  raising 29-18
permanent intervisibility object 48-5
personnel
  air base, setting 35-13
  status 16-6

Personnel and Supplies set data request 35-13
Personnel dialog box 35-13
personnel parameter, editing 71-25
perspective, in display 82-14
Perspective parameter 90-12
perspective view
  model, view 68-21
PerspectiveHeight parameter 90-12
PerspectiveWidth parameter 90-12
phase line 38-2, 38-4
  Entity Left of Line 36-11
physical model 13-8
physical parameter 71-18
physicalWorldParams.mtl, file 75-11
pinning
  range rings 43-19
  sensor contact lines 43-15
  task visualization 43-13
  track history
    spot report 9-9
    TDL message 9-12
pitch 51-3
Pitch updater 90-13
pixel 90-34
Place IED dialog box 27-18
Place IED task 27-18
placing 14-13
  IED 27-18
  new entity 14-17
  props 14-13
  simulation objects 14-13
  tactical graphics 14-13
plan
  abandoning 24-1, 37-44
    confirmation prompt 25-5
    from a plan 37-44
  adding
    statement 37-4
    task 37-4
  altitude condition 36-9
  assigning to multiple simulation objects 37-41
  concepts 35-1
  condition, If 37-5
  conditional statement 36-3
  considerations for creating 37-3
  console message, sending 37-35
  copying 37-42
  creating object 37-35
plan (continued)
  custom 18-2, 18-10, 18-13
  default 18-2, 18-10, 18-13
  deleting
    object 37-35
    simulation object 37-36
    statement 37-12
  drag and drop statement 37-42
  editing 37-3
    statement in 37-11
  entity 36-2
    when aggregated 23-11
  file 6-1, 12-2
    editing 12-9
  Follow Entity task 36-17
  force hostility 9-15
  global 11-3, 36-2
    writing 37-27
  global command, adding 37-31
  global task, view control 37-37
  including scenario events 8-12
  individual 11-3
  inverting condition parameter 37-23
  issuing 37-36
  manager C-5
  Move to Location task 26-16
  name in condition 37-24
  pasting 14-29
  pattern in condition 37-25
  printing 37-12
  remote simulation object 36-18, 37-41
  restarting 37-43
  saving 37-12
  simulation time condition 36-13
  Tasked by Superior 36-17
  trigger 36-6
  unit, for 37-41
  variable 36-14
  viewing 36-15
  When statement 36-6
  While condition 36-8
  writing 37-3
Plan parameter 12-5
plan view 50-10
Plan View window 4-13
Plan window 36-15, 37-3, 37-4, 37-5, 37-6, 37-7, 37-12, 37-42
planar reflection 45-8
  SpeedTrees 45-10
plan-manager parameter C-5, C-7

planning, path 26-16, 26-19, 61-4
plant, avoiding 13-11
platform
  specifying 68-9
  update, applying 67-25
platform file 74-16
  variable binding
    adding 74-23
    adding to Simulation Object Editor 74-24
  variable bindings 74-22
platoon
  marking 68-23
  preconfigured 23-5
playing, scenario event 8-10
plug-in
  adding 4-36
  configuration, deleting 4-38
  DLL
    removing 4-37
    specifying 4-37
  list, viewing 4-39
  loading 4-33, 4-34, 5-3, 5-4, 5-6, 5-10, 5-19
  managing 4-33
  prevent loading 5-7
  version, checking 5-3
  XML file 4-36
    deleting 4-39
Plug-ins dialog box 4-33
Plug-ins Editor 4-33, 4-36, 4-37, 4-38, 4-39
pluginsDirectory parameter B-4
PNG 54-4
point 38-4
  civilian visit 19-10
  feature, list of 55-13
  moving to 26-18
  orbiting around 51-13
  patrolling between 26-23
  reversing 40-21
  strong, constructing 30-19
  suppressive fire at 25-30, 27-19
  terrain profile, sampling rate 56-3
point feature, in front-end and back-end 54-5
point-to-point communications
  configuring B-8
  specifying 5-22
point-to-point intervisibility 48-3
POL, status 16-6
polarized stereo 82-21
  configuring 82-23
policy
  Clamp Near Clip When Magnified 53-10, 82-10
  Distance To Attached 82-11
  Reduce Near Clip 82-10
  Reduce Z Fighting 82-10
polygon 38-5, 54-7, 90-34
  count 57-2
  not going through 51-27
  terrain, viewing 55-18
  underwater 57-28
port 13-7, 74-14
  default 5-21
  for scenario files B-2
  group 13-7, 74-14
  number B-8
    specifying at startup 5-21
  proxy 58-5
  specifying, at startup 5-21
position 51-3
  camera 82-20
  observer 51-14, 51-19, 52-8
  prop, changing 57-26
  window attribute 82-6
position-offset parameter 78-2
posture
  adding 71-27
  affect on footprint, sensing, sensor signature, speed 22-6
  changing 30-11, 35-13
  Deliberate-Attack 22-6
  Deliberate-Defense 22-6
  Hasty-Attack 22-6
  Hasty-Defense 22-6
  Reconnaissance 22-6
  Rout 22-6
  setting 34-31
  transition time, editing 71-21
  Travel 22-6
  unit 22-6
Posture dialog box 34-31, 35-13
Posture set data request 34-31, 35-13
power
  adding new 77-10, D-20
  combat 22-4
  determinant 77-4, D-15
  explosive 76-18, D-10
  laser 34-25

power-list block 77-3, D-14
power-plant-active parameter 9-25
power-type parameter 77-17
precedence, components in OPD 74-16
precipitation 11-6, 11-9
  intensity 11-12
  type 11-12
preconfigured unit 23-5
Prefer Roads 34-28
preferred movement mode 28-15
preferred road data 28-15
Preferred Targets dialog box 35-14
Preferred Trajectory Height dialog box 34-32
Preferred Trajectory Height set data request 34-32
preloading, model 6-24
Prepare Aircraft for QRA Status dialog box 30-30
Prepare QRA task 30-30
preprocessing
  terrain, paged 58-6
primary color, unit 24-10
primary simulation object 52-11, 52-14
printing
  plan 37-12
  window 4-14
priority, weather zone 11-13
prisoner-of-war 34-42
probability, damage D-18
probability table, direct and collateral damage 76-13, D-8
probability-coefficient list 76-12, D-7
process
  background 32-32
  object mappings to 12-9
  VR-Forces 3-2
process ID, front-end 5-10
processing
  MetaFlight 63-2
  streamed data 59-8
product
  technical support lxi
  upgrades lxi
profile, navigation data 61-13
profiler, function 6-10
projected terrain database 51-7
projection
  supported 1-13
  switching 2D to 3D 50-2
Projection Resize Policy 82-7
  configuring 82-11
Projection Units 82-7
promotion-id parameter 78-2
prompt
  confirmation, abandoning plan 25-5
  quit, disabling 4-24
  task confirmation, enabling or disabling 25-5
prone 34-31
prop 14-12, 54-5
  adding
    from feature layer 57-22
    to Props Palette 14-35, 87-3, 90-13
    to terrain 57-20
  attaching observer to 52-2, 52-7
  configuring 54-4
  creating 14-3, 14-4, 14-7, 14-11, 14-13, 14-35, 90-13
  exporting to shapefile 57-27
  extracting 57-20
    from terrain 54-4
  filtering attachment list 52-6
  heading, setting 14-20
  importing from shapefile 57-27
  model definition, changing 57-26, 66-11
  opacity 57-25
  opaque 57-25
  placing 14-13
  saving 14-35, 90-13
  transparent 57-25
  type 57-26
  viewing list of 57-24
property
  buttonGroup 33-6
  channel 82-7
  editing system 69-18
  object, setting 14-3
  parameter 74-26
  scenario, editing 7-26
  state 74-26
    user-defined 74-26
  tactical graphic, default 68-31
  window 82-6
Props Palette 14-3, 14-4
  adding objects to 14-35, 87-3, 90-13
  category, adding 90-14
propulsion noise 9-24, 9-25
Protest Along Line dialog box 28-14
Protest Along Line task 28-14

Protest Around Location dialog box 28-14
Protest Around Location task 28-14
protesting, crowd 28-14
Provide Suppressive Fire on Location task 25-30, 27-20
Provide Suppressive Fire task 25-30, 27-19
providing, supplies 22-17
proximity detonation 76-19, D-11
proximity fuse 34-14
proximity sensor 76-19, D-11
proxy
  connecting through 58-5
  port 58-5
pruning
  element definitions 68-25
  model definitions 68-25
  object type mappings 68-25
  visual models 68-25
published object type 74-6
publishing
  entity velocity when paused B-4
  radio trasmitter 79-4
  tactical graphics 40-8
publish-transmitter parameter 79-4
publish-transmitters parameter 79-4
publishZeroVelocityWhenPaused parameter B-4
pulse repetition rate 91-2

Q

-q command-line option 5-11
Qt Designer 33-2, 74-24
  adding design elements 33-8
  parameter, adding 33-2
  selecting for script dialog box use 33-2
  UI file, details 74-25
Qt Linguist 2-11
quadcopter 26-26, 26-27
quality, ocean 11-27
query
  feature 65-6
  named 65-3
question
  asking from script 32-25
  scripted task 16-26
queue
  releasing entity from 29-23
  waiting in 28-15
Quick Launch toolbar 8-10
Quick Reaction Alert, launching aircraft 30-27
quick reaction alert, preparing aircraft for 30-30
quit prompt, disabling 4-24
quitting
  plan 37-44
  retask procedure 25-6

R

-r command-line option 5-11, 63-5
radar 34-32
  affect of rain 75-15
  coverage 9-20
    configuring 9-23
  jamming 9-20, 22-18, 22-19
  sensor 34-17
radar coverage
  displaying 9-22
  enabling 9-22
Radar Coverage Settings page 9-23
radar mode
  search 34-32
  track 34-32
Radar Mode dialog box 34-32
Radar Mode set data request 34-32
radar sensor, sector blanking 34-36
Radar-Jamming-Strength-Receiving 22-19
radio 34-9, 34-14
  communication line 43-25
    color 43-28
    enabling and disabling 43-26
    lifetime 43-27
  configuring 79-4
  message
    set data request 29-11
    task 29-13
Radio Button Choice parameter 31-20, 33-6
Radio Communications Settings page 43-26, 43-27, 43-28
radio transmitter, publishing 79-4
radius
  emitter volume 91-3
  sheaf 30-25
rain 11-6, 11-9
  effect on radar 75-15
  intensity 11-12

rain 11-6, 11-9 (continued)
  splash effect 46-2
rain-to-attenuation-coefficient parameter 75-15
Raise Periscope task 29-18
random
  DI-Guy appearance 69-13
  simulation object 14-33
Random condition 36-12
Random Number Seed parameter 7-7
randomized simulation object 69-22
random-number-seed parameter 12-6
range
  focal 53-13
  ground clamping 54-11
  intervisibility line 48-14
  laser codes 9-18
  Simulation Time Scale Toolbar, changing 7-35
range ring 43-16
  configuring display of 43-19
  displaying for entity 43-19
  enabling and disabling 43-18
  pinning 43-19
  sensor 43-16
Range Rings Settings tab 43-19
range-coefficients determinant 77-5, D-16
range-determinant parameter 76-12, D-7, D-23
range-from parameter 77-4, D-15
range-list determinant 77-5, D-16
range-list parameter 76-12, D-7
raster driver, CDB 60-3
raster map
  display order 57-9
  displaying 57-6
Raster Maps page 57-6, 57-10, 64-11
rate
  climb, turn, and descent 26-9
  turn 26-8
ray
  crepuscular 45-13
  intervisibility 48-4
  sun 45-13
reactive task 19-10, 25-11, 29-25, 31-5, 32-7, 32-31
  canceling 25-17
  creating 31-33
  disabling 25-14
  enabling 25-13
  Visit Interest Points 19-10
reader writer key value pair 71-27
Real DB 1-21
real time
  checkpointing in 7-41
  running faster or slower than 3-12
recalling
  view 51-31
  view state 51-30
receive buffer size 5-9, 5-14, B-8
Receive Text Message condition 36-12
receiving
  radio messages 79-4
  supplies 22-17
recently used scenario 7-26
Reconnaissance, posture 22-6
recording, to Logger 7-50
recovering, embedded entity 29-19
recovery 34-10
recvBufferSize parameter B-8
redo, ellipse rotation 40-20
Reduce Near Clip, policy 82-10
Reduce Z Fighting, policy 82-10
Reduce Z Fighting Ratio, attribute 82-10
reference ellipsoid B-3, B-6
  configuring B-6
reflection
  map 1-15, 64-10
  planar 45-8
refresh rate 6-24
refreshing, visual models 68-25
Refuel from Tanker dialog box 30-30
Refuel from Tanker task 30-30
refueling, aircraft 30-30
refueling boom 29-16, 29-18
Refueling Mission dialog box 30-31
Refueling Mission task 30-31
regenerateNavigationDataAtRuntime parameter B-5
regenerating
  navigation data 62-2
  ocean height map 57-19
regenerating navigation data 61-16
regenerating spot report definition 68-16
region
  DDM 6-11
  update threshold 6-11
register, trigger 36-7
Register Trigger dialog box 37-9
registering, triggers 37-9
registrant_feature_list.csv 59-8
regulating, time 4-19
reinitializing, scenario 7-38

rejectMismatchedScenarioMessages parameter B-5
rejoining exercise 3-7
relative path 12-7
relative-heading parameter 17-9
relative-position parameter 17-9
Release Bomb on Laser Spot task 27-20
Release Bomb on Location task 27-21
Release Bomb on Target task 27-21
Release Next Entity From Queue task 29-23
reloading, visual models 68-25
remapping
  back-ends 3-8
  batch mode 7-48
  simuilation objects 7-23
remapping objects 7-32
remote
  attachment, key-names 81-3
  control 1-15
    of the observer 51-35
  entity, attaching components to 7-5, 81-2
  object 3-9, 74-9, C-6
    network interface 74-12
  simulation object 3-9, 25-3
    markings 36-18
    plan 37-41
    use in plans 36-18
  stealth, displaying model 50-12
remote graphics, keyword 82-17
Remote Sensor Receiver system 34-33
Remote Sensor set data request 34-33
remoteAttachment example scenario 81-2
remoteConfigurations directory 81-3
remote-configurations entry 81-3
remote-objects parameter 74-9, C-6
Remove Jamming Target dialog box 34-33
Remove Jamming Target set data request 34-33
removing
  assembly 71-13
  channel 82-5
  cloud cover 11-21
  force 67-33
  formation 70-10
  navigation area 61-11
  object from Favorites list 14-36
  object from OOB 20-6
  objects from selection group 15-10
  observer 50-9
removing (continued)
  simulation objectfrom unit 23-8
  subordinate 70-5
  system 69-18
  tag 67-27, 67-29
  triggers from plan 37-10
  view 51-32
  window 82-5
renaming
  formation 70-10
  observer mode 50-6
  overlay 39-5
  scenario 7-24
  selection group 15-10
  tag 67-27, 67-29
render bin 6-17
Render Settings page 6-9, 6-10, 6-18, 6-23, 45-14, 46-5, 53-15, 55-9, 60-9, 60-12, 64-12, 64-13, 64-15, 64-19, 82-12
rendering, indirect 64-14
Reorganize set data request 34-33
reorganizing
  concepts for units 21-4
  OOB members 20-7
  unit 34-33
    versus closing formation 21-5
repair 34-10
repeat image attribute 57-8
replace image attribute 57-7
replacing
  saved view list 51-34
  subordinate 70-6
replicating, settings and data 3-16
report
  NBC 30-31
  obstacle 30-32
reportedHostAddressString parameter B-5
repository C-4
require statement, Lua 32-5
Reserve Space 17-5
resizeable window 82-2
resizing
  boxes and text overlay objects 40-20
  windows 82-11
resolution
  ocean height map 57-18
  tree 90-30
resource
  information 16-4

