

VR-Forces

Users Guide

VT MÄK
A company of VT Systems



VR-Forces

Users Guide

VT MÄK
A company of VT Systems

Copyright © 2016 VT MÄK
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 MÄK.

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

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 liv.

VT MÄK
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-4.4.1-1-160616



Contents

Preface

How This Manual is Organized xliii
VR-Forces Documentation xlviii
MÄK Products xlix
How to Contact Us lii
Document Conventions liii
Mouse Button Naming Conventions liv
Third Party Licenses liv
Boost License liv
libXML and libICONV lv
Lua lv
Freefont OpenType Font Set lv
Third-Party Licenses for VR-Vantage Applications lv

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 1-6
1.1.2. Realistic Display of Vehicles and Terrain (3D View) 1-7
1.1.3. Create Complex Scenarios 1-8
1.1.4. Simulation Object Types Supported 1-9
1.1.5. Mission Planning 1-11
1.1.6. Simulation Object Tasks 1-11
1.1.7. Scripted Tasks 1-12
1.1.8. Tactical Graphics 1-12
1.1.9. Terrain Agility and Composability 1-13
1.1.10. Flexible, Intuitive Graphical User Interface 1-14
1.1.11. Overlays 1-14

1.1.12. Behaviors 1-14
1.1.13. Observer Attach Modes 1-15
1.1.14. Special Effects and Simulation Object Information Visualization 1-15
1.1.15. Dynamic Ocean 1-15
1.1.16. Lighting Effects 1-15
1.1.17. Accurate Vehicle Positioning 1-16
1.1.18. Batch Mode Operation 1-16
1.1.19. Remote Control 1-16
1.2. The VR-Forces Toolkit 1-17
1.2.1. Plug-in Architecture 1-17
1.3. DI-Guy 1-18
1.4. Support for External Communications Servers 1-18
1.5. Helpful Utilities 1-19
1.6. Third-Party Software and Content 1-20
1.6.1. SilverLining 1-20
1.6.2. Triton Ocean SDK 1-21
1.6.3. GL Studio 1-21
1.6.4. SpeedTree 1-22
1.6.5. 3D Models, Terrain, and Graphical Content 1-22
1.6.6. OpenSceneGraph 1-23
1.6.7. osgEarth 1-23
1.7. Distributed Simulation Standards Supported 1-23

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 2-2
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 MÄK 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 MÄK 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
2.5.4. Merging Translation 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 3-11
3.5. Interactions 3-12
3.6. Advanced Terrain Navigation 3-12
3.6.1. How Navigation Data is Generated 3-12
3.6.2. Path Finding 3-14

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 4-10
4.2. The VR-Forces Window 4-12
4.2.1. Opening New Windows 4-14
4.2.2. Printing the VR-Forces Display 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 4-16
4.3.3. Configuring Session Messages and Join at Startup 4-17
4.4. Using HLA Time Management 4-18
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. Closing a Terrain 4-23
4.6. Managing VR-Forces Settings 4-23
4.6.1. Synchronizing Settings Among VR-Forces Installations 4-25
4.6.2. Global Settings and Observer-Specific Settings 4-25
4.7. Exiting VR-Forces 4-25
4.8. Configuring Simulation Connections 4-26
4.8.1. Opening the Simulation Connections Configuration Dialog Box 4-26
4.8.2. Adding a Simulation Connection 4-27
4.8.3. Editing a Simulation Connection 4-28
4.8.4. Simulation Connection Parameters 4-28
4.8.5. Deleting a Simulation Connection 4-30
4.8.6. Configuring Auto Connect 4-31
4.8.7. Displaying Connection Information 4-32

4.9. Managing Plug-ins 4-33
4.9.1. Loading Plug-ins 4-33
4.9.2. Adding a Plug-in 4-36
4.9.3. Specifying the DLLs for a Plug-in 4-37
4.9.4. Adding a Plug-in Configuration 4-38
4.9.5. Deleting a Plug-in Configuration 4-38
4.9.6. Deleting a Plug-in 4-39
4.9.7. Viewing a List of Loaded Plug-ins 4-39
4.10. The VR-Forces Log Files 4-40

Chapter 5. Command-line Options

5.1. Command-Line Options for vrfGui 5-3
5.2. Command-line Options for vrfSim 5-8
5.3. Command-line Options for vrfLauncher 5-13
5.3.1. Running in Combined Mode from the Command Line 5-14
5.3.2. Using the -- Command-line Argument 5-14
5.4. Protocol-Independent Command-line Options 5-14
5.4.1. Specifying the Site ID and Application Number 5-14
5.4.2. Specifying the Language to Use in the GUI 5-15
5.4.3. Setting the Notification Level 5-15
5.4.4. Loading Plug-ins 5-16
5.5. Command-line Options for HLA Federations 5-16
5.5.1. Specifying the Federation Execution 5-16
5.5.2. Specifying the FED File 5-17
5.5.3. Specifying the RPR FOM Version 5-17
5.5.4. Specifying a FOM Mapper 5-17
5.5.5. Specifying the RPR FOM Revision 5-17
5.5.6. Specifying FOM Mapper Initialization Data 5-18
5.5.7. Specifying a FED File That is Appropriate for the FOM Mapper 5-18
5.5.8. Enabling Time Management 5-18
5.6. Command-line Options for DIS Exercises 5-19
5.6.1. Specifying the Port Number 5-19
5.6.2. Specifying Point-to-Point or Multicast Operation 5-19
5.6.3. Specifying the Multicast Time-to-Live 5-19
5.6.4. Using Asynchronous I/O 5-19
5.6.5. Specifying the Exercise ID 5-20
5.6.6. Specifying the DIS Version 5-20

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 6-5
6.3. Displaying Performance Statistics 6-6
6.3.1. Displaying the VR-Forces Function Profiler 6-7

6.3.2. Displaying OSG Statistics 6-8
6.4. Filtering Simulation Objects Using Interest Management 6-8
6.4.1. Enabling Interest Management 6-8
6.4.2. Configuring Interest Management 6-9
6.5. Miscellaneous Performance Configuration Options 6-11
6.5.1. Limiting Use of Spot Reports 6-11
6.5.2. Using Asynchronous I/O for DIS Exercises 6-11
6.5.3. Using Incremental Compiling with Streamed Data 6-11
6.6. Setting the Tick Rate 6-12
6.6.1. Specifying the Tick Rate for Components 6-12
6.6.2. Tuning the Network Interface 6-13
6.6.3. Tuning the State Repository Tick Rate 6-13
6.7. Configuring Graphics Quality 6-13
6.8. Balancing Visual Quality Against Network Performance 6-14
6.8.1. Enabling Configuration of Performance Options 6-15
6.9. Trajectory Smoothing 6-16
6.9.1. Configuring Trajectory Smoothing 6-17
6.10. Tuning the Target Frame Rate 6-17
6.11. Coloring Draw Calls 6-18
6.12. Configuring VSync 6-19

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
7.1.4. Importing (Merging) Scenarios 7-13
7.1.5. Importing and Exporting MSDL 7-14
7.1.6. Scenario Building Blocks 7-16
7.2. Loading a Scenario 7-17
7.2.1. Loading a Recently Loaded Scenario 7-18
7.2.2. Loading a Scenario from the Command Line 7-18
7.2.3. Load Balancing a Scenario 7-19
7.2.4. Displaying Scenario Information 7-21
7.2.5. Editing the Scenario Description 7-23
7.2.6. Sample Scenarios 7-23
7.3. Running a Scenario 7-24
7.3.1. Changing the Simulation Speed 7-24
7.3.2. Pausing a Scenario 7-25
7.3.3. Rewinding a Scenario 7-25
7.3.4. Closing a Scenario 7-27
7.4. Saving a Scenario 7-27
7.4.1. Saving a Previously Saved Scenario 7-29
7.4.2. Saving a Scenario to a New Name 7-29

7.5. Checkpoints and Snapshots 7-29
7.5.1. Saving Checkpoints 7-30
7.5.2. Deleting Checkpoints 7-33
7.5.3. Creating Snapshots 7-34
7.5.4. Clearing the Snapshot List 7-35
7.6. Pausing a Scenario Automatically 7-35
7.6.1. Setting the Scenario End Time for a New Scenario 7-35
7.6.2. Setting the Scenario End Time for an Open Scenario 7-36
7.7. Running VR-Forces in Batch Mode 7-36
7.7.1. The Batch File 7-37
7.7.2. Creating a Batch File 7-37
7.7.3. Editing a Batch File 7-38
7.7.4. Running VR-Forces in Batch Mode 7-40
7.7.5. Recording Batch Scenarios 7-41
7.8. Sending Run and Pause Messages to Simulation Participants 7-42
7.9. Recording VR-Forces Simulations with the MÄK Data Logger 7-42

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. Starting a Scenario Event from a Plan 8-11
8.5. Ending a Scenario Event 8-12
8.6. Sorting the Event List 8-13
8.7. Deleting a Scenario Event 8-13
8.8. Showing and Hiding Scenario Events 8-14

Chapter 9. Target Detection and Combat Features

9.1. Displaying Simulation Objects Based on 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 9-8
9.1.4. Applying Spot Reports to Tactical Graphics 9-9
9.1.5. Displaying Labels for Spot Reports 9-9
9.1.6. Using Spot Reports in Tasks 9-9
9.2. Managing Force Hostility Relationships 9-10
9.2.1. Changing a Force's Hostility in a Plan 9-12
9.3. Detecting Targets 9-13
9.3.1. Target Detection and Spot Reports 9-15
9.3.2. Lasing Targets 9-15
9.4. Using Sonar 9-17
9.4.1. Propulsion Noise and Sonar 9-18
9.5. Launching Counter Measures (Chaff and Flare) 9-19

9.6. Modeling Artillery Munitions 9-21
9.7. Taking Damage from Munitions 9-22

Chapter 10. Indirect Fire and Ballistic Missiles

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 10-8
10.2.2. Editing Missile Target Events 10-10
10.2.3. Deleting Missile Target Events 10-10

Chapter 11. Setting Environment Conditions

11.1. Introduction to Environment Conditions 11-2
11.2. Setting the Date and Time of Day 11-2
11.2.1. Setting the Date 11-3
11.2.2. Setting the Time of Day 11-4
11.2.3. Choosing the Illumination Model 11-4
11.3. Specifying the Environment Conditions 11-5
11.3.1. Adding an Environment Condition 11-7
11.3.2. Editing an Environment Condition 11-7
11.3.3. Deleting an Environment Condition 11-8
11.4. Setting Weather Conditions 11-8
11.4.1. Setting the Wind Speed and Direction 11-9
11.4.2. Setting Visibility (Fog) 11-10
11.4.3. Specifying Precipitation Type and Intensity 11-11
11.4.4. Specifying Cloud Cover 11-11
11.5. Creating Local Weather Zones 11-12
11.5.1. Editing a Weather Zone's Environment 11-14
11.6. Configuring Marine Conditions 11-15
11.6.1. Enabling Marine Effects 11-16
11.6.2. Configuring Marine Conditions 11-16
11.6.3. Enabling Tidal Stream Wakes 11-18
11.6.4. Experimental Ocean Features 11-19
11.7. Setting the Thermocline 11-20

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 12-8
12.4. The Object Map File 12-9

12.5. Editing the Order of Battle File 12-10
12.5.1. Saving Default Parameters to the Order of Battle File 12-10
12.6. The Scenario Extras File 12-10
12.7. Temporary Scenario Directories 12-11

Chapter 13. Using a Communications Effects Server

13.1. Introduction 13-2
13.2. Configuring VR-Forces 13-2
13.2.1. Enabling the External Communication Model 13-2
13.2.2. Configuring the External Communication Model 13-3
13.3. Using VR-Forces with QualNet 13-4
13.3.1. Running VR-Forces with QualNet 13-4
13.4. Example Scenario 13-5
13.4.1. Running the Scenario 13-5

Chapter 14. Example Entity-Level Scenarios

14.1. The Breaching Scenario 14-2
14.1.1. Create the Scenario 14-3
14.1.2. Create the Minefield 14-4
14.1.3. Create the Opposing Forces 14-5
14.1.4. Create the Tactical Graphics 14-7
14.1.5. Create the Friendly Simulation Objects 14-8
14.1.6. Write the Plans 14-10
14.1.7. Save the Scenario 14-17
14.1.8. Running the Scenario 14-18
14.2. The Embarkexample Scenario 14-19
14.2.1. Create the Simulation Objects 14-20
14.2.2. Create the Tactical Graphics 14-23
14.2.3. Write the Plans for the Entities 14-25

Section III Simulation Objects

Chapter 15. Introduction to Simulation Objects

15.1. Simulation Objects 15-2
15.1.1. How Simulation Objects are Identified 15-2
15.1.2. How Simulation Objects are Organized 15-5
15.1.3. How Simulation Objects Communicate 15-6
15.1.4. VR-Forces Uses a Component Architecture 15-7
15.1.5. Simulation Object Behaviors and Tasks 15-9
15.2. Entity-Level Modeling and Aggregate-Level Modeling 15-9

Chapter 16. Creating and Placing Objects

16.1. Creating Objects 16-3
16.2. Selecting the Object to Create 16-4
16.2.1. Selecting the Object to Create on an Object Palette 16-5

16.2.2. Selecting the Object to Create on the Create Menu 16-8
16.2.3. Draw Mode 16-9
16.3. Placing Objects 16-11
16.3.1. Placing an Object Using Default Values (Click to Create) 16-11
16.3.2. Specifying an Object's Properties Before You Create It (Click to Locate) 16-13
16.4. Specifying an Object's Altitude 16-14
16.4.1. Setting Altitude Dynamically 16-15
16.4.2. Setting Altitude in the Create Object or Edit Object Dialog Box 16-15
16.4.3. Specifying the Altitude for All of the Vertices in a Route .. 16-16
16.5. Specifying an Object's Heading Dynamically 16-17
16.6. Locking the Mouse to the Object Being Created 16-18
16.7. Moving Objects 16-19
16.7.1. Dragging an Object to a New Location 16-19
16.8. Copying and Pasting Objects 16-21
16.8.1. Copying Objects 16-21
16.8.2. Pasting Objects 16-22
16.8.3. Pasting Specific Entity Characteristics 16-23
16.9. Simulation Object Groups 16-25
16.10. Creating Cultural Features 16-25
16.11. Creating Props 16-26
16.12. Adding an Object to the Favorites List 16-26

Chapter 17. Selecting Objects

17.1. The Objects List Panel 17-2
17.2. Selecting Simulation Objects, Tactical Graphics, and Props 17-4
17.2.1. Selecting Objects in the Window 17-5
17.2.2. Selecting Objects in the Objects List Panel 17-6
17.2.3. Selecting a Simulation Object in the Object Console Summary Panel 17-6
17.2.4. Unselecting Objects 17-7
17.3. Using Selection Groups 17-8
17.3.1. Creating a Selection Group 17-8
17.3.2. Selecting a Selection Group 17-10
17.3.3. Adding Objects to a Selection Group 17-10
17.3.4. Removing Objects from Selection Groups 17-11
17.3.5. Renaming a Selection Group 17-11
17.3.6. Deleting a Selection Group 17-11

Chapter 18. Viewing Information about Objects

18.1. Displaying Information About an Object 18-3
18.1.1. Viewing Information for Multiple Objects 18-10
18.2. Displaying Entity Labels 18-10
18.2.1. Displaying Text Labels for 3D Models 18-13

18.2.2. Turning Entity Labels On and Off 18-14
18.2.3. Pinning 3D Text Labels to the Window 18-14
18.2.4. Displaying Labels for 2D Icons 18-15
18.2.5. Configuring Shadowing and Background Color for 2D Labels 18-17
18.3. Using Extended Labels 18-18
18.3.1. Adding an Extended Label 18-19
18.3.2. Changing an Extended Label's Index 18-20
18.3.3. Deleting an Extended Label 18-20
18.4. Simulation Object Icons 18-21
18.4.1. Changing the Size of 2D Icons 18-22
18.4.2. Rotating 2D Icons to a Simulation Object's Heading 18-23
18.5. Displaying HAT Lines for Entities (3D Only) 18-24
18.6. Displaying Threat and Sensor Range Rings 18-25
18.6.1. Enabling and Disabling the Display of Range Rings 18-27
18.6.2. Pinning Range Rings for a Simulation Object 18-27
18.6.3. Configuring the Display of Range Rings 18-28
18.7. Displaying Entity Resources 18-30
18.7.1. Viewing Numerical Resource Data 18-30
18.8. Displaying Bounding Volumes 18-31
18.8.1. Showing Bounding Volumes Only for Selected Simulation Objects 18-32
18.9. Configuring Object Console Messages 18-32
18.9.1. Setting the Notification Level for Console Messages 18-33
18.9.2. Viewing All Console Messages 18-34
18.9.3. Answering Questions from Scripted Tasks 18-35
18.9.4. Saving Console Messages 18-35
18.9.5. Displaying an Object Console Warning Icon 18-36
18.9.6. Clearing the Object Console 18-36
18.10. Viewing Object Counts 18-37

Chapter 19. Creating Simulation Objects

19.1. Creating Simulation Objects 19-3
19.1.1. Default Placement of New Entities 19-3
19.1.2. Placing Entities Inside Buildings 19-4
19.1.3. Placing Simulation Objects on Other Simulation Objects .. 19-4
19.1.4. Simulation Object Resources 19-4
19.1.5. Deleting Simulation Objects 19-5
19.2. Editing Simulation Objects 19-5
19.3. Embarking and Disembarking Simulation Objects 19-7
19.3.1. Embarking and Disembarking Simulation Objects Instantly 19-8
19.3.2. Disembarking Simulation Objects Using the Disembark Command 19-11
19.4. Embedded Entities 19-12
19.4.1. Configuring an Entity to Support Embedded Entities 19-12