resource (continued)
monitoring entity 16-21
pacing and tracking 35-14
specifying 34-34
status at creation 14-22
tracking unit 23-10
Resource condition 36-12
Resource Information, page 16-21
resource-item, adding 71-27
Resources dialog box 34-34
Resources Pacing/Tracking dialog box 35-14
Resources Pacing/Tracking set data request 35-14
Resources set data request 34-34
Restart Plan command 37-43
restarting, plan 37-43
Restore set data request 34-34
restoring
embedded entity parent 17-11
simulation object health and stores 34-34
unit 23-10, 34-34
view 51-31
restriction, unit movement 65-9
result, detonation 77-18
resuming, simulation 7-34
resupply 22-17, 34-10
modes 35-15
Resupply Mode dialog box 35-15
Resupply Mode set data request 35-15
retreat 30-28
retrograde movement 30-28
reusable software object 90-10
reuseEntityIdentifiersOnScenarioLoad parameter B-5
Reverse Direction 26-12
reversing, line direction 40-21
reverting
category changes 67-29
navigation area edits 61-11
window layout 83-6
revision, RPR FOM 5-8, 5-13
revolving, observer 51-13
rewinding
scenario 7-36, 7-38
scripted task 31-36
to a snapshot or checkpoint 7-36
RGB 54-4
RID file 2-10, 5-19
rid.mtl file 2-10, 5-19
right-click menu 83-10
configuring 84-7
rightclickmenu, element 84-7
ring, range 43-16
road
data, using in path planning 34-28
driving on 25-22
driving on left or right side 25-22
network 25-21
using route as 26-12
Role dialog box 35-15
Role set data request 35-15
roll 51-3
roll up, rule 71-14
Roll updater 90-13
rolling back to a snapshot 7-36
rolling up assembly 71-15
rope, sliding down 26-6
Rotary Wing Airlift Cargo to Location task 26-26
Rotary Wing Land dialog box 26-26
Rotary Wing Land task 26-26
Rotary Wing Retrieve Cargo task 26-27
Rotary Wing Unload Cargo task 26-27
rotary-wing, landing at airport 26-11
rotary-wing entity
altitude 26-8, 26-9
attacking with gun 27-3
heading 26-8, 26-9
holding pattern 26-24
landing 26-26
terrain avoidance 25-29
rotary-wing-entity-param parameter 74-12, C-8
rotary-wing-entity-state-repository 74-12
rotating, ellipse 40-20
rotation path, smoothing 6-21
RotDir parameter 90-21
RotOffsetDeg parameter 90-21
rotor wash 46-2
enabling and disabling 46-6
high quality 46-6
roughness type, mapped to soil type 13-12
round 10-2
round-down parameter 77-8, D-19
Rout, posture 22-6
route 38-2, 38-5
altitude
displaying 38-8
setting 14-19

route 38-2, 38-5 (continued)
computed, creating 40-9
extruded 40-5
following 26-12
patrolling 26-23
using as road 26-12
RPR FOM 5-20
revision 5-8, 5-13
specifying version B-7
version 5-8, 5-13
rprFomVersion parameter B-7
RSO 90-10
RSOName parameter 90-12
RTI 2-9
configuring for time management 4-20
definition of 2-9
RTI_CONFIG environment variable 2-10
RTI_timeMgmt parameter 4-20
RTI_useRtiExec parameter 4-20
rtiexec 4-20
rule, roll up 71-14
rules of engagement
pasting 14-29
setting 34-35
Rules of Engagement dialog box 34-35
Rules of Engagement set data request 34-35
run
scenario, using PDU 7-50
run mode B-6
starting back-end in 5-11
run-duration parameter 7-47
run-duration-time parameter 12-6
runInTimeManagementMode parameter 4-20, B-7
running 34-31
batch scenario 7-48
Boundary Generation Tool 59-16
simulation 7-34
runway 25-26
rwKeywordValuePairs.xml 71-27

S

-S command-line option 5-7, 5-8, 5-13, 5-14, 5-22, 52-7
-s command-line option 5-11, 5-18, 63-5
S-57
buoys and beacons 59-22
converting to shapefile 65-7
data 57-28
S-57 (continued)
signal sequence 59-30
s57_attr_map.csv 59-24, 59-27
s57_defaults.csv 59-27
s57_layer_list.csv 59-22, 59-27
s57_modeldef_by_name_map.csv 59-25
Sail Heading dialog box 26-28
Sail Heading task 26-28
salvo 10-2
sample, scenario 7-29
sampling rate
freehand line 40-3
intervisibility, configuring 48-10
San Luis Obispo terrain 55-22
satellite, setting orbit 34-29
Save Scenario dialog box 7-22
saved views file 6-1
scenario 12-2
saving
checkpoint 7-39
display engine configuration 82-5
element definition 88-15
global variables 32-13
hierarchy and leaves 88-19
load-balanced scenario 7-23
model definition 87-13
observer view, individual 51-34
plan 37-12
prop 14-35, 90-13
scenario 3-7, 7-21, 7-38
existing 7-22
under new name 7-24
schemas 86-2
screen capture 4-14
scripts 31-36
settings 3-13, 3-15
tactical graphics 39-6
terrain 57-3
view 51-30, 51-31
to file 51-34
window layout 83-3, 83-6
scale
factor, material ambient and diffuse value 45-4
SpeedTree, random 90-31
terrain 55-17
Scaled Speed check box 51-28
scaling
3D models 44-5
icon, 2D 44-3

scaling (continued)
LOD distance, indirect rendering 64-15
model, linear and XR 44-5
scenario 6-1
batch 7-46
building blocks 7-14
checkpoint 7-39
checkpoints and snapshots 7-38
closing 7-37
collaborative development 7-14
component attachment table 7-6
compressed format 6-1, 12-2
creating new 7-3
creating objects from OOB 20-6
description, editing 7-29
end time 7-7, 7-44
new scenario 7-44
file 3-7, 6-1, 12-2
editing 12-3
parameters 12-4
pathnames 12-7
specifying which to use B-5
format, saving to different 7-24
frame mode 7-7
frame time 7-7
importing 7-14
information, displaying 7-26
load balancing existing 7-30
load balancing new 7-30
load-balanced
saving 7-23
starting 7-32
loading 5-7, 5-10, 7-24, 7-25
at startup 5-10
recently used 7-26
merging 7-14
MSDL, importing 7-16
name 7-5
parameter
setting 7-5
version 12-7
properties, editing 7-26
random number seed 7-7
reinitializing 7-38
remoteAttachment 81-2
renaming 7-24
reopening 7-38
resuming 7-34
rewinding 7-36, 7-38
scripted task 31-36
scenario 6-1 (continued)
rolling back to snapshot or checkpoint 7-36
running 7-34
sample 7-29
saving 3-7, 7-21, 7-38
existing 7-22
under new name 7-24
scenario objects 12-10
simulation model set 7-6
SMS, multiple 7-8
starting 7-34
starting date and time 7-12
startup observer view 51-33
temporary directory 12-11
terrain 12-8
display 7-6
simulation 7-5
time multiplier 7-6
zoom to extents on load 51-23
scenario event 8-2
adding audio, video, and graphics 8-5
creating 8-3
deleting 8-15
enabling and disabling 8-14
ending 8-13
intelligence object 8-8
sending 8-10
sending from plan 8-12
Scenario Event condition 36-12
Scenario Event dialog box 8-12
Scenario Event Manager 8-3, 8-8, 8-10, 8-13, 8-14, 8-15
Scenario Event Settings page 8-16
Scenario Event Toolbar 8-2
scenario extras file 12-2, 18-10
Scenario Name parameter 7-5
scenario objects 12-10
file 12-10, 39-6
default values in 12-10
scenario objects file 12-2
Scenario Properties dialog box 7-29
scenario script 31-3
Scenario Snapshots dialog box 7-36, 7-40, 7-42, 7-43
Scenario Startup dialog box 4-6, 4-8
Scenario Toolbar 7-42
scenario-data parameter 12-7
ScenarioDescription parameter 12-7
ScenarioExtentInformation parameter 12-7

Scenario-Extras parameter 12-5
scenarioFileName parameter B-5
scenario-filename parameter 7-47
scenario-name parameter 12-6
Scenario-Scripts parameter 12-5
scenario-specific
  GUI settings file 12-2
  spawn pattern 18-10
Scene Graph tab 64-22
scheduling
  periodic checkpoint 7-41
  spawn event 18-7
schema 86-2
  copying 86-4
  creating 86-3
  deleting 86-4
  model definition 86-2
  parameter
    adding 86-5
    deleting 86-6
    editing 86-6
  saving 86-2
  tactical graphic model 68-33
  visual definition 86-2
Schema Editor page 86-3, 86-4, 86-5, 86-6, 87-5
SciTE 31-4
scnx, file 6-1, 12-2
screen, capturing image 4-14
screen capture
  saving 4-14
  setting save location 4-14
Screen Number attribute 82-6
script
  adding to task toolbar 31-29
  availability, by simulation object type or behavior set 31-25
  creating 31-6
  deleting 31-41
  description 31-13
  extended name 31-12
  frame rate 6-6
  ID 31-27
    editing 31-11
  image list 31-20
  laying out using Designer 33-2
  location, object 32-19
  making available by behavior set 31-27
  metadata 31-5
  parameter types 31-14
script (continued)
  previewing dialog box 31-10
  saving 31-36
  script, editing 31-44
  script ID 31-8, 31-11
  set data request 31-3
  simulation object type supported 31-26
  system and scenario 31-3
  Task menu location 31-23
Script Settings page 6-6, 31-22, 31-44
scripted
  behavior 26-4
  movement, configuring 69-24
scripted set 31-5
  accessing state properties 74-29
  creating 31-31
scripted task 31-3, 31-4
  accessing state properties 74-29
  actioncategory 25-34
  adding to folder 31-38
  adding to toolbar 84-9
  Affects Controls check box 31-21
  background process 32-32
  concurrent 31-31
  copying 31-41
  editing 31-37
  entry point 32-6
  execution flow 32-8
  exporting and importing 31-39
  filtering list 31-37
  folder
    adding 31-38
    deleting 31-38
    renaming 31-38
  importing 31-40
  in system definition 31-27
  interactive 32-25
  language 31-30
  organizing 31-38
  package 31-39
  parameters, getting list of 32-18
  question 16-26
  reactive task 25-11, 32-7, 32-31
  removing from folder 31-38
  rewinding 31-36
  script ID 32-14
  script storage location 31-39
  system
    creating 31-42

scripted task 31-3, 31-4 (continued)
system
making available on Task menu 31-42
storage location 31-42
Update_EW_Degradation 22-19
script-ids parameter 31-27
scripts file 12-2
scriptstext editor, specifying 31-44
sea
choppiness 11-6
spray 46-2
enabling and disabling 46-4
state 11-6, 11-25
Douglas 11-22
swell 11-6, 11-25
sea spray, size 46-5
search, radar mode 34-32
search and rescue 26-22, 29-20
seasonal buoys and beacons 59-29
secondary
simulation object 52-11, 52-14
window 4-13
sector
navigation area 61-7
unit 22-5, 43-16
Sector Blanking set data request 34-36
sector of responsibility, setting 34-37
Sector of Responsibility dialog box 34-37
Sector of Responsibility set data request 34-37
Sector Search Operation dialog box 29-20
Sector Search Operation task 29-20
segment
angle, setting 91-4
arc size 91-4
edge 91-5
emitter volume 91-4
outline, displaying 91-5
Selected Simulation Engine Toolbar 14-6, 14-7
selecting
database for front-end 7-3
entity, from Object Console Summary Panel 15-6
object 15-4
in Objects List Panel 15-6
multiple list box 25-7
to create 14-4
objects 68-3
selecting (continued)
observer 50-6
selection group 15-9
simulation objects 15-5
unit 23-6, 24-2
selection box 15-5
selection group 19-5
adding object to new 15-9
adding to 15-10
creating 15-8
creating from keyboard 15-9
creating from menu 15-9
deleting 15-10
file, scenario 12-2
introduction 15-8
removing object from 15-10
renaming 15-10
selecting 15-9
Selection Group View 15-2
SelectionGroups parameter 12-5
self, as condition parameter 37-23
self reflection, DIS 5-14
send buffer size 5-9, 5-14, B-8
Send NBC Report dialog box 30-31
Send Obstacle Report dialog box 30-32
Send Radio Set task 29-11
Send Radio Task task 29-13
Send Text Message command 37-34
Send Text Message task 29-14
sendBackendLogToNetwork parameter B-5
sendBufferSize parameter B-8
sendFedTime parameter B-7
sending
console message 37-35
data, configuring B-3
messages 13-8
NBC report 30-31
obstacle report 30-32
radio messages 79-4
scenario events 8-10
from plan 8-12
set data request via radio message 29-11
task via radio message 29-13
text message (command) 37-34
text message (task) 29-14
view control, in plan 37-37
send-spot-reports-on-own-force parameter 9-6
SenSim 47-2
sensing, electronic emission 22-18