19.4.2. Deploying Embedded Entities 19-15
19.4.3. Deploying an Entity with a Task 19-15
19.4.4. Recovering Deployed Entities 19-16
19.4.5. Assigning Tasks and Plans to Embedded Entities 19-16
19.4.6. Restoring an Entity that has Embedded Entities 19-16
19.5. Collision, Obstacle, and Feature Avoidance 19-17
19.6. Entity Movement and Soil Type 19-18
19.7. Using a Joystick to Control Entities 19-19
19.7.1. Switching Between Systems 19-20
19.7.2. Disabling Joystick Control 19-21
19.8. Using the Keyboard to Control an Entity 19-21
19.9. Using Ground Clamping 19-21
19.9.1. Configuring Ground Clamping 19-23
19.10. Exaggerating the Scale of 3D Entity Models 19-24
19.11. Hiding 3D Models 19-27
19.12. Changing Force Colors 19-27

Chapter 20. Generating Traffic and Crowds

20.1. Generating Traffic 20-2
20.2. Creating Spawn Points 20-2
20.3. Creating Sink Points 20-4
20.4. Spawn Patterns 20-5
20.4.1. How Spawn Events are Scheduled 20-6
20.4.2. Spawning Entities in Bursts 20-8
20.4.3. Using a Custom Plan 20-9
20.4.4. Default Spawn Patterns and Scenario-Specific Patterns 20-9
20.5. Creating a Spawn Pattern 20-10
20.5.1. Copying a Spawn Pattern 20-14
20.6. Editing a Spawn Pattern 20-15
20.7. Deleting a Spawn Pattern 20-16
20.8. Creating Multiple Simulation Objects and Crowds 20-16
20.9. Creating Multiple Simulation Objects 20-17
20.9.1. Adding an Area for Multiple Simulation Object Creation 20-20
20.9.2. Specifying the Simulation Objects to Create 20-20
20.10. Creating Crowds 20-21
20.10.1. Crowd Tasks 20-22
20.11. Pedestrian-Oriented Tactical Graphics 20-22
20.11.1. Using Civilian Visit Points 20-22

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 22-2
22.1.1. Unit Combat 22-3
22.1.2. Unit Footprints 22-4
22.1.3. Unit Posture 22-4
22.1.4. Unit Movement 22-6
22.1.5. Limited Munition Attack 22-7
22.1.6. Contamination Areas 22-7
22.1.7. Unit Sensors 22-8
22.2. Combat Engineering Objects 22-9
22.2.1. Concealed Obstacles 22-11
22.2.2. Creating Combat Engineering Objects 22-12
22.2.3. How Engineering Units Create Engineering Objects 22-12
22.3. Logistics 22-14
22.3.1. Receiving Supplies 22-16
22.3.2. Providing Supplies 22-16
22.4. Electronic Warfare 22-16
22.4.1. Jamming Communications 22-17
22.4.2. Jamming Radar 22-18
22.4.3. Sensing Electronic Emissions 22-19

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. Writing Plans for Units 23-12
23.8. Deleting a Unit 23-13

Chapter 24. Displaying Units

24.1. Selecting Units 24-2
24.2. Expanding and Collapsing Units 24-2
24.2.1. Expanding and Collapsing Units in the Echelon View 24-4

24.2.2. Expanding and Collapsing the Echelon View by Level of Aggregation 24-4
24.3. Displaying Ghosted Icons 24-6
24.4. Displaying Unit Icons and Bounding Volumes 24-9
24.4.1. Specifying the Color Scheme for Unit Bounding Volumes 24-11

Section V Tasks, Sets, and Scripted Tasks

V.1. Introduction to Tasks 24-1
V.2. Introduction to Set Data Requests 24-2

Chapter 25. Assigning Tasks

25.1. Assigning Tasks to Simulation Objects 25-3
25.1.1. C++ Tasks and Scripted Tasks 25-3
25.1.2. Concurrent Task Execution 25-4
25.1.3. How do I Know which Simulation Object can Execute a Task? 25-6
25.1.4. Escaping the Task Assignment Process 25-6
25.1.5. Specifying Parameters for Tasks 25-6
25.1.6. Viewing Task Status 25-8
25.1.7. Filtering the Object Selection Lists 25-9
25.1.8. 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 Scripted Tasks 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
25.6. Fixed-Wing Entity Tasks and Behaviors 25-22
25.6.1. Placement of Newly Created Fixed-Wing Entities 25-22
25.6.2. Fly Task Behavior is Different from Move Task Behavior 25-23
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 25-27
25.6.6. How Fixed-Wing Entities Land 25-28
25.7. Rotary-Wing Entity Tasks and Behaviors 25-29

25.7.1. Controlling Rotary-Wing Orientation 25-31

Chapter 26. Tasks for Entity-Level Scenarios

26.1. Task Procedures 26-4
26.2. Animated Movement 26-5
26.3. Arm Mine at Depth 26-6
26.4. Attack Aircraft with Guns 26-6
26.5. Civilian Idle 26-7
26.6. Civilian Protest 26-7
26.7. Come to Stop 26-7
26.8. Convoy Along 26-8
26.9. Convoy To 26-9
26.10. Crowd Along Line 26-9
26.11. Crowd Around Location 26-10
26.12. Crowd Around Object 26-10
26.13. Crowd in Front of Entity 26-11
26.14. Deploy Sonobuoy 26-11
26.15. Deploy Sonobuoys Along Route 26-12
26.16. Disembark 26-13
26.17. Disembark All 26-13
26.18. Disperse Crowd 26-13
26.19. DI-Guy Animation 26-14
26.20. Drop Naval Depth Charge 26-14
26.21. Drop Naval Depth Charge at Location 26-15
26.22. Drop Naval Mine 26-16
26.23. Drop Naval Mines Along Route 26-17
26.24. Embark 26-18
26.25. Execute Close Air Support 26-21
26.26. Explode Charge at Depth 26-23
26.27. Fire at Target 26-23
26.28. Fire Cruise Missile 26-24
26.29. Fire for Effect Tasks 26-25
26.29.1. Firing Naval Guns 26-26
26.30. Fixed-Wing Land 26-26
26.31. Fixed Wing Takeoff 26-27
26.32. Flee From Entities 26-27
26.33. Fly Altitude 26-28
26.34. Fly Heading 26-29
26.35. Fly Heading and Altitude 26-30
26.36. Follow Entity 26-31
26.36.1. How Fixed-Wing Entities Follow Entities 26-31
26.37. Lase Target 26-32
26.38. Launch Anti-Submarine Missile (Vertical) 26-32
26.39. Launch Counter Measures 26-33
26.40. Launch Smoke 26-33
26.41. Launch Torpedo 26-34

26.41.1. Anti-ship (Fixed Depth) 26-34
26.41.2. Anti-Submarine 26-35
26.42. Lower Periscope 26-35
26.43. Move Along Route 26-36
26.44. Move Into Formation 26-37
26.45. Move to Altitude 26-38
26.46. Move to Depth 26-38
26.47. Move to Location 26-39
26.47.1. Adding Multiple Move to Location Tasks to a Plan 26-40
26.48. Move to Location (Plan Along Roads) 26-40
26.49. Move to Location (Plan Path) 26-41
26.50. Move to Waypoint 26-42
26.51. Move to Waypoint (Plan Along Roads) 26-43
26.52. Move to Waypoint (Plan Path) 26-44
26.53. Orbit 26-45
26.54. Patrol Route 26-45
26.55. Patrol Between 26-46
26.56. Pattern Hold (Location) 26-46
26.57. Pattern Hold (Waypoint) 26-47
26.58. Place IED 26-48
26.59. Protest Around Location 26-49
26.60. Protest Along Line 26-49
26.61. Suppressive Fire Tasks 26-50
26.61.1. Provide Suppressive Fire 26-51
26.61.2. Provide Suppressive Fire at Location 26-52
26.62. Raise Periscope 26-52
26.63. Release Bomb Tasks 26-53
26.64. Rotary-Wing Land 26-54
26.65. Sail Heading 26-54
26.66. Send Radio Set 26-55
26.67. Send Radio Task 26-56
26.68. Send Text Message 26-57
26.69. Sonar Dip 26-57
26.70. Strafe Ground Target 26-58
26.71. Sweep Naval Mines 26-60
26.72. Taxi and Launch from Carrier 26-60
26.73. Taxi on Carrier 26-61
26.74. Turn to Heading 26-61
26.75. User Task 26-62
26.76. Wait Tasks 26-62
26.76.1. Wait 26-62
26.76.2. Wait Duration 26-63
26.76.3. Wait Elapsed 26-63
26.77. Wander 26-63
26.78. Wander In Area 26-64
26.79. Reactive Tasks 26-64

Chapter 27. Tasks for Aggregate-Level Scenarios

27.1. Introduction to Aggregate-Level Tasks 27-4
27.2. Activate Jammer 27-4
27.3. Air-to-Ground Attack 27-4
27.4. Attack by Fire 27-5
27.5. Attack with Anti-Air Missile 27-5
27.6. Attack with Anti-Ship Missile 27-5
27.7. Attack with Land-Attack Missile 27-6
27.8. Attack with Torpedo 27-6
27.9. Automatic Air Defense 27-7
27.10. Biological Attack 27-7
27.11. Bomb Location 27-8
27.12. Breach Obstacles 27-9
27.13. Change MOPP Level 27-9
27.14. Change Posture 27-10
27.15. Chemical Attack 27-10
27.16. Construct Abatis 27-11
27.17. Construct Anti-Tank Ditch 27-12
27.18. Construct Barbed Wire 27-12
27.19. Construct Barricade 27-13
27.20. Construct Berm 27-13
27.21. Construct Booby Trap 27-14
27.22. Construct Bridge 27-15
27.23. Construct Dry Gap 27-16
27.24. Construct Fortified Area 27-17
27.25. Construct Fortified Line 27-18
27.26. Construct Minefield 27-18
27.27. Construct Strong Point 27-19
27.28. Deactivate Jammer 27-19
27.29. Destroy Bridge 27-20
27.30. Destroy Ditch 27-20
27.31. Destroy Explosive 27-20
27.32. Destroy Fortification 27-21
27.33. Destroy Obstacle 27-21
27.34. Disembark 27-21
27.35. Disembark All 27-21
27.36. Embark 27-21
27.37. FASCAM 27-22
27.38. Halt Movement 27-22
27.39. Improve Booby Trap 27-22
27.40. Improve Breach 27-23
27.41. Improve Bridge 27-23
27.42. Improve Ditch 27-23
27.43. Improve Fortification 27-24
27.44. Improve Obstacle 27-24

27.45. Indirect Fire 27-25
27.46. Move Along Route 27-26
27.47. Move Along Route Retrograde 27-26
27.48. Move to Location 27-26
27.49. Move to Location Retrograde 27-26
27.50. Move To Waypoint 27-26
27.51. Move to Waypoint Retrograde 27-26
27.52. Nuclear Attack 27-27
27.53. Patrol Route 27-27
27.54. Patrol Between 27-27
27.55. Road Construction Tasks 27-27
27.56. Send NBC Report 27-28
27.57. Send Obstacle Report 27-28
27.58. Send Radio Set 27-28
27.59. Send Radio Task 27-28
27.60. Send Text Message 27-28
27.61. User Task 27-29
27.62. Wait Tasks 27-29
27.63. Background Process List 27-29

Chapter 28. Creating Scripted Tasks

28.1. Introduction to Scripted Tasks 28-3
28.1.1. Task Type Scripted Tasks 28-4
28.1.2. The Reactive Task Type Scripted Task 28-5
28.1.3. Background Process Type Scripted Task 28-5
28.1.4. Script Metadata 28-5
28.1.5. The Lua Scripting Language 28-5
28.2. Creating a New Scripted Task 28-6
28.2.1. Specifying the Script ID 28-11
28.2.2. Specifying the Task Menu Icon 28-11
28.2.3. Specifying an Extended Name 28-12
28.2.4. Specifying Task Parameters 28-12
28.2.5. Specifying the Task Menu Location 28-16
28.2.6. Specifying Action Categories 28-17
28.2.7. Configuring a Script's Availability to a Simulation Object Type 28-18
28.2.8. Specifying the Programming Language for a Scripted Task 28-23
28.2.9. Creating Reactive Tasks 28-24
28.2.10. Creating Background Processes 28-25
28.3. Saving Scripted Tasks 28-27
28.4. Editing a Scripted Task 28-28
28.5. Filtering the List of Scripted Tasks 28-28
28.6. Organizing Scripted Tasks into Folders 28-29
28.6.1. Adding a Folder 28-29
28.6.2. Renaming a Folder 28-29

28.6.3. Deleting a Folder 28-29
28.6.4. Adding Scripted Tasks to a Folder 28-29
28.6.5. Removing a Scripted Task from a Folder 28-29
28.7. Exporting and Importing Scripted Tasks 28-30
28.7.1. Importing a Scripted Task Package 28-31
28.8. Copying a Scripted Task 28-32
28.9. Deleting a Scripted Task 28-32
28.10. Creating a System Scripted Task 28-33
28.10.1. Including System Scripted Tasks on the Task Menu 28-34
28.11. Specifying a Script Editor 28-36
28.12. Editing Lua Files 28-37

Chapter 29. Writing Scripts for Scripted Tasks

29.1. The VR-Forces Lua Interface 29-3
29.1.1. Lua Classes 29-3
29.1.2. Lua Modules 29-6
29.2. Script Loading and Execution 29-7
29.2.1. Script Entry Points 29-7
29.2.2. The Scripted Task Execution Sequence 29-9
29.2.3. Limitations for Checkpointing Scripted Tasks 29-12
29.2.4. Editing Scripted Tasks While a Scenario is Running 29-14
29.3. Tasks and Subtasks 29-15
29.3.1. Monitoring the Status of Tasks and Subtasks 29-16
29.3.2. Stopping Tasks 29-16
29.3.3. Task Parameters 29-16
29.4. Geometry 29-18
29.4.1. *Location3D* 29-18
29.4.2. *Vector3D* 29-18
29.4.3. *VectorOffset3D* 29-20
29.4.4. *VectorGeoc3D* 29-20
29.5. Coding Messages for Translation 29-21
29.6. Error Detection 29-21
29.7. Interactive Scripts 29-22
29.8. A Basic Scripted Task 29-23
29.8.1. Create and Move to Waypoint Metadata 29-24
29.8.2. The Create and Move to Waypoint Lua Script 29-27
29.9. A Simple Reactive Task 29-28
29.10. Background Processes 29-29

Chapter 30. Setting Simulation Object State

30.1. Setting Simulation Object State and Attributes 30-4
30.2. Activation 30-5
30.3. Active Sonar Mode 30-6
30.4. Activity 30-6
30.5. Aiming 30-7
30.6. Altitude 30-7

30.7. Ammunition 30-8
30.8. Ammunition Pacing/Tracking 30-8
30.9. Appearance 30-9
30.10. Armed 30-9
30.11. Capabilities 30-10
30.12. Collision Avoidance Types 30-11
30.13. Concealed 30-12
30.14. Contamination 30-12
30.15. Counter Measures Auto Launch 30-12
30.16. Current Speed 30-13
30.17. Destroyed 30-13
30.18. Detonation Fuse Type 30-14
30.19. DI-Guy Appearance 30-15
30.20. DI-Guy Hand Item 30-15
30.21. Disembarked 30-16
30.22. Embarked 30-16
30.23. Emitter 30-17
30.24. Engagement Results 30-18
30.25. Equipment Pacing/Tracking 30-19
30.26. Food, Water, Fuel, Oil and Lubricant 30-19
30.27. Food, Water Fuel, Oil and Lubricant Pacing/Tracking 30-20
30.28. Force 30-21
30.29. Formation 30-22
30.30. Heading 30-23
30.31. Health 30-23
30.32. IFF 30-24
30.33. Jam Targets 30-25
30.34. Lase Autonomous 30-25
30.35. Laser Code 30-26
30.36. Location 30-26
30.37. MOPP Level 30-27
30.38. Morale 30-27
30.39. Navigation Preferences 30-28
30.40. Notify Level 30-28
30.41. Ordered Speed 30-29
30.42. Percent Complete 30-29
30.43. Posture (Unit) 30-30
30.44. Posture (Human) 30-31
30.45. Preferred Targets 30-32
30.46. Radar Mode 30-33
30.47. Reorganize 30-33
30.48. Resources 30-34
30.49. Resources Pacing/Tracking 30-34
30.50. Restore 30-35
30.51. Resupply Mode 30-35
30.52. Rules of Engagement 30-36

30.52.1. How Fire-When-Fired-Upon Works 30-36
30.53. Sector of Responsibility 30-37
30.54. Sensor Aim 30-38
30.55. Sensor Zoom 30-38
30.56. Sonar Depth 30-39
30.57. Spot Reports 30-40
30.58. Superior 30-41
30.59. Supplying 30-41
30.60. Surrendered 30-41
30.61. Synchronize Laser Code 30-42
30.62. Target 30-42
30.63. Tasked by Superior 30-43
30.64. Tracer Use 30-43
30.65. Unit State 30-43
30.66. Weapons Pacing/Tracking 30-44

Section VI Plans

VI.1. Introduction to Plans 30-1

Chapter 31. Plan Concepts

31.1. Introduction to Individual Plans and Global Plans 31-2
31.2. Conditional Statements 31-4
31.2.1. The If Statement 31-6
31.2.2. When Statements (Triggers) 31-7
31.2.3. While Statements 31-9
31.2.4. Conditional Tests 31-10
31.3. Viewing Plans 31-14
31.4. Considerations for Creating Plans 31-14
31.4.1. Considerations for Using Triggers 31-15
31.4.2. Moving In Formation 31-16
31.4.3. Using the Tasked-By-Superior Request in a Plan 31-16
31.4.4. Following Simulation Objects 31-16
31.4.5. Planning Tasks for Aircraft 31-17
31.4.6. Using Non-VR-Forces Simulation Objects in Plans 31-17