sensing target 76-9, D-4
sensor 9-16, 13-7, 75-2, 76-9, D-4
  See also. sensor signature
  adding new 75-11
  affect of posture 22-6
  aiming 34-38
  apparent signature 75-9
  attaching to remote entity 81-2
  channel attribute 82-7
  configuring 75-2
  connecting 75-3
  detection probability 75-9
  detection table 75-16
  detection types 75-11
  emission 22-20
  footprint 22-10
  gimbaled 34-38, 34-40, 53-8
    field of view 53-9
    field of view color 34-39
    setting azimuth 53-12
  information 16-4
  mode, full color 47-5
  parameter, editing 71-21
  proximity 76-19, D-11
  radar 34-17
    sector blanking 34-36
  range ring 43-16
  sensor contact line 43-14
  SIGINT 22-20
  signature, apparent 75-16
  signature-object-sensor 75-2
  system 69-15
  system definition file 74-17
    editing 75-4
  unit 22-10
  view 53-8, 53-10
  visual, illumination 75-14
  zooming 34-40
Sensor Aim set data request 34-38
sensor component, adding 75-13
sensor contact line
  configuring 43-16
  enabling 43-14
  enabling and disabling 43-14
  specifying objects that can display 43-15
sensor contact line pinning 43-15
Sensor Control Panel 53-10
sensor effects 47-2, 47-3
  blur, noise, and gain 47-4
  configuring 47-4
Sensor Enabled dialog box 34-38
Sensor Enabled set data request 34-38
Sensor Field of View Color dialog box 34-39
Sensor Field of View Color set data request 34-39
Sensor Settings page 47-4, 47-5
sensor signature 9-24, 69-16, 75-5
  adding 75-14
  affect of posture 22-6
  modifying 40-11
  predefined 69-14
  setting 69-14
  standard 75-6
sensor view 53-8
Sensor Zoom set data request 34-40
sensor-signatures.csv 69-14, 75-6
separation-distance parameter 25-10
separation-tolerance parameter 25-10
separator, menu 84-2
server
  terrain 54-7, 54-9
    caching 64-3
    cutting in site 59-15
session 3-6
  ID 4-3, 5-8, 5-9
    assigning 4-9
    configuring B-5
    setting at startup 5-10
  joining at startup 4-17
  message, configuring 4-17
  starting and stopping 7-50
sessionId parameter B-5
set
  scripted 31-5
    creating 31-31
Set Ammunition 35-3
Set Appearance 45-6
Set command, default 33-1
set data request 33-1
  Activation 41-2
  Active Sonar Mode 34-4
  Add Jamming Target 34-4
  AI Enabled 34-5
  Aiming 34-6
  Allow Crossing set 41-3
  Altitude 25-24, 34-7
  Ammunition Pacing/Tracking 35-4
  Appearance 34-8
  Armed 34-9

set data request 33-1 (continued)
Arrowhead Style 41-3
Assign Aircraft 35-4
Assume Unknown is Hostile 34-9
Attention Grabbing 34-9
Automatically Close Doors 34-10
Capabilities 34-10
Collision Avoidance Types 34-11
Color 41-4
Concealed 34-12
Contamination 35-6
Counter Measures Auto Launch 9-26, 34-12
Current Speed 34-13
Destroyed 34-13
Detonation Fuse Type 34-14
DI-Guy Characteristics 34-15
DI-Guy Hand Item 34-15
Disembarked 17-7, 34-16
Embarked 17-5, 34-16
Emitter 30-4, 30-19, 34-17
Engagement Results 35-7
Engagement Zone 34-18
Equipment Pacing/Tracking 35-8
Facilities 35-8
Fill Style 41-4
Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking 35-10
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant 35-9
Force 34-18, 41-4
Formation 34-19
in plan 36-17
Has Drag Chute 35-10
Heading 34-20
Hide Occupants 34-20
IFF 34-21
Invisible 34-22
Invulnerable 34-22
Jammer State 34-22
Jamming Mode 34-23
Lase Autonomous 9-19, 34-24
Laser Code 9-18, 9-19, 34-24
Laser Power and Wavelength 34-25
Line Width 41-4
Loadout 35-11
Location 34-26
Look At 34-26
Navigation Enabled 34-27
set data request 33-1 (continued)
Navigation Preferences 34-28
Notify Level 34-28
tactical graphic 41-5
Orbit 34-29
Ordered Speed 34-30
Pen Style 41-5
Percent Complete 35-12
Personnel and Supplies 35-13
Posture 34-31, 35-13
Preferred Trajectory Height 34-32
Radar mode 34-32
Remote Sensor 34-33
Remove Jamming Target 34-33
Reorganize 34-33
Resources 34-34
Resources Pacing/Tracking 35-14
Restore 34-34
Resupply Mode 35-15
Role 35-15
Rules of Engagement 34-35
script 31-3
Sector Blanking 34-36
Sector of Responsibility 34-37
sending by radio 29-11
Sensor Aim 34-38
Sensor Enabled set 34-38
Sensor Field of View Color 34-39
Sensor Zoom 34-40
Sonar Depth 34-40
Spot Reports 34-41
Stow Hand Item 34-42
Superior 34-42
Supplying 35-16
Surrendered 34-42
Synchronize Laser Code 9-19, 34-43
tactical graphics 41-2
Target 34-43
Tasked by Superior 25-11, 34-44
TDL Messaging 34-44
Tracer Use 34-44
Unit State 34-45
Weapons Pacing/Tracking 35-17
Set menu
editing 84-8
icon 31-12
sorting 84-3
Set MOPP Level 35-12
Set Morale 35-12

set statement 33-1
  See also. set data request
Set toolbar, adding to 84-9
setCheckpointMode function 32-13
setDeadReckoningAlgorithmToStaticOnPause parameter B-5
setMainThreadToHighPriority parameter B-5
setting
  altitude 14-18, 34-7
    dialog box 14-18
    manually 14-18
    route 14-19
  altitude for fixed-wing entity 25-24
  ambient air temperature 11-6, 11-10
  appearance 34-8
  autonomous lasing 34-24
  capabilities 34-10
  collision avoidance types 34-11
  color, unit bounding volume 24-10
  combat engineering object percent complete 35-12
  concealment 34-12
  date 11-4
  emitter 34-17
  emitter volume radius 91-3
  factory and default 3-13
  formation 34-19
  frame rate 5-3
  heading 14-3, 34-20
    manually 14-20
  icons 3-13
  IFF 34-21
  laser code 34-24
  loading 3-13
  location 14-3, 34-26
  LOD scale 45-10
  MOPP level 35-12
  morale 35-12
  name 14-3
  notification level 5-18, 34-28
    entity 16-24
  ordered speed 34-30
  overlay 14-3
  posture 34-31
  precipitation
    intensity 11-12
    type 11-12
  prop opacity 57-25
  properties, object 14-3
  resources 35-9
setting (continued)
  rules of engagement 34-35
  saving 3-13
  sector of responsibility 34-37
  segment arc size 91-4
  simulation object 33-1
    parameters 33-1
    to be destroyed 34-13
  simulation speed with time management 7-34
  superior 34-42
  support point 69-5
  target 34-43
  time of day 11-5
  time zone 11-5
  user 3-15
  visibility 11-11
  weather 11-9
settings
  GUI, saving 83-3
  GUI layout 5-4
  property, default 68-31
  scenario GUI 12-2
  Simulation Object Editor, GUI layout 67-5
  synchronizing 3-16
Settings menu 3-13
shader 1-15, 64-10
  debugging 64-12
shadow
  configuring 45-23
  enabling and disabling 45-20
  SpeedTree, enabling and disabling 45-22
  text, 2D label 16-16
Shadow Settings page 45-20, 45-23
shake, camera 51-25
shaking camera 51-25
shape terrain data 54-3
shapefile
  buoys and beacons 59-22
  converting S-57 to 65-7
  exporting overlays to 39-7
  exporting props to 57-27
  importing props from 57-27
  importing SpeedTrees from 57-27
  importing to overlay 39-7
  layer name 59-10
  mapping to feature 57-11
sharing, resources 6-13

sheaf radius 30-25
ship
  wake, configuring 90-18
shooter, laser 9-18
shooter-and-impact-point range 77-5, D-16
shooter-to-impact range 77-5, D-16
short name 68-7
shortcut menu 83-10, 84-7
Show Function Profiler Overlay 6-10
Show Ground Truth For 9-5
Show Occupants 29-10
Show Occupants check box 44-8
Show Only Active Environmentals 40-24
Show Performance Statistics Overlay check box 6-9
Show Session Terrain Change Prompts 4-18
Show Spot Reports from Viewpoint of Force 9-4, 9-5
Show Spot Reports Sent By 9-5
showCockpits keyword 82-17
showHuds 82-7
showing
  objects 38-6, 38-7
  symbol decoration 16-15
  vertex labels 38-8
showLines parameter 91-6
showRemoteGraphics keyword 82-17
SHP, file 54-3
SIGGRP, attribute 59-30
SIGINT sensor 22-20
signal
  interaction 43-25
  sequence 59-30
signal group attribute 59-30
signal period attribute 59-30
signal_slot 74-25
signature
  apparent 75-9, 75-16
  sensor 9-24, 75-5
signature propagator, configuring 75-11
signature-object-sensor 75-2
SignificantSize_SwitchDistance.xml 60-17
SIGPER, attribute 59-30
SIGSEQ attribute 59-30
SilverLining 1-19
  license 1-19
simMgmtPduProcessLevel parameter B-5
SimObject class 32-3
simple driver 59-19
simple-radio-comm-model 13-6
Simthetiq 1-21
sim-time parameter 12-6
simulated world, navigating 51-3
simulation
  closing 7-37
  date 7-12
  engine 3-2
  objects 6-4
  pausing 7-36
  resuming 7-34
  rewinding 7-36
  running 7-34
    automatically 7-45
  speed, with time management 7-34
  start mode B-6
  starting 7-34
  time 3-10, 4-19, 7-12
    checkpointing in 7-41
    evaluating in plan 36-13
Simulation Connections Configuration dialog box 4-33
Simulation Control Toolbar 4-11, 7-34, 7-36, 7-37
Simulation Date/Time condition 36-13
simulation engine
  closing 4-23
  exiting 4-23
  remapping objects to 7-32
  specifying 14-6, 14-7
simulation management message, configuring B-5
simulation model set 67-6
  AggregateLevel.sms 22-3
  configuration 7-9
    default 7-11
  creating 67-23
  default 5-2
  deleting entity 68-27
  entity-level and aggregate-level 13-9
  file 67-6
  hostility file 9-13
  multiple 7-8
  new, creating 67-23
  opening 67-22
  platform file 74-16
  specifying on command-line 5-11

simulation model set configuration, default 7-9
Simulation Model Sets parameter 7-6
simulation object
See also entity and unit
adding to
selection group 15-10
unit 23-7
adding to Order of Battle 20-3, 20-4, 20-5
assigning plan to multiple 37-41
attaching observer to 52-8
attaching tactical graphic to 40-16
attribute, information 16-4
automatically attaching tactical graphic to 40-16
behavior 13-8
centering observer on 51-24
changing
name 68-7
undoing changes 68-12
changing tag 68-4
collision and obstacle avoidance 13-11
communication model 13-6
components 74-13
concepts 13-2
console, setting notification level 34-28
console message 16-23
saving 16-26
sending 37-35
viewing 16-25
copying 14-28
count 16-28
creating 14-3, 14-4, 14-7, 14-13
from existing 68-27
in plan 37-35
multiple 19-5
creating from OOB 20-6
creating multiple 19-2
creation, enabling and disabling 68-8
deleting 14-32
in plan 37-36
deleting in plan 37-35
destroyed 34-13
dragging 14-23
to new location 34-26
editing 14-25, 67-3
element definition 88-2
enumeration, published and matching 74-6
simulation object (continued)
filtering attachment list 52-6
filtering list of 25-9
filtering using interest management 6-10
firing at target 9-16
heading 34-20
setting 14-20
hiding 9-2
hiding list of in Simulation Object Editor 67-13
hostility 9-13
icon 44-3
CID level and color 9-16
image 90-9
identifiers 13-2
information 16-3
multiple 16-8
inset view 53-2
intervisibility fan 48-4
label 13-4, 16-14
extended 16-17
list, spot report 9-8
list of 67-26
local and remote 3-9
model 67-6
editing 68-3
reloading 68-25
move to location 26-15
name 16-9, 68-7
length restrictions 13-3
pasting 14-29
patrol between 26-23
patrol route 26-23
pinning intervisibility object to 48-12
placing 14-13
plan
issuing to 37-36
viewing 36-15
platform, updating 67-25
properties, specifying at creation 14-15
random creation of 14-33
remote
in task 25-3
plans 37-41
removing, from unit 23-8
resource, monitoring 16-21
resource at creation 14-22
restoring health and stores 34-34

simulation object (continued)
selecting 15-4, 15-5
from Object Console Summary Panel 15-6
in Objects List Panel 15-6
setting
appearance 34-8
location 34-26
state and parameters 33-1
setting icon for icon list 68-11
specifying resource 34-34
spot report, enabling or disabling 34-41
state 13-8, 33-1
storing list of 12-10
systems
adding 69-16
editing 69-15
target detection 9-16
task 13-8
tasking by superior 34-44
tick rate, tuning 6-15
track history 43-5
trailing effect 43-3
unselecting 15-7
UUID 13-2, 13-3
viewing in Navigation Lab 61-22
views of 15-2
waiting 29-15
ways to add to scenario 7-14
writing plan 37-3
zooming to extents 51-23
Simulation Object Editor 1-18, 67-3
command-line arguments 67-4
configuring DI-Guy character and appearance 69-11
configuring systems 67-3
embedded entity 17-9
force
adding 67-31
editing 67-33
removing 67-33
formation, bounding box 70-11
generating navigation data for entity 61-17
managing forces 67-31
simulation object group 68-28
starting 67-4, 80-3
variable binding
adding 74-24
system definition file 74-26
simulation object group 7-14, 14-32
connection 68-29
creating 68-28
terrain 5-4
Simulation Object Independently Tasked condition 36-13
Simulation Object parameter 31-20, 31-21, 33-7
simulation object type
script availability 31-25
script validity 31-26
Simulation Object Type Mappings dialog box 89-2, 89-7
filtering element definition list 89-6
Simulation Object variable 36-14
Simulation Objects Palette 14-3, 14-4
assigning or removing simulation object 68-8
Simulation Ocean Settings page 57-28
Simulation Selection Toolbar 4-11
Simulation Settings page 16-18, 16-19, 37-11
Simulation Terrain parameter 7-5
simulation time 3-11
Simulation Time condition 36-13
Simulation Time Scale Toolbar 4-11, 7-34
range, changing 7-35
Simulation Time Toolbar 4-11
simulationModelSet parameter B-6
Simulation-Model-Set-Files parameter 12-6
simulationObjectEditor.csv 67-30
simulation-run-duration parameter 7-47
sink point 18-2
creating 18-5
SISO Enumerations Document 3-8, 77-18
SISO Reference for Enumerations for Simulation Interoperability 74-4
site, cutting in 59-15
site ID 3-4, 5-8, 5-9, 5-11, 12-9
configuring B-6
specifying 5-11
at startup 5-18
siteId parameter B-6
sitting 34-31
situational awareness 50-10
size
region 6-11
sea spray 46-5
SpeedTree, randomizing 90-31
splash effect 46-5