Chapter 32. Writing Plans

32.1. Writing Plans for Simulation Objects 32-2
32.1.1. Adding a Task or Set Data Request to a Plan 32-3
32.1.2. Adding a Conditional Statement to a Plan 32-4
32.1.3. Editing a Statement 32-10
32.1.4. Deleting Statements from a Plan 32-10
32.1.5. Printing Plans 32-10
32.1.6. Saving Changes to a Plan 32-11
32.2. Creating Global Plans 32-11
32.2.1. Controlling When Global Plans Run (Quick Launch) 32-13
32.2.2. Opening an Information Dialog Box for a Global Plan 32-13

32.3. Adding Global Commands to a Plan 32-14
32.3.1. Adding a Global Task or Set to a Plan 32-15
32.3.2. Sending Console Messages 32-16
32.3.3. Creating Objects from Within a Plan 32-17
32.3.4. Deleting Objects from Within a Plan 32-17
32.3.5. Issuing a Plan 32-18
32.3.6. Sending View Control Messages 32-19
32.4. Writing Plans for Units 32-22
32.5. Writing a Plan for Multiple Simulation Objects 32-22
32.6. Writing Plans for Remote Entities 32-22
32.7. Copying Plans and Plan Statements 32-23
32.8. Restarting a Plan 32-23
32.9. Abandoning a Plan 32-24

Section VII Tactical Graphics

Chapter 33. Introduction to Tactical Graphics

33.1. Introduction 33-2
33.2. Points 33-4
33.3. Phase Lines 33-4
33.4. Routes 33-5
33.5. Areas 33-5
33.6. Obstacles 33-5
33.7. Displaying Tactical Graphics 33-5
33.7.1. Showing and Hiding Tactical Graphics 33-7
33.7.2. Showing and Hiding Tactical Graphics Labels 33-8
33.7.3. Displaying Height-Above-Terrain Lines for Vertices 33-9

Chapter 34. Overlays

34.1. Creating and Editing Overlays 34-2
34.1.1. Creating an Overlay 34-3
34.1.2. Selecting Overlays 34-3
34.1.3. Locking an Overlay 34-4
34.1.4. Hiding the Objects on an Overlay 34-4
34.1.5. Changing an Overlay's Name 34-5
34.1.6. Deleting an Overlay 34-5
34.2. Saving and Loading Tactical Graphics and Overlays 34-6
34.2.1. Loading Tactical Graphics and Overlays 34-6

Chapter 35. Creating and Editing Tactical Graphics

35.1. Creating Tactical Graphics 35-2
35.1.1. Creating Freehand Lines 35-2
35.1.2. Drawing Circles and Ellipses 35-3
35.1.3. Drawing Boxes 35-3
35.1.4. Creating Volumetric Tactical Graphics 35-4

35.1.5. Adding Text to an Overlay 35-4
35.1.6. Assigning Tactical Graphics to an Overlay 35-5
35.1.7. Displaying a Terrain Profile When You Create a Line 35-6
35.1.8. Publishing Tactical Graphics 35-7
35.1.9. Creating a Computed Route 35-7
35.1.10. Creating an Intel Collection Area 35-9
35.1.11. Creating Routes for Aircraft 35-10
35.2. Attaching Tactical Graphics to a Simulation Object 35-11
35.2.1. Attaching a Tactical Graphic Automatically 35-11
35.3. Deleting Tactical Graphics 35-11
35.4. Editing Tactical Graphics 35-12
35.4.1. Editing Vertices 35-13
35.4.2. Resizing Boxes and Text Objects 35-15
35.4.3. Rotating an Ellipse 35-16
35.4.4. Changing the Direction of a Line 35-16
35.4.5. Changing the Color of a Tactical Graphic 35-16
35.4.6. Changing Linear and Areal Style Properties 35-17
35.5. Active and Inactive Tactical Graphics 35-18

Chapter 36. Remote Graphics

36.1. Displaying Remote Graphics 36-2
36.1.1. Viewing a List of Remote Graphics 36-4

Section VIII Visual and Audio Effects

Chapter 37. Displaying Tactical Information and Effects

37.1. Displaying Trailing and Decal Effects (3D Only) 37-3
37.1.1. Specifying Models for Trailing Effects and Decal Effects 37-4
37.2. Displaying Track Histories 37-5
37.2.1. Configuring Track Histories 37-6
37.2.2. Displaying a Terrain Profile for a Track History 37-8
37.3. Viewing Fire and Detonation Effects 37-8
37.4. Viewing Fire and Detonation Lines 37-9
37.5. Viewing Unit Combat Graphics 37-10
37.6. Visualizing Tasks 37-12
37.6.1. Pinning Task Visualizations for a Simulation Object 37-13
37.7. Displaying Sensor Contact Lines 37-13
37.7.1. Enabling the Display of Sensor Contact Lines 37-14
37.7.2. Specifying which Simulation Objects Display Sensor Contact Lines 37-14
37.7.3. Configuring Sensor Contact Lines 37-15
37.8. Displaying Laser Designators 37-16
37.9. Displaying Supply Lines 37-17
37.10. Displaying Tactical Smoke 37-17
37.11. Displaying Electromagnetic Emissions 37-18
37.12. Displaying Radio Communication Lines 37-19

37.12.1. Enabling Radio Communications Graphics 37-20
37.12.2. Configuring the Lifetime and Display Mode 37-21
37.12.3. Configuring the Color of Radio Communication Lines .. 37-22

Chapter 38. Lighting Effects

38.1. Displaying Lighting Effects 38-2
38.2. Enabling and Disabling Lighting 38-3
38.2.1. Configuring Ambient and Diffuse Lighting 38-4
38.2.2. Specifying the Emissive Lighting Scale 38-6
38.3. Displaying Dynamic Lighting 38-6
38.4. Displaying Ocean Planar Reflections 38-7
38.4.1. Configuring Ocean Planar Reflections 38-8
38.5. Lens Flare 38-9
38.6. Displaying Crepuscular (Sun) Rays 38-11
38.7. Configuring the Skybox Cube Map 38-12
38.8. Flashlight Mode 38-13
38.9. Disabling Lighting and Clouds 38-13
38.10. Displaying Shadows (3D Only) 38-14
38.10.1. Configuring Shadows 38-16

Chapter 39. Marine Effects

39.1. Displaying Water Effects 39-2
39.1.1. Enabling Sea Spray 39-4
39.1.2. Enabling High Quality Rotor Wash 39-4
39.2. Displaying Wakes and Spray Effects 39-5
39.3. Enabling Buoyancy for Surface Entities 39-6

Chapter 40. Using Sensors

40.1. Using Sensors 40-2
40.2. Enabling Sensors 40-3
40.3. Configuring Sensors 40-4
40.4. Adding New Sensors 40-5

Chapter 41. Intervisibility

41.1. Intervisibility (Line-of-Sight) 41-2
41.2. Types of Intervisibility 41-3
41.2.1. Point-to-Point Intervisibility 41-3
41.2.2. Intervisibility Fan 41-4
41.2.3. Entity Intervisibility 41-4
41.3. Intervisibility Objects can be Transient or Permanent 41-5
41.3.1. Creating a Transient Intervisibility Object 41-6
41.3.2. Creating a Permanent Intervisibility Object 41-7
41.4. Editing an Intervisibility Object 41-8
41.5. Configuring Intervisibility Lines and Fans 41-9
41.6. Pinning an Intervisibility Object to a Simulation Object 41-10
41.7. Showing and Hiding Intervisibility Lines 41-10

41.8. Displaying Intervisibility Line Information 41-11
41.9. Displaying an Intervisibility Line's Terrain Profile 41-12
41.10. Deleting Intervisibility Objects 41-13

Chapter 42. Sound Effects

42.1. Introduction to Sound Effects 42-2
42.2. Enabling Sound Effects 42-2
42.2.1. Setting the Sound Volume 42-3
42.2.2. Specifying the Range in which Sounds are Heard 42-3
42.3. Associating Sound Files with Entities and Events 42-4
42.3.1. Mapping a Sound File to an Entity 42-5
42.3.2. Mapping Sound Files to Fire and Detonation Events 42-8

Section IX The Observer and Viewing Scenarios

Chapter 43. The Observer and Observer Modes

43.1. Switching Between the 2D and 3D Projections 43-2
43.2. Observer Modes 43-2
43.3. Changing the Observer Mode 43-3
43.4. Creating and Editing Observer Modes 43-3
43.4.1. Creating an Observer Mode 43-4
43.4.2. Deleting an Observer Mode 43-5
43.4.3. Editing an Observer Mode 43-5
43.4.4. Renaming an Observer Mode 43-5
43.5. Adding an Observer 43-6
43.5.1. Adding an Observer on the Observer Settings Page 43-6
43.5.2. Adding an Observer in the Display Engine Configuration Editor 43-7
43.6. Removing an Observer 43-9
43.7. Introduction to XR Mode 43-10
43.8. Model Sets 43-11
43.8.1. Changing the Model Set 43-11
43.9. Communicating Information About the Observer 43-11
43.9.1. Making the Observer Visible to Other Applications 43-11
43.9.2. Viewing and Attaching to Other Observers 43-12
43.10. Displaying Models for Remote Observers 43-12

Chapter 44. Moving the Observer

44.1. Moving the Observer 44-3
44.1.1. Using the Compass to View the Observer's Heading 44-4
44.1.2. Editing the Compass's Attributes 44-5
44.2. Observer Movement Functions and Keyboard Mappings 44-6
44.2.1. 3D Observer Movement Coordinate Systems 44-7
44.2.2. 2D Observer Coordinate System 44-7

44.2.3. The Attached Context and Observer Movement (3D Only) 44-8
44.3. Moving the Observer from the Keyboard in the 3D View 44-10
44.3.1. Navigation Options 44-12
44.3.2. Orbiting 44-13
44.3.3. Moving the Camera 44-13
44.4. Moving the Observer Using the Mouse (3D View) 44-14
44.4.1. Dragging the Terrain 44-14
44.4.2. Changing the Observer's Orientation with the Mouse 44-15
44.4.3. Dragging the View 44-15
44.4.4. Orbiting the Terrain with the Mouse 44-15
44.4.5. Teleporting the Observer 44-16
44.5. Moving the Observer in the 2D View 44-16
44.5.1. Moving the Observer from the Keyboard in the 2D View 44-17
44.5.2. Moving the Observer Using the Mouse in the 2D Projection 44-18
44.6. Zooming the Observer 44-18
44.6.1. Zooming the Observer in the 3D Projection 44-19
44.6.2. Zooming the Observer in the 2D Projection 44-20
44.6.3. Setting the Scale Factor for Paged LODs 44-20
44.6.4. Zooming to Extents 44-20
44.7. Using the Depth of Field Effect 44-22
44.7.1. Configuring Depth of Field 44-24
44.8. Centering the Observer on a Simulation Object 44-25
44.9. Simulating Camera Jitter 44-25
44.10. Forcing Orientation North 44-26
44.11. Enabling and Disabling View Constraints (3D Only) 44-27
44.12. Controlling Navigation Speed 44-27
44.12.1. Enabling or Disabling Speed Scaling 44-27
44.12.2. Changing the Observer's Movement Speed 44-28
44.13. Saving and Recalling Views 44-29
44.13.1. Recalling a View 44-30
44.13.2. Deleting Views 44-31
44.13.3. Specifying the Startup View 44-31
44.13.4. Saving Views to a File 44-31
44.13.5. Loading Views 44-32
44.14. Controlling the Observer from Other Applications 44-32
44.14.1. Enabling the Processing of View Control Messages 44-32

Chapter 45. Attaching the Observer to Objects

45.1. Attaching the Observer to a Simulation Object or Prop 45-2
45.1.1. Attaching to an Articulated Part 45-4
45.1.2. Attaching the Observer in Mimic Track or Tether Track Mode 45-4

45.1.3. Attaching the Observer in Mimic Location Track or Tether Location Track Mode 45-4
45.1.4. Filtering the Object Attachment List 45-5
45.1.5. Detaching from a Simulation Object 45-5
45.1.6. Hiding the Attached Model (3D Projection Only) 45-5
45.1.7. Attaching to Props 45-5
45.1.8. Improving Performance when Attaching to Ground Objects 45-6
45.2. Attach Modes in the 3D View 45-6
45.2.1. Absolute Mode 45-7
45.2.2. Follow Mode 45-7
45.2.3. Mimic Mode 45-8
45.2.4. Mimic Track Mode 45-9
45.2.5. Mimic Location Track Mode 45-9
45.2.6. Space Follow Mode 45-10
45.2.7. Tether Mode 45-11
45.2.8. Tether Track Mode 45-12
45.2.9. Tether Location Track Mode 45-12
45.2.10. Track Mode 45-13
45.3. Attach Modes in the 2D View 45-14
45.4. Attaching the Observer at Startup 45-14

Chapter 46. Inset Views, Sensor FOVs, and HUDs

46.1. Inset Views 46-2
46.2. Displaying Cockpit Displays (3D Only) 46-4
46.3. Displaying Instrument Panels 46-6
46.3.1. Configuring Entities to Use Instrument Panels 46-7
46.4. Displaying Sensor Views (3D Only) 46-8
46.4.1. Displaying a Sensor Field of View Cone 46-9
46.4.2. Controlling the Sensor View 46-10

Section X Terrain

Chapter 47. Introduction to Terrain Usage in VR-Forces

47.1. Composing Terrains (and Creating MTF Files) 47-2
47.2. Supported File Formats 47-2
47.2.1. Terrain Data 47-3
47.2.2. Elevation Data 47-3
47.2.3. Supported Image Formats 47-4
47.2.4. Supported Feature Data Formats in the Front-end 47-4
47.3. Feature Layers and Props 47-4
47.3.1. Geometric Representation of Point Feature Data 47-5
47.3.2. Feature Data Types Supported by the Back-end 47-5
47.4. Coordinate Systems 47-6
47.4.1. The UTM Coordinate System 47-6
47.4.2. Geocentric Terrain Databases 47-6

47.5. Loading Supported Databases 47-7
47.6. Using Large Terrain Databases 47-7
47.6.1. Terrain Paging 47-8
47.6.2. Terrain Streaming 47-9

Chapter 48. Displaying Terrain Information

48.1. Introduction 48-2
48.2. Configuring Visible Surfaces (GDB Soil Types) 48-2
48.3. Coloring Terrain Based on Elevation 48-4
48.3.1. Configuring Elevation Color Settings 48-5
48.4. Displaying Grid Lines 48-7
48.4.1. Configuring Grid Lines 48-8
48.5. Displaying Contour Lines 48-9
48.5.1. Configuring Contour Lines 48-10
48.6. Displaying Models and Terrain in Wireframe Mode 48-11
48.7. Filtering Earth File Layers 48-13
48.8. Displaying Feature Data (2D) 48-14
48.8.1. Displaying a List of Vector Features 48-15
48.8.2. Changing the Colors Used for Feature Data 48-16
48.9. Displaying Coordinates and Soil Type for a Location 48-17
48.10. Viewing the Map Scale 48-18
48.11. Terrain Databases Provided with VR-Forces 48-18

Chapter 49. Using Terrain Profiles

49.1. Terrain Profiles 49-2
49.1.1. Displaying a Line's Terrain Profile 49-3
49.1.2. Configuring Terrain Profiles 49-3
49.2. Analyzing Terrain Using a Terrain Profile Line 49-4

Chapter 50. Composing Terrains

50.1. Creating a Composed Terrain 50-3
50.1.1. Considerations and Limitations for Building Terrains 50-4
50.1.2. Saving a Terrain 50-5
50.2. Adding Elevation Data (Terrain Patches) to a Terrain 50-5
50.3. Adding Images to a Terrain 50-7
50.3.1. Changing the Display Order of Raster Maps 50-10
50.4. Adding a Feature Layer 50-11
50.5. Adding a Dynamic Ocean Layer 50-13
50.5.1. Extracting Water Textures and Adding an Ocean Layer ... 50-15
50.5.2. Adding a Dynamic Ocean Layer Directly 50-16
50.5.3. Setting the Ocean LOD Elevation 50-16
50.5.4. Configuring the Ocean Height Map 50-17
50.6. Specifying Simulation Ocean Settings 50-19
50.7. Using Shader-based Effect Maps 50-20
50.7.1. Debugging Shaders 50-22
50.7.2. Reloading Shaders 50-23

50.8. Adding Props to a Terrain 50-24
50.8.1. Extracting Props from a Terrain Patch 50-24
50.8.2. Adding Props from a Feature Layer 50-26
50.8.3. Viewing a List of Props 50-28
50.8.4. Selecting Props 50-28
50.8.5. Setting the Opacity of Props 50-29
50.8.6. Changing a Prop's Type 50-30
50.8.7. Changing a Prop's Position or Orientation 50-30
50.8.8. Changing a Prop's Model Definition 50-30
50.9. Configuring File Caching 50-31
50.9.1. Caching Terrain Server Data 50-32
50.9.2. Caching osgEarth Terrain Data Offline 50-32
50.9.3. Enabling Texture Compression 50-33
50.9.4. Clearing the File Cache 50-33
50.10. Displaying DDS Textures Correctly 50-34
50.10.1. Flipping DDS Textures Globally 50-35
50.11. Converting OpenFlight File Texture Data 50-36

Chapter 51. Paged and Streaming Terrains

51.1. Connecting to Terrain Servers 51-2
51.1.1. Adding Terrain Server Connections 51-4
51.1.2. Connecting to Terrain Servers through a Proxy 51-5
51.2. Loading MetaFlight Terrains 51-5
51.3. Preprocessing Paged Terrains 51-5
51.4. Manually Paging-In Terrain 51-6
51.5. Displaying Information About Paged and Streaming Terrains 51-7
51.5.1. Displaying Information about Feature Paging 51-9
51.5.2. Displaying Additional Terrain Paging Information 51-10
51.5.3. Configuring Terrain Paging for the Back-End 51-12
51.6. Building Efficient MetaFlight Terrains for VR-Forces 51-12