size (continued)
water droplet 46-5
window 82-6
Size group box 69-5
Skip Task, task 25-11
skipping task 25-9
skybox cube map 45-15
omitting clouds 45-15
Skybox Cube Map Regeneration Threshold 45-15
slider
visual quality, configuring 6-20
slot
adding 72-2, 72-5, 72-11
configuring 72-5, 72-11
embarkation 72-2, 78-6
exclusion 72-9
ingress and egress technique 72-10
name 29-9, 72-2
parameters 72-7
passenger and driver 29-10
type 29-9, 72-2
smallCompass keyword 51-5, 82-17
smoke
creating 27-15
plume 43-8
trail 43-3
smooth period 6-21
smoothing 6-21
SMS
migrating 67-20
object list 67-26
vrfSim.opd 67-11
sms-filename parameter 7-47
snap to grid, formation 70-14
snapshot 7-38
automatic deletion 7-36
clearing list 7-43
creating 7-42
deleting 7-43
enabling periodic 7-43
rolling back to 7-36
saving as checkpoint 7-40
snow 11-6, 11-9
intensity 11-12
SO file 4-33
SOEUILayoutSettings 67-5
SOG. See simulation object group
soil type 64-20
configuring for GDB 55-19
soil type 64-20 (continued)
mapped to roughness type 13-12
point and feature 55-16
soil-list parameter 13-12
solid line 40-23
sonar
active, mode 34-4
active and passive 9-24
depth 34-40
dipping 29-23
information 16-4
sonobuoy 29-22
thermocline 9-24, 11-29
Sonar Depth dialog box 34-40
Sonar Depth set data request 34-40
Sonar Dip task 29-23
sonobuoy, deploying 29-22
sorting, Task and Set menus 84-3
sound effects 49-2
enabling 49-2, 49-3
mapping to entities 49-4
Space Follow mode 52-12
spawn burst 18-9
spawn event, scheduling 18-7
spawn interval 18-7, 18-9, 18-13
variance 18-7, 18-9, 18-13
spawn pattern 18-6
bursts 18-13
copying 18-14
default 18-10
deleting 18-16
description 18-11
editing 18-15
name 18-11
new 18-11
scenario-specific 18-10
timing 18-12
Spawn Pattern Manager 18-3, 18-5, 18-11, 18-14, 18-15, 18-16
spawn point 18-2
activating and deactivating 18-4
creating 18-3
enabling and disabling 41-2
spawned, entity 18-2
specifying
application number 5-9
bounding volume 69-3
damage model 69-17
entity to create 19-9
FED file 5-21

specifying (continued)
frame rate mode 12-6
HLA federation execution 5-19
language 5-3
log file 5-4
movement model 69-17
notification level 5-5
platform 68-9
point-to-point and multicast modes 5-22
port, at startup 5-21
resource 34-34
segment angle 91-4
simulation model set 5-11
site ID 5-11
terrain profile sampling frequency 56-3
unit target 35-14
specular
color 45-4
map 64-10
specular map 1-15
speed 16-9
gain 51-29
scaling 51-27
enabling and disabling 51-28
setting 34-30
specifying measurement unit 83-12
tidal current 11-22, 11-25
wind 11-9
speed scaling, mode 51-28
speed-to-rpm-factor parameter 9-25
SpeedTree 1-19, 59-8
configuring 90-30
displaying in planar reflection 90-33
exporting to shapefile 57-27
importing from shapefile 57-27
license 1-21
LOD, zoom 5-7
model definition 87-5
scale, random 90-31
shadows, enabling and disabling 45-22
SpeedTree Randomization page 90-32
SpeedTree Settings page 45-22, 90-30
speedTreeManager 59-8
SpeedTrees, planar reflection 45-10
spheroidSemiMajor parameter B-6
spheroidSemiMinor parameter B-6
splash effect 46-2
size 46-5
spot report 5-3, 9-2, 30-31, 30-32
affect on performance 6-14
spot report 5-3, 9-2, 30-31, 30-32 (continued)
custom viewpoint 9-5
decay rate 9-7
enabling and disabling 9-3
for simulation object 34-41
icon
changing 68-18
updating 68-16
label 9-8
role in target detection 9-18
tactical graphics, for 9-8
track history 9-9
pinning 9-9
using in task 9-8
viewpoint 9-4
Spot Report Settings page 9-3, 9-4, 9-5, 9-7, 9-8, 9-9, 34-41
Spot Reports dialog box 34-41
Spot Reports set data request 34-41
spotlight 45-2, 45-6
spray, enabling and disabling 46-7
spreadsheet, tag 67-31
squatting 34-31
squawk indicator 43-25
color 43-28
enabling and disabling 43-26
lifetime 43-27
standard beam 34-17
standard terrain damage 77-16
standard-power-type parameter 77-16
standard-terrain-damage-table 77-17
standard-terrain-damage-table block 77-16
standard-terrain-damage-type parameter 77-16
standing 34-31
starting
back-end 4-3
front-end 4-3
scenario, load-balanced 7-32
scenario event 8-10
simulation 7-34
Simulation Object Editor 67-4, 80-3
VR-Forces 4-3
startInRunMode parameter B-6
startMinimized parameter B-6
Start/Resume message 7-50
startup
back-end minimize B-6
option 5-1

startup (continued)
tutorial 4-9
state
sea 11-25
simulation object, information 16-4
update, sending 50-11
state property 74-26
accessing from lua 74-29
adding 74-27
data types supported 74-28
displaying in Information dialog box 74-29
user-defined 74-26
state repository 76-10, C-4, D-5
hierarchy 74-10, 74-12
statement
adding to plan 37-4
deleting 37-12
editing in plan 37-11
plan, drag and drop 37-42
state-repository parameter C-4, C-6
state-repository-min-tick-period parameter 6-16
state-repository-min-tick-period-variance parameter 6-16
statistics
displaying 6-7
VR-Vantage performance 6-9
status
combat engineering object 22-14
entity 16-21
navigation area 61-15
personnel, ammunition, weapons, and POL 16-6
subtask 32-15
task 25-8, 32-15
statusUpdateHeartbeat parameter B-6
Stealth
control 37-37
observer mode 3-16
window 4-13
stencil bits 5-7
stencil buffer 6-16
step-function 77-8, D-19
stereo display, types of 82-21
Stop/Freeze message 7-50
stopping
movement 30-22
scenario event 8-13
subtask 32-15
task 25-9, 32-15
stopping-distance-factor parameter 78-4
stores, restoring simulation object 34-34
Stow Hand Item dialog box 34-42
Stow Hand Item set data request 34-42
Stow Refueling Boom task 29-18
Strafe Ground Target dialog box 27-23
Strafe Ground Target task 27-22
strafing, target 27-7
strategic missile 10-8
strategic missile weapon system 27-14
streamed data
performance 6-14
processing 59-8
streaming
beacon 59-27
buoys and beacons 59-22
navigation lights 59-28
terrain 54-9
configuring cache 64-7
street light 45-2, 45-6
strength 22-4
strong point
creating 30-19
improving 30-24
style
fill 40-23
line 40-23
pen 40-23
subcomponent C-4
classes 74-9
sub-dataset
elevation, CDB 60-4
extended imagery, CDB 60-16
geospecific, CDB 60-8
geospecific airport lightpoint, CDB 60-9
geospecific environmental lightpoint, CDB 60-10
geotypical, CDB 60-15
imagery, CDB 60-5
imagery lightmap, CDB 60-6
subgoal 61-5
submarine
lowering periscope 29-17
moving to depth 26-14
raising periscope 29-18
subnet mask 5-9, B-8
subnetMask parameter B-8
subordinate
adding to unit 70-4
disaggregated unit 23-9

subordinate (continued)
editing 70-7
in condition 37-23
information 16-7
order, changing 70-6
removing from unit 70-5
replacing 70-6
task status 36-13
unit, aggregated unit 23-9
Subordinates tab, unit 23-9
subsurface entity
moving to depth 26-14
RPM 9-25
subsurface-entity-param parameter 74-12
subsurface-entity-param parameter C-8
subsurface-entity-state-repository 74-12
subtask
Lua 32-14
state
canceled 32-15
complete 32-15
running 32-15
status 32-15
stopping 32-15
sun, ray 45-13
sunbeam 45-13
sunrise-time parameter 75-15
sunset-time parameter 75-15
superior
setting 34-42
tasking simulation object 34-44
Superior dialog box 34-42
Superior set data request 34-42
super-type 74-4
supplies 22-15
information 16-7
providing 22-17
receiving 22-17
starting or stopping delivery 35-16
Supplies page 22-14
supplies parameter, editing 71-25
supply line 43-23
Supplying dialog box 35-16
Supplying set data request 35-16
support, technical lxi
support point, setting 69-5
suppressDataAvailableException parameter B-6
suppression-degrades-sensor-performance-by parameter 25-33
suppression-detonation-insult parameter 25-32
suppression-detonation-range parameter 25-32
suppression-insult-level parameter 25-32
suppression-recovery-time parameter 25-32
suppressive fire
at location 25-30
at point, line, or area 25-30, 27-19
suppressSelfReflect parameter B-8
surface
characteristics 64-20
clamping to 54-9
damage specification 77-8, D-19, D-23
transparency 11-22, 11-25, 82-16
type, mapped to roughness type 13-12
surface entity
buoyancy 46-8
RPM 9-25
wake 43-3
configuring 90-18
zig zag to heading task 26-29
surface-entity-param parameter 74-12, C-8
surface-entity-state-repository 74-12
surfChar.map 64-20
surge depth 11-6, 11-22, 11-25
surrender
lifeform, testing 36-11
surrendered 36-11
Surrendered dialog box 34-42
Surrendered set data request 34-42
swell 1-14, 11-6, 11-22
sea 11-25
switch bead 92-9
switch node 90-21, 90-23
displaying 68-21
visual model 68-20
Switch Node States 68-20
switch type, dynamic terrain 62-4
SwitchStateMapper, feature 90-23
symbol, location of 2-4
symbol decoration, showing and hiding 16-15
Symbol Decoration Settings page 16-14, 16-15, 16-16
synchronization, vertical 6-24
Synchronize Laser Code dialog box 34-43
Synchronize Laser Code set data request 9-19, 34-43
synchronized, ocean 5-6
synchronizing
back-ends 4-19
data and settings 3-16

synchronizing (continued)
laser codes 34-43
sysdef file. See system definition file
system 74-17
active-radar-sensor 22-19
adding to simulation object 69-16
aggregate-comms-jammer 22-18
aggregate-radar-jammer 22-19
configuring in Simulation Object Editor 67-3
connection 74-18
damage 77-2, D-13, D-28, D-31
editing, properties 69-18
embedded entity 17-9
emissions-sensor 22-20
entity 69-15
explosive device 76-19, D-12
marker in 3D model view 69-18
missile launcher 76-14
movement, properties 71-20
Naval Depth Charge Deployment 27-4
Naval Mine Deployment 27-5, 27-6
naval mine explosive device 76-19, D-12
Remote Sensor Receiver 34-33
removing 69-18
script 31-3
scripted task, creating 31-42
sensor 75-2
detection probability table 75-9
simulation object, editing 69-15
tactical radar jammer 9-20
weapon 76-2
system definition, specifying scripted task in 31-27
system definition file 74-17, 76-2
applying new variable bindings 67-25
configuring connections 74-18
creating 74-17
sensor 75-4
variable binding 74-22
adding 74-23
addingto Simulation Object Editor 74-26
system scripted task, making available on Task menu 31-42

T

-T command-line option 5-12
-t command-line option 63-5
tab
Create Object 14-13
overlay 39-4
Scene Graph 64-22
Viewpoint 9-3
table
damage D-22
detection 9-16
Lua 32-14
taskParameters 32-16
tactical data link, message 9-10
Tactical Data Link Settings page 9-10, 9-11, 9-12
tactical graphic 38-2, 89-2
active and inactive 40-24, 41-2
attaching to simulation object 40-16
automatically attaching to simulation object 40-16
box, drawing 40-4
changing name 14-25
changing state with set data requests 41-2
color, changing 40-21
compared to combat engineering object 22-13
console, notification level 16-24
creating 14-3, 14-4, 14-7, 14-11, 14-13, 40-2
editing 40-17
element definition 88-2
export to shapefile 39-7
extruded 40-5
force 41-4
HAT lines 38-8
hiding from Overlays tab 38-7
importing shapefile to 39-7
information 16-3
label 38-8
labels 38-3
loading 39-6
menu icon 68-30
model 68-33
moving, to different overlay 40-6
notify level 41-5
placing 14-13
property
default value 68-31
editing 68-32
keyword 68-32
setting at creation 14-15

tactical graphic 38-2, 89-2 (continued)
publishing 40-8
saving 39-6
selecting 15-5
showing and hiding 38-6, 38-7, 40-24, 41-2
texture 68-33, 68-35
vertex
adding 40-18
deleting 40-20
vertex color 40-21
volumetric 40-5
Tactical Graphic Active condition 36-13, 40-24, 41-2
Tactical Graphics Display Settings page 38-7, 38-8, 40-21, 40-25, 41-2, 42-2
Tactical Graphics Palette 14-4
Tactical Graphics Settings page 40-24
tactical radar jammer system 9-20
tactical smoke, enabling or disabling 43-23
tactical smoke element definition, exporting 88-18
tactical-smoke-param parameter 74-12
tag
adding 67-27
assigning based on 3D model characteristics 67-30
changing 68-4
editing in spreadsheet 67-31
exporting 67-30
importing 67-31
object, changing 68-4
Object Group 14-32
removing 67-27, 67-29
renaming 67-27, 67-29
taillight 45-2, 45-6
takeoff, fixed-wing entity 25-25, 25-26
taking off 26-7
target
bomb 27-7
detecting 76-9, D-4
probability of 75-9
detection 9-16
spot reports 9-18
firing at 9-16, 27-9
identifying with laser beam 9-18
jamming 34-4
setting 34-43
specifying using laser 27-13
strafe 27-7
target (continued)
strafing 27-22
unit, preferred 35-14
Target dialog box 34-43
target selection controller 76-9, 76-15, 76-16, D-4
Target set data request 34-43
targetFrameRate parameter B-6
targeting-capable parameter 34-17
targeting-control parameter 76-9, D-4
target-type parameter D-6
task 25-3
Activate Jammer 30-4
adding to plan 37-4
Air Drop 26-3
Air-to-Air Attack 30-4
Air-to-Ground Air Base Attack 30-4
Air-to-Ground Attack 30-5
Animated Movement 26-4
Anti-ship (Fixed Depth) 27-16
Arm Mine at Depth 27-3
articulated parts, moving 29-16
assigning to unit 25-10
Attack Aircraft with Guns 27-3
Attack By Fire 30-5
Attack with Anti-Air Missile 30-6
Attack with Anti-Ship Missile 30-6
Attack with Land-Attack Missile 30-6
Attack with Torpedo 30-7
Automatic Air Defense 30-7
background 30-34
background process 32-32
Biological Attack 30-7
Bomb Location 30-8
Breach Obstacles 30-9
by superior 34-44
C++ 31-3, 32-14
Car Door 29-17
Change MOPP Level 30-10, 35-12
Change Posture 30-10, 35-13
Chemical Attack 30-11
Circle Around Location 28-3
Civilian Idle 28-3
Close Car Door 29-16
Close Cargo Door 29-16
Close Door 28-3
Close Window 28-4
Come to Stop 26-4
concepts 24-1
concurrent 24-1, 25-4, 31-8, 31-24, 31-31

task 25-3 (continued)
configuring mutually exclusive 25-33
confirmation prompt, enabling or disabling 25-5
Construct Abatis 30-12
Construct Anti-Tank Ditch 30-12
Construct Barbed Wire 30-13
Construct Barricade 30-13
Construct Berm 30-14
Construct Booby Trap 30-14
Construct Bridge 30-15
Construct Dry Gap 30-16
Construct Fortified Area 30-17
Construct Fortified Line 30-18
Construct Minefield 30-18
Construct Strong Point 30-19
Convoy Along 25-10, 26-5
Convoy To 25-10, 26-5
Create Pedestrians 28-4
Crowd Along Line 28-4
Crowd Around Location 28-5
Crowd Around Object 28-5
Crowd in Front of Entity 28-6
Deactivate Jammer 30-19
Deploy Refueling Boom 29-16
Deploy Sonobuoy 29-22
Deploy Sonobuoys Along Route 29-22
Destroy Bridge 30-20
Destroy Ditch 30-16, 30-20
Destroy Explosive 30-20
Destroy Fortification 30-21
Destroy Obstacle 30-21
DI-Guy Animation 28-7
DI-Guy Gesture 28-7
Disembark 29-3
Disembark All 29-3
Disperse Crowd 28-8
Drop Fuel Tanks 30-21
Drop Naval Depth Charge 27-4
Drop Naval Depth Charge at Location 27-4
Drop Naval Mine 27-5
Drop Naval Mines Along Route 27-6
Embark 17-3, 29-4
escaping retask procedure 25-6
Escort Air Unit 30-22
Execute Close Air Support 27-7
Explode Charge at Depth 27-9
FASCAM 30-22
Fast Rope Insertion 26-6

task 25-3 (continued)
filtering simulation object list 25-9
Find Cover 28-8
Find Firing Position 28-9
Fire at Target 27-9
Fire Cruise Missile 27-10
Fire-for-Effect 27-11
Fixed Wing Land 25-27, 26-7
Fixed Wing Takeoff 25-26, 26-7
Flee From Entities 28-11
Fly Altitude 26-8
Fly Heading 26-8
Fly Heading and Altitude 26-9
Follow Entity 26-10
Generate Air Traffic From Flight Plans 26-11
gesture 28-7
Halt Movement 30-22
ID 32-14, 32-15
Improve Booby Trap 30-23
Improve Breach 30-23
Improve Bridge 30-23
Improve Ditch 30-16, 30-24
Improve Fortification 30-24
Improve Obstacle 30-24
independent 24-1
Indirect Fire 30-25
Land at Air Base 30-25
Land at Airport Near Location 26-11
Lase Target 9-19, 27-13
Launch Anti-Submarine Missile (Vertical) 27-13
Launch Ballistic Missile at Location 27-14
Launch Ballistic Missile at Target Object 27-14
Launch Counter Measures 9-26
Launch Flight 30-26
Launch QRA Aircraft 30-27
Launch Smoke 27-15
Lower Periscope 29-17
Lua 32-14
parameter 32-16
Move Along Route 26-12
Move Along Route Retrograde 30-28
Move Between Cover to Location 28-11
Move Into Formation 26-13
Move to Altitude 26-14
Move to Depth 26-14

task 25-3 (continued)
Move to Location 26-15
for plan 26-16
Move to Location (Plan Along Roads) 25-21, 26-16
Move to Location Retrograde 30-28
Move to Waypoint 26-18
Move to Waypoint (Plan Along Roads) 25-21, 26-19
Move to Waypoint Retrograde 30-28
movement, affect of aggregation state change 23-10
mutually exclusive 24-1, 25-4, 36-6
Nuclear Attack 30-29
Open Cargo Door 29-18
Open Door 28-12
Open Window 28-12
Orbit 26-21
Orbit Object 26-21
Parachute to Target 28-13
Parallel Pattern 26-22
parameters 25-6
Patrol Between 26-23
Patrol Route 26-23
Pattern Hold (Location) 26-24
Pattern Hold (Waypoint) 26-24
performing 13-7
Place IED 27-18
Prepare QRA 30-30
procedures, introduction 25-3
Protest Along Line 28-14
Protest Around Location 28-14
Provide Suppressive Fire 25-30, 27-19
Provide Suppressive Fire on Location 25-30, 27-20
Raise Periscope 29-18
reactive 19-10, 25-11, 29-25, 31-5, 32-7, 32-31
Refuel from Tanker 30-30
Refueling Mission 30-31
Release Bomb on Laser Spot 27-20
Release Bomb on Location 27-21
Release Bomb on Target 27-21
Release Next Entity From Queue 29-23
retask 25-3
Rotary Wing Airlift Cargo to Location 26-26
Rotary Wing Land 26-26
Rotary Wing Retrieve Cargo 26-27
Rotary Wing Unload Cargo 26-27
task 25-3 (continued)
Sail Heading 26-28
scripted 31-3, 31-4
action category 25-34
adding to task toolbar 31-29
adding to toolbar 84-9
available by behavior set 31-27
concurrent 31-31
execution flow 32-8
filtering list 31-37
frame rate 6-6
Sector Search Operation 29-20
Send Radio Set 29-11
Send Radio Task 29-13
Send Text Message 29-14
simulation object 13-8
Skip Task 25-11
skipping 25-9
Sonar Dip 29-23
spot report, using in 9-8
state
canceled 32-15
complete 32-15
running 32-15
status 16-4, 25-8, 32-15
stopping 25-9, 32-15
Stow Refueling Boom 29-18
Strafe Ground Target 27-22
Throw Grenade at Location 27-24
toolbar, adding to 84-9
Turn to Heading 26-29
unit concepts 21-5
user 29-24
viewing current 36-15
visualizing 43-12, 43-13
Wait 29-15
Wait Duration 29-15
Wait Elapsed 29-15
Wait for Escort 30-33
Wait in Queue 28-15
Wander 28-15
Wander In Area 28-16
who can execute 25-6
Zig Zag to Heading 26-29
Task menu 31-4
editing 84-8
extended name 31-12
icon 31-12
location for script 31-23
sorting 84-3

Task menu 31-4 (continued)
system scripted task, hiding 31-42
task visualization B-6
pinning 43-13
Tasked by Superior set data request 25-11, 34-44
using in plans 36-17
tasking, by superior 34-44
task-manager parameter C-5, C-7
taskParameters, table 32-16
taskVisualizationPublicationMode parameter B-6
TDL, message 9-10
TDL message
configuring 9-11
track history, pinning 9-12
TDL Messaging dialog box 34-44
TDL Messaging set data request 34-44
tearing off menu 83-8
technical support lxi
technique
egress 72-10
ingress 72-10
teleporting observer 51-17
temperature, air 11-6, 11-10
template
earth file 59-6, 64-5
unit 71-2
temporary directory, scenario 12-11
terminating, application 4-23
terrain
See also terrain database and terrain server
agility 1-12, 54-2
altitude point
changing color 55-5
deleting 55-4
inserting 55-4
analyzing 56-5
avoidance 25-29
bounding volume, drawing 64-22
building 57-4
caching 64-2, 64-3
caching earth files 1-18
coloring by elevation 55-3
composability 54-2
coordinates 55-16
creating 57-2
damage-by-power 77-16
databases provided 55-21
debugging 64-16
terrain (continued)
deleting cached 64-6
displaying raster image on 57-6
dragging with mouse 51-14
DTED support 57-2
dynamic 5-2, 62-3
damage 77-16
global dynamic terrain processor 7-7
navigation data regeneration 61-16
earth file, debugging 64-7
embedding buildings in 14-34
embedding cultural feature 69-9
embedding cultural features in 14-34
extracting props from 54-4
feature data, configuring 65-2
formats, supported 54-3
GDB, soil type 55-19
geocentric 54-6
indirect rendering 64-14
intersector line 64-16
large 54-7
Little Pond 55-21
loading, from command line 5-7
Makland 55-22
MetaFlight
designing 63-2
loading 58-5
processing 63-2
opening 54-7
orbiting, with mouse 51-16, 51-19
paged
configuring LODs 64-7
flipping DDS textures 64-17
loading 5-10
preprocessing 58-6
page-in area 38-2, 58-6
paging 54-8
manually 58-6
MetaFlight 58-5
patch 54-3
adding to terrain 57-4
extracting props from 57-20
placement of new entities 14-17
profile. See terrain profile
prop
adding 57-20
list 57-24
querying 65-3
San Luis Obispo 55-22
saving 57-3

terrain (continued)
scale 55-17
server. See terrain server
shape 54-3
simulation object group 5-4
soil type 55-16
streaming 54-9
cache 64-7
underwater 57-28
wireframe mode 55-8
Terrain Contents page 57-4, 57-13, 57-15, 57-16, 57-17, 58-3, 64-3, 64-8, 66-8
terrain damage, standard 77-16
terrain database 1-12, 54-6
changing for a scenario 12-8
configuring B-6
path to B-3
loading 4-21
at startup 5-12
provided 55-21
terrain draw area 55-18
Terrain Information Panel 55-13, 64-22
terrain profile
concepts 56-2
configuring 56-3
displaying, line creation 40-7
line 38-2, 56-5
opening automatically 56-3
sampling frequency 56-3
terrain profile line 56-5
track history 43-7
vertex, showing 56-3
Terrain Profile Settings page 48-15, 56-3
Terrain Profile window 56-2, 56-3
intervisibility line 48-15
terrain server 54-7, 54-9, 58-2
adding 58-4
caching data 64-3
connecting through proxy 58-5
cutting in site 59-15
mask, adding 59-15
Terrain Settings dialog box
Add Props page 57-22
Earth Layers page 55-10
Edit Existing Props page 57-24, 57-25, 57-26, 66-11
Extract Ocean page 57-15
Extract Props page 57-21, 66-9
Navigation Areas page 61-8
Raster Maps page 57-6, 57-10
Terrain Settings dialog box (continued)
Simulation Ocean Settings page 57-28
SpeedTree Randomization page 90-32
Terrain Contents page 57-4, 57-13, 57-16, 57-17, 58-3, 64-8, 66-8
Visible Surfaces page 55-19
terrain-check parameter 25-29
Terrain-Database parameter 12-5
terrainDatabase parameter B-6
TerraPage file 54-3, 58-6
testing, amount of resources 36-12
Tether Location Track mode 52-14
Tether mode 51-8, 51-24, 52-8, 52-13
Tether Track mode 52-14
text
creating 40-5
in title bar 83-11
object, resizing 40-20
sending as command 37-34
sending as task 29-14
shadow 16-16
text editor 31-4
scripts, specifying 31-44
texture 90-34
atlas, buoy and beacon 59-26
compression, enabling and disabling 64-6
coordinate system for image 57-8
data, converting 64-20
DDS 64-17
flipping, paged terrain 64-17
map 1-15, 64-10
mapping to model 90-22
model 90-22
ocean planar reflection
height 45-10
width 45-10
sharing 6-13
size, ocean height map 57-17, 57-19
skinning
metafile 90-23
model 90-22
tactical graphic 68-33, 68-35
tiled and virtual 63-4
water 57-14
texture atlas 90-22
thermocline, sonar 9-24, 11-29
Thermoclines page 11-29
this, Lua object 32-3
threat ring 43-16

threshold
  data updates B-3
  DIS heartbeat B-7
  object timeout B-8
Throw Grenade at Location dialog box 27-24
Throw Grenade at Location task 27-24
Throw Grenade at Target dialog box 27-24
tick
  rate, tuning 6-15
tick rate, scripts 6-6
tidal current
  direction 11-25
  speed 11-25
  speed and direction 11-22
tidal stream, wake 11-26, 90-20
TIFF 54-4
tiled texture 63-4
time
  advancement 3-11
  evaluating in plan 36-13
  scenario
    end 7-44
    ending 7-7
  setting 11-3, 11-5
  simulation 3-10, 7-12
  zone 7-12, 11-5
    setting 11-5
time constrained 4-19
time management
  concepts 4-19
  configuring 4-19
  enabling 4-19
    for back-end 4-20, 5-21
  RTI requirements 4-20
  setting simulation speed 7-34
Time Multiplier parameter 7-6
Time Multiplier toolbar 12-5
time of day 3-10, 11-3
  advancement 3-11
  illumination 75-14
time regulating 4-19
time scale
  parameters, changing automatically 7-35
time stamp 5-7, 5-12
time stamp order, configuring B-7
time stamp ordered message 4-19
time zone 11-3
timed fuse 34-14
time-multiplier parameter 12-5
timeout
  DIS 5-10
  interval 5-8
  object B-8
timestamp 5-3, 5-7
timing, spawn pattern 18-12
tire track 43-3
title bar
  text, adding 83-11
To Echelon Level 24-6
tool, osgearth_cache 64-3
toolbar
  adding task or set 84-9
  Attachments 52-3
  button 4-11
  configuring 84-2, 84-9
  Display Settings 16-15
  displaying and hiding 83-3
  element 84-9
  Environment Settings 7-12, 7-13, 11-4, 11-5
  icon, setting 84-10
  locking 84-10
  Map Scale 55-17
  Objects 15-6, 17-2, 17-6, 23-6
  Quick Launch 8-10
  Scenario 7-42
  Scenario Event 8-2
  Simulation Control 7-34, 7-36, 7-37
  Simulation Objects Palette 14-4
  Simulation Time Scale 7-34
  task, adding script 31-29
toolbarrow, element 84-9
topmark, buoy and beacon 59-30
topmark_attr_map.csv 59-30
topological model 3-13
TOPSHP attribute 59-30
torpedo 30-7
  anti-submarine 27-13
  launching 27-16
tracer 34-44, 43-8
  configuring 76-11, D-6
Tracer Use dialog box 34-44
Tracer Use set data request 34-44
tracer-type parameter 76-11, D-6
track, radar mode 34-32
track history
  configuring 43-6
  displaying 43-5

track history (continued)
enabling and disabling for all simulation objects 43-5
length 43-6
spot report 9-9
pinning 9-9
TDL message
pinning, 9-12
terrain profile 43-7
Track History Settings page 43-6
Track mode 51-8, 52-8, 52-15
tracking
ammunition 35-4
entity, resources 16-21
equipment 35-8
resource 35-14
weapon 35-17
tracking beam 34-17
traffic
generating 18-2
spawn pattern 18-6
trail, model 43-4
trailing effects 43-3, 89-2
enabling and disabling 43-3
model 43-4
simulation object 43-3
trajectory
indirect fire 34-32
smoothing 6-21
trajectory history. *See* track history
transform, multiple 65-5
transforming, attribute 65-4
transient intervisibility
displaying 48-6
object 48-5
transition blending, LODs 64-15
transition time
posture, editing 71-21
translating, GUI text 2-11
translation 71-27
transmitter, publishing radio 79-4
transparency 90-34
alpha to coverage 6-17
prop, setting 57-25
surface 11-6, 11-22, 11-25, 82-16
transponder, IFF 34-21
trap, creating 30-14
Travel, posture 22-6
tread track 43-3
tree
prop type 57-26
setting avoidance 34-11
SpeedTree 90-30
triangle, vertex label 38-8
triangle icon 16-27
trigger 36-6
automatically checking for unused 37-11
cleaning up unused 37-10
deciding order in plan 36-16
name 36-7
reactive task 25-11
registering 37-9
registration 36-7
taskless 36-16
unregister 37-10
Triton Ocean SDK 1-20
Triton.config, ocean configuration file 51-23
TrueType font, 2D icon 90-4
tuning
components 6-15
network performance 6-19
performance
component 6-15
network interface 6-16
state repository 6-16
visual performance 6-19
turn rate 26-8, 26-9
Turn to Heading dialog box 26-29
Turn to Heading task 26-29
turret, attaching to 52-5
tutorial, starting VR-Forces 4-9
TXP file 54-3
type
adding 71-27
prop 57-26
slot 29-9, 72-2
type attribute 90-6