Chapter 52. Editing Earth Files

52.1. Earth Files 52-3
52.2. A Simple Earth File 52-3
52.2.1. Including External Files into Earth Files 52-5
52.3. Loading Multiple DTED Files 52-6
52.4. Processing Streamed Data 52-6
52.5. Adding Feature Layers to an Earth File 52-7
52.5.1. Configuring Point Features 52-9
52.5.2. Configuring Linear Features 52-10
52.5.3. Configuring Areal Features 52-11
52.6. Extruded Buildings 52-12
52.7. Using Cut-in Sites for High Resolution Insets 52-13
52.7.1. Using the Boundary Generation Tool 52-14
52.8. Loading CDB Databases 52-16
52.8.1. Preprocessing CDB Databases 52-16

52.9. Earth File Options 52-17
52.10. Differential Processing of Elements and Attributes 52-17
52.10.1. Differential Processing of model Elements 52-18
52.11. Streaming Buoys and Beacons 52-19
52.11.1. Specifying Model Definitions for Streamed Buoys 52-20
52.11.2. Modeling Buoys Using Texture Atlases 52-23
52.11.3. Configuring Streaming Beacons 52-24
52.11.4. Configuring Streamed Navigation Lights 52-25
52.11.5. Configuring Seasonal Buoys and Beacons 52-26
52.11.6. Streaming Daymarks 52-27
52.11.7. Streaming Topmarks 52-27
52.11.8. Generating Signal Sequences 52-28

Chapter 53. Generating Navigation Data

53.1. Introduction 53-2
53.2. Creating Navigation Areas 53-3
53.2.1. Editing a Navigation Area 53-4
53.2.2. Reverting Edits to Navigation Area 53-5
53.2.3. Displaying Navigation Areas 53-5
53.2.4. Removing a Navigation Area 53-5
53.3. Generating Navigation Data 53-5
53.3.1. Navigation Data Profiles 53-6
53.3.2. Generating Navigation Data for a Navigation Area 53-6
53.3.3. Using Newly Generated Navigation Data 53-8
53.4. Generating Navigation Data for Entities 53-8
53.5. Importing Navigation Areas and Navigation Data 53-10
53.6. Displaying Navigation Data 53-11
53.6.1. Opening Navigation Lab from an Information Dialog Box 53-13
53.6.2. Showing Simulation Object Movement in Navigation Lab 53-14

Chapter 54. Processing MetaFlight Files

54.1. Processing MetaFlight Files for Use in VR-Forces 54-2
54.2. Splitting Datasets into Individual MetaFlight Files 54-2
54.3. Converting Virtual Texture Datasets to Tiled Textures 54-3
54.3.1. The mftTool 54-3
54.3.2. Using the mftTool to Process MetaFlight Files 54-5
54.3.3. Convert Geometry Grid Datasets with Tiled Textures 54-6
54.4. Convert Source Data into MEDF Format 54-6
54.5. Create an MTF File for the MetaFlight Terrain 54-7

Chapter 55. Configuring Feature Data

55.1. Introduction 55-2
55.2. Specifying Different Feature Names for the Front-End and Back-end 55-2

55.3. Named Queries 55-3
55.4. Feature Configuration Attributes 55-4
55.5. Attribute Transformations 55-4
55.5.1. Transformations as Aliases 55-4
55.5.2. Transformations as Derived Attributes 55-5
55.5.3. Transformations as New Attributes 55-5
55.5.4. Writing Multiple Transforms to Account for Possible Failures 55-5
55.6. Source-Specific Configuration 55-5
55.6.1. Loading Source-Specific Configuration Files 55-6
55.6.2. Redefining Queries 55-6
55.7. Extracting Feature Data from GDB to Shapefiles 55-7
55.8. Converting S-57 Feature Data to Shapefiles 55-7
55.9. Configuring Aggregate-Level Movement Restrictions 55-9
55.10. Dynamic Features 55-10

Chapter 56. Terrain Tutorials

56.1. Composing a Terrain through the GUI 56-2
56.1.1. Add Elevation Data (Add a Terrain Patch) 56-3
56.1.2. Add Imagery 56-5
56.1.3. Add a Feature Layer 56-9
56.1.4. Add Props 56-10
56.2. Creating a Terrain with an Earth File 56-14
56.2.1. Add a Map Element 56-15
56.2.2. Add Elevation Data to the Earth File 56-15
56.2.3. Add Imagery to the Earth File 56-15
56.2.4. Add Feature Data to the Earth File 56-16
56.2.5. Add Options to the Earth File 56-18
56.2.6. Save the Earth File 56-19
56.2.7. Load the Earth File and Save is as an MTF File 56-19
56.3. Extracting Props from a Terrain 56-20
56.3.1. Create a New Terrain and Add a Terrain Patch 56-21
56.3.2. Extract the Props 56-21
56.3.3. Change the Model Definition for a Prop 56-24

Section XI Configuring Visual Models

XI.1. Introduction to VR-Forces Object Modeling 56-2

Chapter 57. Model and Element Definitions

57.1. Managing Model and Element Definitions 57-2
57.2. Creating and Editing Schemas 57-3
57.2.1. Creating Schemas 57-4
57.2.2. Deleting Schemas 57-5
57.2.3. Copying Schemas 57-5
57.2.4. Adding a Parameter to a Schema 57-6
57.2.5. Deleting a Parameter from a Schema 57-7

57.2.6. Editing a Schema Parameter 57-7
57.3. Creating and Editing Model Definitions 57-8
57.3.1. Creating a Model Definition 57-9
57.3.2. Deleting a Model Definition 57-10
57.3.3. Copying a Model Definition 57-11
57.3.4. Filtering the List of Model Definitions 57-11
57.3.5. Editing a Model Definition 57-12
57.3.6. Saving Model Definitions 57-14
57.4. Creating and Editing Element Definitions 57-14
57.4.1. Creating an Element Definition 57-15
57.4.2. Editing an Element Definition 57-16
57.4.3. Deleting an Element Definition 57-20
57.4.4. Saving Element Definitions 57-20

Chapter 58. Mapping Models and Effects

58.1. Introduction to Object Type Mapping 58-2
58.1.1. Adding an Object Type Mapping 58-3
58.1.2. Editing an Object Type Mapping 58-4
58.1.3. Filtering the Element Definition List 58-5
58.1.4. Deleting an Object Type Mapping 58-6
58.1.5. How VR-Forces Maps DI-Guy Models 58-6
58.2. Clearing the Model Instancing Cache 58-7

Chapter 59. Configuring Specific Visual Models

59.1. Configuring 2D Icons 59-2
59.1.1. Editing Font-Based 2D Icons 59-2
59.1.2. Using Images for 2D Icons 59-4
59.1.3. Adding Images for Simulation Object Icons 59-9
59.2. Adding New Cockpit Display Models 59-10
59.2.1. Installing a Cockpit DLL 59-11
59.2.2. Creating a Model Definition for a Cockpit 59-11
59.3. Adding Object Models as Props 59-13
59.3.1. Adding a Category to the Props Palette 59-13
59.4. Configuring Wakes 59-14
59.4.1. Configuring Tidal Stream Wakes 59-16
59.5. Adding Wind-based Controls to Models 59-16
59.6. Flipping DDS Textures for a Model 59-17
59.7. Configuring SpeedTree Trees 59-18
59.7.1. Randomizing the Size of SpeedTrees 59-20
59.8. Best Practices for Creating Models for VR-Forces 59-22
59.9. Compressing Model Files 59-23

Chapter 60. Configuring Emitter Volumes

60.1. Configuring Emitter Volumes 60-2
60.1.1. Configuring Emitter Volume Color 60-2
60.1.2. Controlling Emitter Volume Radius 60-3

60.1.3. Configuring Emitter Volume Segments 60-4

Chapter 61. The WRM Specification (DIS Notes)

61.1. Introduction 61-2
61.2. The OpenFlight File Format 61-2
61.2.1. Node Name and Comment Fields 61-3
61.2.2. External References and Instancing 61-3
61.3. Modeling for DIS Interoperability 61-3
61.3.1. The DIS Attribute Lexicon (DAL) 61-3
61.3.2. DAL Keywords 61-4
61.3.3. Model Coordinate Systems 61-5
61.4. Articulated Parts 61-7
61.5. Entity Appearances 61-8
61.6. Animation 61-11
61.7. DIS Keyword Values 61-12

Chapter 62. Model Tutorials

62.1. Adding a Model to VR-Forces 62-2
62.1.1. Create a Model Definition 62-2
62.1.2. Add an Element Definition 62-5
62.1.3. Add a Model Mapping 62-8
62.1.4. Test the Model Mapping 62-10

Section XII Creating and Editing Simulation Models

Chapter 63. The Simulation Object Editor and SMSs

63.1. Introduction to the Simulation Object Editor 63-2
63.2. Starting the Simulation Object Editor 63-3
63.3. Simulation Model Sets 63-5
63.3.1. Including Simulation Model Sets in other Simulation Model Sets 63-6
63.3.2. SMS Priority for Included and Multiple SMSs 63-8
63.3.3. Lua Scripts in Included SMSs 63-9
63.3.4. Application Level Files in Included SMSs 63-9
63.3.5. Search Paths for Referenced Files in an SMS 63-10
63.3.6. Promoting a Model to the Loaded SMS 63-11
63.4. Migrating an SMS to a New Release 63-12
63.4.1. Upgrading Old Simulation Model Sets 63-13
63.5. Opening a Simulation Model Set 63-15
63.6. Creating a New Simulation Model Set 63-16
63.6.1. Creating a Completely New Simulation Model Set 63-17
63.6.2. Specifying the Default Simulation Model Set for Scenarios 63-17
63.7. Applying Platform Updates 63-17