U

-u command-line option 63-5
UI file 74-24
variable binding, adding 74-25
unconstrained 4-19
underFireDistance parameter 34-35
under-fire-parameter parameter 36-11
underFireTime parameter 34-35

underwater, visibility 11-22, 11-25
Underwater Acoustics PDU 9-24
undo, ellipse rotation 40-20
undocking, panel 83-2
undoing
  category changes 67-29
  simulation object edits in Simulation Object Editor 68-12
unexploded ordnance, destroying 30-20
unique
  buoy 59-25
  ID 13-3
unique ID, UUID 7-38
unit C-8
  activity 35-3
  adding simulation object to 23-7, 34-42
  aggregated state 23-9
  aggregate-level 23-5
    aggregated 30-5
  ammunition, weapons, equipment 71-4, 71-7
  artillery 30-25
  assigning tasks 25-10
  attrition 22-12
    from combat engineering object 22-12
  bounding box, Simulation Object Editor 70-11
  bounding volume, color 24-10
  changing state manually 23-12
  Circle Around Location task 28-3
  closing formation 21-5
  collapsing 24-4
  collapsing to root 24-4
  convoy 26-5
    behavior 25-10
  count of 16-28
  creating 23-2, 23-3, 70-4
    preconfigured 23-5
  creating in Order of Battle Panel 20-5
  crowd, creating 19-3
  deleting 23-13
  disaggregated and aggregated 21-2
  disaggregated state 23-9
  disaggregation area 23-12
  display 83-12
  displaying ghosted icons 24-4
  echelon ID 21-3
  editing 70-4
  embark task 29-4
  entity, individual tasks and plans 23-11
unit C-8 (continued)
  entity-level 21-2
  expanded and collapsed 21-2
  expanding 24-4
  expanding all 24-4
  expanding and collapsing 24-2
  footprint 22-5
  formation 21-4, 34-19, 78-2
    adding 70-9
    automatic layout 70-13
    copying 70-12
    default 70-10
    editing 70-8
    manual layout 70-13
    removing 70-10
    renaming 70-10
    snap to grid 70-14
  information 16-3, 16-7
  introduction 23-8
  measurement 83-12
  movement 21-4, 22-8
    affected by combat engineering objects 22-11
  moving to formation 26-13
  new 71-2
  organization concepts 21-3
  parameter type 74-39
  posture 22-6
  preconfigured 23-5
  removing simulation object from 23-8, 34-42
  reorganizing 21-4, 21-5, 34-33
  restoring 23-10, 34-34
  restricting movement 65-9
  sector 22-5, 43-16
  selecting 23-6, 24-2
  sensor 22-10
  setting formation in plan 36-17
  specifying as Aggregate As option 70-3
  state
    affect on combat 23-10
    affect on movement task 23-10
    affect on resource tracking 23-10
    at creation 23-6
    changing at runtime 23-10
    initiating change 23-11
    setting 34-45
    showing in GUI 23-9

unit C-8 (continued)
  subordinate
    adding 70-4
    changing order 70-6
    editing 70-7
    in condition 37-23
    removing 70-5
    replacing 70-6
  target, preferred 35-14
  task behavior 21-5
  task status of subordinates 36-13
  template 71-2
  writing plan for 37-41
Unit Display Settings page 16-19, 24-5, 24-9, 24-10, 43-10, 43-23
Unit State dialog box 34-45
Unit State set data request 34-45
Universal Transverse Mercator 54-6
unloading
  cargo 26-27
  entities from entities 17-2
unlocking, overlay 39-4
unpublishing, tactical graphics 40-8
unregistering, trigger 37-10
unselecting
  object 15-7
  simulation object 15-7
unused range 77-5, D-16
unused trigger
  automatically checking for 37-11
  cleaning up 37-10
update
  frequency of B-3
  messages 50-11
  rate, intervisibility 48-10
Update_EW_Degradation, script 22-19
updater 90-10
  AOA 90-12
  attribute 90-10
  FeetAGL 90-12
  FeetMSL 90-13
  FeetPerMinute 90-13
  Headlight 90-13
  KilometersPerHour 90-13
  KnotsPerHour 90-13
  MilesPerHour 90-13
  Pitch 90-13
  Roll 90-13
  Value 90-13
  Yaw 90-13
updating
  platforms 67-25
  visual models 68-25
  window layout 83-6
upgrades lxi
upgrading, weapon and damage systems 67-21
Use Extracted Geometry 57-20
useAbsoluteTimeStamps parameter B-7, B-8
useAdvisories parameter B-7
useAsyncIO parameter B-8
useCustomMapDatum parameter B-6
USE-DEFAULT keyword 12-10
useDISGuy parameter B-7
useDrawControlForTaskVisualizations parameter B-6
useIpv6 parameter B-8
use-logger-control parameter 7-47
user
  settings 3-15
  task 29-24
user data, specifying directory 5-7
User Task dialog box 29-24
user-defined, formation 78-4
user-defined state property 74-26
using
  Boundary Generation Tool 59-16
  navigation data 61-16
UTM 54-6
  coordinate system 51-7
  database 7-3
UUID 13-2
  scheme 5-6
  simulation object 13-3
UV, coordinates 57-8
UV offset 90-24
UvAtlasMapper, feature 90-23

V

-v command-line option 5-7, 5-12, 5-16, 63-5
Value updater 90-13
vapor trail 43-3
variable
  Created Object 36-14
  global, saving 32-13
  plan 36-14
  Simulation Object 36-14
  tick rate 6-15

variable binding 74-22
  adding to platform file or system definition file 74-23
  adding to Simulation Object Editor 74-22
  applying changes 67-25
  default value 74-23
  platform file 74-22
  Simulation Object Editor, adding new 74-24
variable frame mode 6-22
variable-frame keyword 12-6
variable-frame run-to-complete 3-12, 12-6
variance, spawn interval 18-7, 18-9
varying, heartbeat B-8
VBO 5-3
vector
  geocentric 32-22
  Lua 32-19, 32-22
  network 25-21
vector feature
  configuring avoidance of 78-5
  drawing 55-15
  list 55-13
vector feature draw area 55-15
Vector3D class 32-19
VectorGeoc3D class 32-22
VectorOffset3D class 32-22
vehicle
  displaying driver in 44-8
  model 1-6
velocity
  displaying 16-10
  specifying measurement unit 83-12
version
  displaying 5-7
  plug-in, checking 5-3
  RPR FOM 5-8, 5-13, B-7
version parameter, scenario file 12-7
vertex
  adding 40-18
  altitude, displaying 38-8
  deleting 40-20
  editing 40-19
  line, terrain profile 56-3
  showing and hiding 38-8
  tactical graphic, color 40-21
vertex buffer object 5-3
video, adding to scenario event 8-5
video card, processor speed 6-4
view
  clearing list 51-32
  constraint 51-27
  default 51-33
  deleting 51-32
  dragging with mouse 51-16
  floor 51-27
  inset 53-2, 53-3
  loading saved 51-34
  model 68-21
  navigating 51-3
  observer, transitioning between 51-31
  recalling 51-30
  restoring 51-31
  saved, scenario 12-2
  saved with scenario 6-1
  saving 51-30, 51-31, 51-34
    individual 51-34
  sensor 53-8, 53-10
  zooming 51-19
    2D 51-21
    3D 51-20
view constraint, disabling 14-17
view control, sending in plan 37-37
view control message 51-35
  enabling processing of message 51-35
  script 51-35
view mode
  Absolute 52-8
  Follow 52-9
  Mimic 52-10
  Mimic Location Track 52-11
  Mimic Track 52-11
  Space Follow 52-12
  Tether 52-13
  Tether Location Track 52-14
  Tether Track 52-14
  Track 52-15
viewing
  current task 36-15
  list of props 57-24
  Logger files 7-50
  obstructed terrain in terrain profile 48-15
  other MAK 3D GUI viewers 50-12
  plan 36-15
  plug-in, data 4-39
  terrain polygons 55-18

viewpoint
spot report 9-4
custom 9-5
Viewpoint tab 9-3
viewport 82-7
changing 82-15
virtual texture 63-4
Virtual Texture Datasets 63-3
visibility 11-9, 11-11
modifier 22-12
observer 11-6
underwater 11-6, 11-22, 11-25
water 82-16
visible surface, configuring 55-19
Visible Surfaces page 55-19
Visit Interest Points reactive task 19-10
visual, markings 68-23
visual definition 86-2, 88-2, 88-8
adding 88-8
attribute 88-9
adding 88-12
editing 88-9, 88-11
Entity Image Symbol 90-3, 90-4, 90-9
Military Symbol Icon 90-3
schema 86-2
visual effect, displaying 68-22
visual marking, randomized 68-23, 68-24
Visual Markings dialog box 68-23
visual model
adding 68-18
details 68-20
editing 68-13
exporting 68-25
importing 68-24
pruning 68-25
refreshing 68-25
tactical graphic 68-33
Visual Model Editors dialog box 87-2, 89-7
Visual Models Panel 68-13
visual quality 57-17
visual quality slider, configuring 6-20
visual sensor 75-14
visual type
changing
model, changing type 68-22
visualization, task B-6
visualizer 88-2
visualizing, tasks 43-12, 43-13
volumetric, tactical graphic 40-5
vrf class 32-3
vrf-aggregate-state-repository 74-12
vrf-base-object-param parameter 74-39, C-3
vrf-base-object-param parameter type C-7
vrfGui 3-2
vrfGui.log file 4-40
vrfLauncher
combined mode 5-17
special command-line arguments 5-17
vrf-moving-object-state-repository 74-12
vrfMsgDump 1-18
vrf-object-param parameter C-7
vrf-object-param parameter 74-12
vrf-object-state-repository 74-12
VR-Forces
entity or object 3-9
Launcher 4-4
starting 4-3
window 4-11, 4-21
VR-Forces Objects Panel 52-2
VR-Forces window 7-7
vrf-overlay-object-state-repository 74-12
vrfSim
log file 5-11
running as daemon B-2
vrfSim 3-2
daemon, starting as 5-10
vrfsim
enabled 59-19
layer 65-2
VRFSIM_OSGEARTH_CACHE_PATH environment variable 64-3
vrfsim: 59-18
vrfSim.log file 4-40
vrfSim.mtl 61-12
vrfSim.mtl file B-2
vrfSim.opd
remote-configurations block 81-3
required in every SMS 67-11
vrfSimSettings.xml file 6-19
vrf-tactical-smoke-state-repository 74-12
vrfutil, functions 32-5
VRV_ALPHA_TO_COVERAGE_RENDERBIN environment variable 6-17
VR-Vantage, view control 37-37
VR-Vantage Control Toolkit 51-35
VSync 5-3, 6-24
VTDS 63-3
vulnerability 22-4
modifier 22-11

W

-w command-line option 63-6
wading 34-31
Wait Duration dialog box 29-15
Wait Duration task 29-15
Wait Elapsed dialog box 29-15
Wait Elapsed task 29-15
Wait for Escort, task 30-33
Wait for Escort dialog box 30-33
Wait in Queue dialog box 28-15
Wait in Queue task 28-15
Wait task 29-15
Wait Until 36-8
Wait Until Expression dialog box 37-6
wake 1-14, 1-20, 43-3
  enabling and disabling 46-7
  model definition 11-26
  ship, configuring 90-18
  tidal stream 11-26, 90-20
wakeLength parameter 11-26
walking 34-31
wall, not walking through 51-27
wall clock time 3-11
wall-clock time 4-20
wander, crowd 28-16
Wander In Area task 28-16
Wander task 28-15
warfare, electronic 9-20, 22-18, 22-19, 22-20
warfare model, aggregate 22-3
warhead actuator 76-20, D-12
warhead parameter 76-11, D-6
warhead parameter 77-3, D-14
warhead-detonation-actuator 76-19, D-11
warning, icon 16-27
warning message B-4
water
  pacing and tracking 35-10
  splash 46-2
  texture 57-14
    extracting 57-15
  visibility 82-16
water droplet, size 46-5
wave
  breaking 11-27
  choppiness 11-22
  effect 46-2
wavelength, laser 34-25
waypoint 38-2, 38-4
  altitude, displaying 38-8
  civilian visit point 19-10
  ordering entity to 26-18
  patrolling between 26-23
weapon
  anti-air 30-7
  laser guided 27-13
  pacing and tracking 35-17
  status 16-6
  system 69-15, 69-16
  system definition file 74-17
  unit 71-4, 71-7
weapon interface 76-9, D-4
weapon system 76-2, 76-8
  bomb-bay 76-20, D-12
  indirect fire 76-17, D-9
  migrating to new release 67-21
  strategic missile 27-14
weapong system D-2
weapon-interface 76-15, 76-16
Weapons Pacing/Tracking dialog box 35-17
Weapons Pacing/Tracking set data request 35-17
weather 11-6
  global 7-7
  setting 11-9
Weather dialog box 11-15
Weather page 11-6, 11-7, 11-8, 11-9, 11-10, 11-11, 11-12, 11-24, 82-16
weather zone 11-13
  priority 11-13
When statement 36-3, 36-6
  building 37-13
While Expression dialog box 37-6
While statement 36-3, 36-8
WidgetTheme, attribute 90-7
width
  changing for line 40-23
  sensor contact line 43-16
wildcard 74-7
wind 1-14
  direction 11-6
  moving flags and windsocks 90-21
  offset 10-2
  speed 11-6
  speed and direction 11-9
WindDirPart 90-21

window
Add Visualizer Definition Attribute 88-12
adding 82-2
attributes, editing 82-6
channel, adding 82-4
closing 28-4
Event 8-10, 8-13
location, changing 82-6
Multiple Object Information 16-8
new 4-13
observer 50-6
opening 28-12
printing 4-14
property 82-6
removing 82-5
secondary 4-13
selecting simulation objects in 15-5
size, changing 82-6
specifying 5-3
Terrain Profile 56-2
types 82-2
window layout
changes, saving 83-6
choosing 83-5
creating 83-4
default
clearing 83-6
setting 83-5
deleting 83-7
restoring factory 83-7
reverting 83-6
updating 83-6
Window Layout Manager dialog box 83-4
Window Name attribute 82-6
Window Type attribute 82-6
Windows, installing on 2-2
windsock, movement in wind 90-21
WindSpeed parameter 90-21
WindState parameter 90-21
WindSwitchPart 90-21
wireframe mode 55-8
world coordinates 51-3
writing
entity plan 37-3
global plan 37-27
plan, for multiple simulation objects 37-41
plans, remote entities 37-41
WRM Entity Model Specification 92-1, 92-10

X-Y-Z

-x command-line option 5-8, 5-9, 5-13, 5-15, 5-19, 5-22, 63-6
XML 59-3
include files 59-6
RTI configuration file 2-10
XML file
plug-in 4-36
deleting 4-39
XR mode 50-10
XR scaling 44-5
xtr file 12-10
XXX_SpeedTrees.xml 60-18
yaw 51-3
Yaw updater 90-13
yellow icon 9-16
yellow triangle icon 16-27
Z Far 82-7
Z Near 82-7
Z-fighting 5-7, 52-7, 57-14, 82-10, 82-16
Zig Zag to Heading dialog box 26-29
Zig Zag to Heading task 26-29
zone
time 7-12
weather 11-13
zoom
ocean, display problems 51-23
scale factor, LOD 51-21
scenario, on load 51-23
SpeedTree LOD 5-7
switching from 3D to 2D 50-2
to entity extents 51-23
to extents 51-23
zooming
observer 51-19
2D 51-21
3D 51-20
performance 51-22
sensors 34-40

VR-Forces Quick Reference Card

Table 1:vrfSim Command-Line Options

Option	Description
(-a | --appNumber) *application_number*	Specifies the application number.
--additionalSystemScriptsDirectory *directory*	Specifies a directory where VR-Forces will look for system scripts when it populates the Scripted Task dialog box script list.
--appDataDir *directory*	Specifies the location of the appData directory.
--appIdRange *lower-upper*	Specifies a range of application IDs that represent objects generated by VR-Forces.
(-B | --batchScenarioFileName) *batch_file*	Runs the specified batch file. (For use with one back-end only.)
(-c | --frontEndPID) *PID*	The process ID for the front-end when running in combined mode.
--countryCodesMappingFile *file_name*	Specifies an alternative file that maps country codes to country names and DIS enumeration values.
(-d | --settingsFile) *file_name*	Specifies a configuration file to load instead of *vrfSimSettings.xml*.
--daemon	Starts that back-end as a daemon process (Linux only).
--dataDir *directory*	Specifies the data directory for the back-end.
--defaultObjectTimeout *timeout*	Specifies the timeout interval for objects.
--diGuyAnimationsFile *file_name*	Specifies an additional configuration file for DI-Guy animations.
--diGuyCharacterDataFile *file_name*	Specifies an additional configuration file for DI-Guy character data.
--disableParallelTick	Execute all tick code serially.
--doNotLoadVrfPlugins	Do not load any VR-Forces plug-ins from the *./plugins* directory.
--enableChannel *channel*	Enables a specific channel for the channel-specific output, even if it is disabled in *channelSettings.mtl*.
--fileNotifyLevel *level*	Notify level in log file to use.
--fullyLoadTerrain	For paged terrains, load all pages.
(-h | --help)	Displays help information in a console window.
--hostAddressString *address*	The IP address of the host computer. (Device Address in Launcher.)
(-i | --sessionId) *session_ID*	Specifies a session ID.
(-L | --scenarioFileName) "*scenario_pathname*"	Loads the specified scenario. Mutually exclusive with the -T option.
--loadPlugin *plugin_file*	Specifies one or more plug-ins to load. (Accepted multiple times.)
--logFileName *filename*	Log file to use.
--logFrameRateStatistics	Specifies that the back-end should log frame rate statistics.
--msdlSIDCMappingFile *file_name*	Specifies an alternative file for mapping SIDC codes to DIS enumerations.
(-n | --notifyLevel) *notification_level*	Sets the level for messages written to the console or the VR-Forces log file (*vrf.log*).
--nonVrfObjectsUUIDScheme *scheme*	Sets the UUID scheme for non VR-Forces objects. Values are entity-identifier, global-identifier, marking-text.
--numCallbackThreads *threads*	Number of threads in the callback queue.

Copyright © 2019 VT MAK. All rights reserved. Document ID: VRF-4.7-QRC-190207

Table 1:vrfSim Command-Line Options

Option	Description
`--outputLogFile` *file_name*	The file to use to write out application output.
`(-q | --doNotUseConsole)`	Specifies that all *vrfSim* output go to the log file.
`(-r | --startInRunMode)`	Starts the back-end in run mode.
`(-s | --siteId)` *site_id*	Specifies the site ID.
`--setMainThreadToHighPriority`	If enabled sets the VR-Forces main thread to high priority. Windows only.
`--simulationModelSet` *sms*	Specifies the simulation model set.
`--startMinimized`	Minimizes the vrfSim console at startup.
`--suppressDataAvailableException`	Do not throw an exception when dataAvailable is not checked in non-blocking terrain calls.
`(-T | --terrainDatabase)` *terrain_database*	Specifies a terrain database to load in the back-end. Mutually exclusive with the `-L` option.
`--targetFrameRate` *rate*	Specifies the frame rate above which vrfSim will yield CPU time.
`--useAbsoluteTimeStamps`	Use absolute time stamps instead of relative time stamps.
`--userDataDir` *directory*	Specifies the user data directory for the back-end.
`(-v | --version)` *version*	Displays version information.
`--waitQueue`	Turn off main thread participation in parallel algorithms.
`(-- | --ignore_rest)`	Ignore all command-line options that are listed after this argument.
HLA Only	
`(-f | --fomMapperLib)` *libname*	FOM Mapper library name.
`(-F | --fedFileName)` *fedfile_name*	FED file name.
`--fomMapperInitData` *data*	FOM Mapper initialization data.
`--fomModules` *module*	Specifies an HLA Evolved FOM module. Can be used multiple times.
`--ignoreAdvisories`	Enables or disables HLA advisory messages. Default: true.
`--mimModule`	Specifies an HLA Evolved MIM module.
`(-N | --federateName)` *federate_name*	Specifies a unique name that can be assigned to the federate. Default: VR-Forces. (HLA Evolved only)
`--noRtiCompilerCheck`	Disable RTI Compiler version check.
`(-p | --federateType)` *type*	Distinguish different categories of federates.
`--rprFomRevision`	RPR FOM revision for RPR FOM 2.0, draft 17.
`--rprFomVersion` *version_number*	RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017 or 2.0)
`-S | --localSettingsDesignator` *designator*	Specifies a string that is passed to the RTI during federate connect when using HLA Evolved.
`--timeManagement`	Enables time management for the back-end.
`(-x | --execName)` *exec_name*	Execution name. Default: VR-Link.
DIS Only	
`(-A | --destAddrString)` *address*	Destination address.

VR-Forces Quick Reference Card

Table 1:vrfSim Command-Line Options

Option	Description
`--attributePduStrategy`	Controls how VR-Forces sends DIS attribute PDUs.
`--deviceAddress` *address*	Specifies the address of the network card to use for UDP traffic.
`--disVersion`	The DIS version. Default: 5.
`--mcastTtl`	Specifies the multicast time-to-live. Default: -1.
`(-S | --multicastAddresses)` *address1 ... addressn*	Specifies one or more multicast addresses.
`(-P | --disPort)` *port*	DIS port.
`--recvBufferSize` *size*	Receive buffer size.
`--sendBufferSize` *size*	Send buffer size. Default: -1.
`--subnetMask` *mask*	Specifies the subnet mask.
`--suppressSelfReflect`	Represses self reflection of the DIS exercise connection.
`--useAsyncIO`	Use asynchronous I/O. Default: False.
`--useIpv6`	Specifies that the DIS exercise connection use IPV6.
`(-x | --exerciseId)` *ID*	Exercise ID.

Copyright © 2019 VT MAK. All rights reserved. Document ID: VRF-4.7-QRC-190207

Table 2:vrfGui Command-Line Options

Option	Description
(-- | --ignore_rest)	Ignore all command-line options after this one.
--alphaBits *bits*	Sets the number of alpha bits (0, 8).
--antiAliasing *level*	Set the anti-aliasing level (0, 2, 4, 8, or 16). Default: 4.
--appDataDir *directory*	Specifies the location of application data.
--appIdRange *range*	Application ID range of VR-Forces back-ends.
(-c | --showConsole)	Display a console window.
(-C | --autoConnect) {*filename* | *connector_name*}	Automatically connect to the network using the specified connector.
--dataDir *directory*	Specifies the location of the data directory .
--defaultSimulationModelSet *sms*	The simulation model set to use as the default when starting up the GUI.
--depthBits *bits*	Sets the depth of graphics to 24 bits or 32 bits. Default: 24.
--dis	Use the DIS protocol and enable DIS-specific options.
--disableDynamicTerrain	Disables the dynamic terrain engine.
--disableOcclusionCulling	Disables occlusion culling, which can improve performance.
--discoveredModelPreloadFileName *filename*	Specifies a file to which the list of models discovered during a session that were not preloaded is written.
--dispSetting *filename*	Specifies the display configuration to load.
--doNotCheckPluginVersions	Do not check the version in a plug-in to make sure it can load in VR-Forces.
--doNotLoadVrfPlugins	Prevents loading of plug-ins in the *./plugins* directory.
--enableVSync	Turns on VSync.
--entDispSetting *filename*	Specifies the entity display setting to load.
--entityDefsDir *directory*	Loads the *.hier* and *.leaf* files in the specified directory.
--envSetting *filename*	Specifies the environment configuration file to load.
(-E | --entTypeMap) *filename*	Load the specified entity type mapping file. The file extension determines the type of mapping.
--factoryRootDir *directory*	Set the root directory used to restore settings to factory defaults.
--fileTransporterReceivePort *port*	Port for the file transporter.
--forceTextShaping	Force all text to use HarfBuzz shaping.
--forceVBO	Forces use of vertex buffer objects in OpenGL.
--fowPerspective *force*	Shows the ground truth and spot reports perspective for the specified force.
--fullScreen	Start in full screen mode.
--frameRate *rate*	Specifies a fixed frame rate for rendering. If 0, it is variable.
(-G | --locale) *language*	Specifies the locale (for localization).
--gui *filename*	Specify GUI configuration file to load.
(-h | --help)	Displays command-line usage information.
--highestPriority	Changes application and the main render thread to the highest available priority.
--hla13	Use HLA 1.3 and enable HLA-specific options.

Table 2:vrfGui Command-Line Options

Option	Description
`--hla1516`	Use HLA 1516 protocol and enable HLA-specific options.
`--hla1516e`	Use HLA Evolved protocol and enable HLA-specific options.
`(-I | --unbatchedRendering)`	Disables instance rendering.
`(-K | --disableKDTrees)`	Disable creation of KD trees for intersections.
`(-l | --logFileName)` *filename*	Specifies the log file name.
`--layoutSettingsFile` *filename*	Specifies the GUI layout settings file to use. *filename* specifies the base filename. It is prefixed with *default_* and the extension *.uisx* is added.
`--loadPlugin` *filename*	Specifies plug-ins to load. Accepted multiple times.
`--loadSimulationObjectGroup` *group*	Creates a new scenario on the simulation object group editing terrain and creates an instance of the simulation object group on that terrain.
`(-L | --loadObservers)` *filename*	Loads the saved observers in the specified file. Can be used multiple times.
`--mainScreenNum` *num*	Specifies the display that the main window should use.
`--mergeEntityTypeMap {`*directory* | *filename*`}`	Merges all entity type mapping files (*.*metx*) in a directory or individual entity type mapping files. Can be used multiple times.
`--mergeHierarchyEntityDef` *filename*	Merges the specified *.hier* file into the default hierarchy. Merging is additive.
`--mergeHierarchyTacticalGraphicsDef` *filename*	Merges the specified *.hier* file into the default hierarchy. Merging is additive.
`--mergeHierarchyUnitDef` *filename*	Merges the specified *.hier* file into the default hierarchy. Merging is additive.
`--mergeLeafEntityDef` *filename*	Merges the specified leaf file into the default entity element definitions.
`--mergeLeafTacticalGraphicsDef` *filename*	Merges the specified leaf file into the default tactical graphics element definitions.
`--mergeLeafUnitDef` *filename*	Merges the specified leaf file into the default unit element definitions.
`--mergeModelDefs {`*directory* | *filename*`}`	Merges all model definition files (*.*ommx*) in the specified directory or individual model definition files. Can be used multiple times.
`--mergeTacticalGraphicsTypeMap {`*directory* | *filename*`}`	Merges all tactical graphics type mapping files (*.*metx*) in a directory or individual tactical graphics type mapping files. Can be used multiple times.
`--mergeUnitTypeMap {`*directory* | *filename*`}`	Merges all unit type mapping files (*.*metx*) in a directory or individual unit type mapping files. Can be used multiple times.
`--modelPreloadFileName` *filename*	Specifies a file that has a list of models to preload.
`(-n | --notifyLevel)` *notify_level*	Specifies the notification level for application messages. Range: 0-4.
`--noAppDataEntityDefs`	Do not load the entity element definitions in *./appData/definitions*.
`--noAppDataTacticalGraphicsDefs`	Do not load the tactical graphics element definitions in *./appData/definitions*.
`--noAppDataUnitDefs`	Do not load the unit element definitions in *./appData/definitions*.
`--noAppDataEntityTypeMap`	Do not load entity type mappings in *./appData/settings/<application>*.
`--noAppDataModelDefs`	Do not load model definitions in *./appData/definitions*.
`--noAppDataTacticalGraphicsTypeMap`	Do not load tactical graphics type mappings in *./appData/settings/<application>*.

Copyright © 2019 VT MAK. All rights reserved. Document ID: VRF-4.7-QRC-190207

Table 2:vrfGui Command-Line Options

Option	Description
`--noAppDataEntityTypeMap`	Do not load unit type mappings in *./appData/settings/<application>*.
`--nonVrfObjectsUUIDScheme` *scheme*	Sets the UUID scheme for non VR-Forces objects. Values are entity-identifier, global-identifier, marking-text. Default: entity-identifier.
`(-O | --hasNTPSync)`	Enables synchronized ocean between different display engines. System clocks must be synchronized.
`--plugin` *filename*	Load the specified plug-in. Can be used multiple times.
`--pseudoFullScreen`	Use pseudo full screen for full screen mode.
`--pvd`	Start the VR-Vantage front-end with only 2D modes available.
`--reportedHostAddressString` *address*	Specifies how VR-Forces reports its IP address to another VR-Forces application for the purpose of file transfers.
`(-S | --suppressHatIntersectOnAttach)`	Suppress the periodic HAT intersection when attached.
`--scenarioFile` *filename*	Specifies a scenario to load.
`--stealth`	Start the VR-Vantage front-end with only 3D modes available.
`--stencilBits` *bits*	Sets the number of stencil bits (0 or 8). Default: 0.
`--tcpSendTimeout`	Sets the timeout (in seconds) for the master display engine to wait for a remote display engine before giving up on a send. Default: 30 seconds.
`(`*terrain_MTF_file* `|` *scene_MSF_file*`)`	Specifies a terrain (MTF) or scene (MSF) file to load.
`--unplug` *filename*	Do not load the specified plug-in. Can be used multiple times.
`--useAbsoluteTimeStamps`	Use absolute time stamps instead of relative time stamps.
`--useFileCommChannel` *channel*	The communication file to be used as a base for communicating text based commands from external applications.
`--userDataDir` *directory*	Specifies the user data directory.
`(-v | --version)`	Display version information and exit.
`(-z | --useReverseZBuffer)`	Specifies use of reverse Z-buffering, which can reduce Z-fighting.
`(-Z | --ignoreZoomForSpeedtreeLod`	Disable zoom for SpeedTree LOD.
HLA Only	
`(-a | --appNumber)` *number*	Application number.
`(-f | --fomMapperLib)` *libname*	FOM Mapper library name.
`(-F | --fedFileName)` *fedfile_name*	FED file name.
`--fomMapperInitData` *data*	FOM Mapper initialization data.
`--fomModules` *module*	Specifies a FOM Module. Can be used multiple times. (HLA Evolved only.)
`(-H | --hostAddressString)` *address*	Specifies the host address.
`(-i | --sessionId)` *ID*	Specifies the session ID for this instance of the front-end.
`(-N | --federateName)` *federate_name*	Federate name. Default: VR-Forces. (HLA Evolved only.)
`(-p | --federateType)` *type*	The federate type distinguishes different categories of federates.
`--rprFomRevision`	RPR FOM revision.
`--rprFomVersion` *version_number*	RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0)
`(-s | --siteId` *ID*	Site ID.
`--localSettingsDesignator` *designator*	Specifies a string that is passed to the RTI during federate connect. (HLA Evolved only.)
`--ignoreAdvisories`	Enables or disables HLA advisory messages. Default: true.

Table 2:vrfGui Command-Line Options

Option	Description
`--useGeographicDdm`	Specifies use of geographic DDM if you are using the MAK RTI. Default: false.
`(-x | --execName)` *exec_name*	Execution name. Default: VR-Link.
DIS Only	
`(-a | --appNumber)` *number*	Application number.
`(-A | --destAddrString)` *address*	Destination address.
`--defaultObjectTimeout` *timeout*	Specifies the timeout interval for objects.
`--deviceAddress` *address*	Specifies the address of the network card to use for UDP traffic.
`--disVersion`	The DIS version. Default: 5.
`(-H | --hostAddressString)` *address*	Specifies the host address. This is usually the same as the device address.
`(-i | --sessionId)` *ID*	Specifies the session ID for this instance of the front-end.
`--mcastTtl`	Specifies the multicast time-to-live. Default: -1.
`--multicastAddresses` *address1 ... addressn*	Specifies one or more multicast addresses.
`(-P | --disPort)` *port*	DIS port.
`--recvBufferSize` *size*	Receive buffer size.
`(-s | --siteId` *ID*	Site ID.
`--sendBufferSize` *size*	Send buffer size. Default: -1.
`--subnetMask` *mask*	Specifies a subnet mask.
`--useAsyncIO`	Use asynchronous I/O. Default: False.
`--useIpv6`	Use the ipv6 protocol, if available.
`(-x | --exerciseId)` *ID*	Exercise ID.

Table 3:Mouse Button Mappings

Navigation Action	Mouse Action	Context
Attach to entity.	**Shift** + left-click	Entity.
Attach to prop.	**Ctrl** + **Shift** + left-click	Prop.
Deselect all.	Left-click	Terrain or sky.
Drag terrain.	Left-click + drag	Terrain or prop.
Object selected, others unselected.	Left-click	Selected entity or prop.
Select entity or prop.	**Ctrl** + left-click	Unselected entity or prop.
Select object and unselect all others.	Left-click	Unselected entity or prop.
Teleport to location.	**Shift** + left-click	Terrain.
Teleport to prop.	**Shift** + left-click	Prop.
Teleport to entity.	**Ctrl** + **Shift** + left-click	Entity.
Unselect.	**Ctrl** + left-click	Selected entity or prop.

Copyright © 2019 VT MAK. All rights reserved. Document ID: VRF-4.7-QRC-190207

Table 3:Mouse Button Mappings

Navigation Action	Mouse Action	Context
Display context-sensitive menu.	Right-click	Entity or prop.
3D Only		
Drag the view.	Middle mouse button drag	All.
Drag heading and pitch (mouse look).	Right Drag	All.
Orbit drag.	**Ctrl** + left-click + drag	Terrain or prop.
Zoom in (magnification).	**Ctrl** + mouse wheel forward	All.
Zoom out (magnification).	**Ctrl** + mouse wheel backward	All.
Reset magnification.	Middle mouse button	All.
Move observer forward.	Mouse wheel forward	All.
Move observer back.	Mouse wheel back	All.
Move observer level observer forward.	**Shift** + mouse wheel forward.	All.
Move observer level observer backward.	**Shift** + mouse wheel backward.	All.
Move flashlight beam.	Right mouse button drag	All.
Expand flashlight beam.	**Alt** + mouse wheel forward.	All.
Contract flashlight beam.	**Alt** + mouse wheel backward.	All.
2D Only		
Rotate terrain.	Right-click + drag	All. Orient North disabled.
2D Zoom in.	Mouse wheel forward; Middle button + drag	All.
2D Zoom out.	Mouse wheel backward; Middle button + drag	All.

In keyboard mappings, K = numeric keypad (K8 means press Keypad 8). To use the numeric keypad for navigation, NumLock must be on.

Table 4:Observer Keyboard Mappings

Observer Movement	Observer	Level Observer	Laptop	2D Level Observer
Move forward.	**w**	unmapped	**w**	unmapped
Move back.	**s**	unmapped	**s**	unmapped
Move left.	**a**	unmapped	**a**	unmapped
Move right.	**d**	unmapped	**d**	unmapped
Move up.	**q**, **Ctrl** + **q**	unmapped	**q**, **Ctrl** + **q**	unmapped
Move down.	**e**, **Ctrl** + **e**	unmapped	**e**, **Ctrl** + **e**	unmapped
Move forward level observer.	unmapped	**w**	unmapped	**w**
Move back level observer.	unmapped	**s**	unmapped	**s**
Move left level observer.	unmapped	**a**	unmapped	**a**
Move right level observer.	unmapped	**d**	unmapped	**d**
Move up level observer.	unmapped	**q**, **Ctrl** + **q**	unmapped	unmapped

VR-Forces Quick Reference Card

Table 4:Observer Keyboard Mappings

Observer Movement	Observer	Level Observer	Laptop	2D Level Observer
Move down level observer.	unmapped	**e**, **Ctrl**+**e**	unmapped	unmapped
Move attached context forward.	**Shift**+**w**	**Shift**+**w**	**Shift**+**w**	**Shift**+**w**
Move attached context back.	**Shift**+**s**	**Shift**+**s**	**Shift**+**s**	**Shift**+**s**
Move attached context left.	**Shift**+**a**	**Shift**+**a**	**Shift**+**a**	**Shift**+**a**
Move attached context right.	**Shift**+**d**	**Shift**+**d**	**Shift**+**d**	**Shift**+**d**
Move attached context up.	**Shift**+**q**	**Shift**+**q**	**Shift**+**q**	unmapped
Move attached context down.	**Shift**+**e**	**Shift**+**e**	**Shift**+**e**	unmapped
Level yaw left.	K **4**	K **4**	Left arrow	Left arrow
Level yaw right.	K **6**	K **6**	Right arrow	Right arrow
Level pitch up.	**Shift**+**j**	**Shift**+**j**	**Shift**+**j**	unmapped
Level pitch down.	**Shift**+**k**	**Shift**+**k**	**Shift**+**k**	unmapped
Level roll right.	**Shift**+**i**	**Shift**+**i**	**Shift**+**i**	unmapped
Level roll left.	**Shift**+**m**	**Shift**+**m**	**Shift**+**m**	unmapped
Yaw left.	**Shift**+**h**	**Shift**+**h**	**Shift**+**h**	**Shift**+**h**
Yaw right.	**Shift**+**l**	**Shift**+**l**	**Shift**+**l**	**Shift**+**l**
Pitch up.	K **8**	K **8**	↑	unmapped
Pitch down.	K **2**	K **2**	↓	unmapped
Roll right.	K **9**	K **9**	K **9**	unmapped
Roll left.	K **3**	K **3**	K **3**	unmapped
Orbit right.	**Ctrl**+**d**, →	**Ctrl**+**d**, →	**Ctrl**+**d**	unmapped
Orbit left.	**Ctrl**+**a**, ←	**Ctrl**+**a**, ←	**Ctrl**+**a**	unmapped
Orbit up.	**Ctrl**+**w**, ↑	**Ctrl**+**w**, ↑	**Ctrl**+**w**	unmapped
Orbit down.	**Ctrl**+**s**, ↓	**Ctrl**+**s**, ↓	**Ctrl**+**s**	unmapped
Look up.	**Ctrl**+K **8**	**Ctrl**+K **8**	**Ctrl**+K **8**	**Ctrl**+K **8**
Look down.	**Ctrl**+K **2**	**Ctrl**+K **2**	**Ctrl**+K **2**	**Ctrl**+K **2**
Look left.	**Ctrl**+K **4**	**Ctrl**+K **4**	**Ctrl**+K **4**	**Ctrl**+K **4**
Look right.	**Ctrl**+K **6**	**Ctrl**+K **6**	**Ctrl**+K **6**	**Ctrl**+K **6**
Look tilt left.	**Ctrl**+K **3**	**Ctrl**+K **3**	**Ctrl**+K **3**	**Ctrl**+K **3**
Look tilt right.	**Ctrl**+K **9**	**Ctrl**+K **9**	**Ctrl**+K **9**	**Ctrl**+K **9**
Zoom In	unmapped	unmapped	unmapped	**Ctrl** + **e**
Zoom Out	unmapped	unmapped	unmapped	**Ctrl** + **q**
Suppress constrain to terrain.	**x**	**x**	**x**	**x**
Water Impact	unmapped	unmapped	unmapped	unmapped
Reset observer: In absolute mode, go to the center of the scene. If attached, reset the attach position and orientation.	**Space**	**Space**	**Space**	**Space**
Toggle Speed Scaling Mode (MSL or AGL)	unmapped	unmapped	unmapped	unmapped

Copyright © 2019 VT MAK. All rights reserved. Document ID: VRF-4.7-QRC-190207

Table 4:Observer Keyboard Mappings

Observer Movement	Observer	Level Observer	Laptop	2D Level Observer
Level observer.	K 5	K 5	Shift+Space+↓	K 5
Toggle Force North Orientation	unmapped	unmapped	unmapped	unmapped
Toggle Orient To Target Constraint	unmapped	unmapped	unmapped	unmapped
Toggle Flashlight	F	F	F	F
Reset the camera.	K 0	K 0		K 0
Increase View Magnification	=	=	=	unmapped
Decrease View Magnification	-	-	-	unmapped
Reset View Magnification	0	0	0	unmapped
Increase View Resolution	unmapped	unmapped	unmapped	=
Decrease View Resolution	unmapped	unmapped	unmapped	-
Detach.	z	z	z	z
Toggle Hide Model On Attach	unmapped	unmapped	unmapped	unmapped
Attach to the next entity allowed by the attachment filter.	Period (.)	Period (.)	Period (.)	Period (.)
Attach to next entity or prop.	>	>	>	>
Attach to the previous entity allowed by the attachment filter.	Comma (,)	Comma (,)	Comma (,)	Comma (,)
Attach to previous entity or prop.	<	<	<	<
Set Mimic Attach Type	unmapped	unmapped	unmapped	unmapped
Set Tether Attach Type	unmapped	unmapped	unmapped	unmapped
Next attach mode.	Shift+z	Shift+z	Shift+z	Shift+z
Increase linear speed.	K 7	K 7	K 7	K 7
Decrease linear speed.	K 1	K 1	K 1	K 1
Increase orbit speed.	K +	K +	K +	K +
Decrease orbit speed.	K -	K -	K -	K -
Increment Rotation Speed Gain	unmapped	unmapped	unmapped	unmapped
Decrement Rotation Speed Gain	unmapped	unmapped	unmapped	unmapped
Toggle speed scaling on or off.	K *	K *	K *	K *
Toggle Prop Transparency	T	T	T	T
Toggle Terrain Scaling	unmapped	unmapped	unmapped	unmapped
Increment Terrain Scale	unmapped	unmapped	unmapped	unmapped
Decrement Terrain Scale	unmapped	unmapped	unmapped	unmapped
Toggle Track Histories	unmapped	unmapped	unmapped	unmapped
Toggle Entity Labels	L	L	L	L
Toggle Rotate Entity Symbols	unmapped	unmapped	unmapped	unmapped
Toggle Tactical Graphics	Shift+G	Shift+G	Shift+G	Shift+G
Toggle Entity Height Above Terrain Lines	unmapped	unmapped	unmapped	unmapped
Toggle Tactical Graphics HAT Lines	unmapped	unmapped	unmapped	unmapped

Table 4:Observer Keyboard Mappings

Observer Movement	Observer	Level Observer	Laptop	2D Level Observer
Toggle Entity Height Above Terrain Mode	unmapped	unmapped	unmapped	unmapped
Toggle Tactical Graphics Height Above Terrain Mode	unmapped	unmapped	unmapped	unmapped
Toggle constrain to terrain on or off.	**/**	**/**	**/**	**/**
Toggle Constrain To Center	**U**	unmapped	unmapped	unmapped
Clear the current movement mode. This key is an escape mechanism if keys are not behaving as you expect. This is typically due to a loss of the window context.	**c**	**c**	**c**	**c**
Next Observer Mode	**]**	**]**	**]**	**]**
Previous Observer Mode	**[**	**[**	**[**	**[**
Toggle view controls on or off.	**Shift**+**c**	**Shift**+**c**		**Shift**+**c**
Clear All Trajectory Histories.	unmapped	unmapped	unmapped	unmapped
Toggle OSG Framerate statistics	**F2**	**F2**	**F2**	**F2**
Print OSG statistics	**Shift**+**[**	**Shift**+**[**	**Shift**+**[**	**Shift**+**[**

Table 5:vrfLauncher Command-Line Options

Option	Description
(-B | --backend)	Start vrfLauncher for the back-end only.
B--	Following --, arguments after B-- are passed only to the back-end.
(-C | --config)	Open the Simulation Connections Configuration tool.
-debug	Following B-- or F--, appends a d to vrfGui or vrfSim to run the debug version. (Windows only.)
(-F | --frontend)	Start vrfLauncher for the front-end only.
F--	Following --, arguments after F-- are passed only to the front-end.
(-G | --locale) *locale*	Specifies the language to use.
--guiArgs	Arguments after --guiArgs are passed only to the front-end.
(-h | --help)	Displays help information in a console window.
--simArgs	Arguments after --simArgs are passed only to the back-end.
--usePredefinedConnection *connection*	Specify a connection name, such as DIS localhost, so that the vrfLauncher can launch without needing to display the Simulation Connections dialog box.
(-v | --version)	Displays version information.
--	Pass all arguments after this to vrfGui and vrfSim, except per B-- or F--.

Copyright © 2019 VT MAK. All rights reserved. Document ID: VRF-4.7-QRC-190207



Link - Simulate - Visualize

150 CAMBRIDGE PARK DRIVE, 3RD FLOOR CAMBRIDGE, MA 02140 617.876.8085 www.mak.com