Chapter 64. Editing Simulation Object Models

64.1. Editing Generic Object Parameters 64-3
64.1.1. Selecting an Object to Edit 64-3
64.1.2. Editing an Object's Categories 64-3
64.1.3. Changing an Object's Object Type Enumeration 64-4
64.1.4. Changing an Object's Name 64-6
64.1.5. Specifying Whether an Object Can Be Created 64-6
64.1.6. Specifying the Object Creation Palette 64-7
64.1.7. Specifying the Default Overlay 64-7
64.1.8. Undoing (Reverting) Changes to an Object 64-8
64.2. Editing Parameters that are Specific to Simulation Objects 64-8
64.2.1. Editing a Simulation Object's Used By Countries List 64-8
64.2.2. Editing a Simulation Object's Bounding Volume (Size) 64-9
64.2.3. Displaying a Simulation Object's Bounding Volume 64-10
64.2.4. Specifying Support Points 64-11
64.2.5. Excluding Entities from Batch Bounding Volume and Support Point Updates 64-11
64.2.6. Editing Behavioral Parameters 64-11
64.2.7. Adding Object Geometry to an Entity 64-13
64.2.8. Editing a Lifeform's DI-Guy Character, Appearance, and Animation 64-14
64.3. Editing a Simulation Object's Visual Model 64-16
64.3.1. Changing the 3D Model or Colorized Model 64-17
64.3.2. Changing the 2D Icon Using a Font 64-18
64.3.3. Changing the 2D Icon Using an Image 64-19
64.3.4. Adding a New Visual Model or Image 64-20
64.3.5. Displaying Details about the Visual Model 64-21
64.4. Editing a Simulation Object's Systems 64-22
64.4.1. Adding a System 64-23
64.4.2. Selecting a Damage Model 64-24
64.4.3. Selecting a Movement Model 64-24
64.4.4. Editing a System's Properties 64-24
64.4.5. Editing a System in the OPD Editor 64-25
64.4.6. Removing a System 64-25
64.5. Configuring Scripted Entity Movement 64-26
64.5.1. Configuring Ballistic Missile Movement 64-27
64.5.2. Configuring Animated Movement 64-28
64.5.3. Editing Movement File Configurations 64-30
64.6. Creating a New Entity 64-31
64.6.1. Creating a New Entity from an Existing Entity 64-32
64.7. Configuring Embarkation 64-33
64.7.1. Adding Ingress and Egress Points 64-35
64.7.2. Configuring Slots 64-38
64.8. Creating a Simulation Object Group 64-41

64.8.1. Setting the Simulation Connection when Creating an SOG 64-42
64.9. Deleting an Object Model 64-42
64.10. Navigating Among Edited Simulation Objects 64-42
64.11. Editing Tactical Graphics 64-43
64.11.1. Editing a Tactical Graphic's Menu Icon 64-43
64.11.2. Setting the Default Values for Tactical Graphics Properties 64-44
64.11.3. Specifying the Properties that can be Edited in the Front-End 64-45
64.11.4. Editing the Visual Model for a Tactical Graphic 64-46
64.12. Editing the Category List 64-48
64.12.1. Adding a New Category 64-48
64.12.2. Removing a Category 64-49
64.12.3. Renaming a Category 64-49
64.12.4. Undoing Category Changes 64-49
64.13. Managing the Forces List 64-50
64.13.1. Adding a Force 64-50
64.13.2. Editing a Force 64-51
64.13.3. Removing a Force 64-52
64.14. Importing Entity Files 64-52

Chapter 65. Editing Units for Entity-Level Scenarios

65.1. Introduction to Editing Units 65-2
65.2. Creating a Unit for Entity-Level Scenarios 65-2
65.3. Specifying an Aggregate As Option 65-3
65.4. Editing Unit Composition 65-4
65.4.1. Adding Subordinates 65-5
65.4.2. Removing Subordinates 65-5
65.4.3. Replacing Subordinates 65-6
65.4.4. Changing the Order of Subordinates 65-6
65.4.5. Editing a Subordinate Unit 65-7
65.5. Editing Formations 65-8
65.5.1. Adding Formations 65-9
65.5.2. Removing Formations 65-10
65.5.3. Renaming Formations 65-10
65.5.4. Specifying the Default Formation 65-10
65.5.5. Expanding and Collapsing the Display of Formations 65-10
65.5.6. Displaying Bounding Boxes 65-11
65.5.7. Copying Formations 65-12
65.5.8. Laying Out a Formation Automatically 65-13
65.5.9. Changing a Formation's Layout by Hand 65-13

Chapter 66. Editing Aggregate-Level Objects

66.1. Creating Objects for Aggregate-Level Modeling 66-3

66.2. Configuring Simulation Objects for Aggregate-Level Modeling 66-3
66.2.1. Aggregating Units into Higher Echelon Units 66-4
66.3. Configuring Ammunition, Weapons, and Equipment 66-5
66.3.1. Defining Ammunition, Weapons, and Equipment Items ... 66-8
66.4. Assemblies 66-10
66.4.1. Creating Assemblies 66-11
66.4.2. Editing Assemblies 66-12
66.4.3. Adding Assemblies to a Unit 66-13
66.4.4. Removing Assemblies 66-13
66.5. Creating Roll Up Rules 66-14
66.6. Rolling Up Assemblies 66-15
66.7. Adding Activities 66-16
66.8. Setting Object Parameters for Aggregate-Level Modeling 66-17
66.8.1. Physical Parameters 66-18
66.8.2. Movement Parameters 66-19
66.8.3. Sensor Parameters 66-21
66.8.4. Attack Parameters 66-22
66.8.5. Defense Parameters 66-23
66.8.6. Electronic Warfare (EW) Parameters 66-23
66.8.7. Personnel and Equipment Parameters 66-24
66.8.8. Supplies Parameters 66-24
66.9. Configuring Combat Engineering Objects 66-25
66.10. Editing Reader/Writer Key/Value Pairs 66-26

Chapter 67. Adding DI-Guy Models to VR-Forces

67.1. Adding New DI-Guy Models to VR-Forces 67-2
67.1.1. Copy DI-Guy Files to the Data Directory 67-2
67.1.2. Add Model Definitions and Element Definitions 67-3
67.2. DI-Guy Configuration Files 67-10
67.2.1. Adding New Characters to the Character Digest 67-11
67.2.2. Adding a New DI-Guy Animation 67-11
67.3. How VR-Forces Maps DI-Guy Models 67-13
67.4. Add a New Entity in the Simulation Object Editor 67-13

Chapter 68. The Object Parameter Database

68.1. The Object Parameter Database 68-2
68.1.1. VR-Forces Classes and Object Parameters 68-3
68.1.2. Object Types 68-4
68.1.3. Parameter Types 68-8
68.1.4. Local and Remote Subcomponents 68-9
68.2. Component Descriptors and Parameters 68-13
68.2.1. Common Elements of Component Descriptors 68-13
68.2.2. Configuring the Priority of Components 68-16
68.3. System Definition Files 68-16
68.3.1. Creating a System Definition File 68-17
68.3.2. Configuring System Connections 68-18

68.4. Adding Variable Bindings to Platform and System Files 68-21
68.4.1. Adding a Variable Binding 68-22
68.4.2. Making Variables Editable in the Simulation Object Editor 68-23
68.4.3. Adding Platform Variable Bindings to the Simulation Object Editor 68-23
68.4.4. Adding System Definition File Variables to the Simulation Object Editor 68-25
68.5. Configuring Units 68-25
68.6. Configuring Tactical Graphics 68-26
68.7. Pathname Interpretation in Simulation Model Set Files 68-26
68.8. Updating Object Parameter Database Files for New Releases 68-27

Chapter 69. Using the OPD Editor

69.1. The Object Parameter Database Editor 69-2
69.2. Starting the OPD Editor 69-2
69.3. Loading a Simulation Model Set 69-3
69.4. Editing Platforms and Systems 69-4
69.4.1. Creating a New Platform or System from an Existing One . 69-5
69.4.2. Editing Individual Parameters 69-6
69.4.3. Deleting Parameters and Components 69-7
69.4.4. Restoring Parameters 69-7
69.5. Adding New Components and Parameters 69-8
69.5.1. Adding New Components 69-8
69.5.2. Changing A Component's Priority 69-9
69.5.3. Creating A Connection 69-10
69.5.4. Adding New Resource Parameters 69-11
69.6. Editing Variables 69-11
69.7. Regenerating Platform Files 69-12
69.8. Saving Changes to the Object Parameter Database 69-12

Chapter 70. SMS Configuration Files

70.1. Detection Tables 70-2
70.2. Probability and Selection Tables for Munitions 70-3
70.2.1. Ammo Select Tables 70-3
70.2.2. Hit Probability Tables 70-4
70.2.3. Damage Probability Tables 70-6
70.2.4. Configuring an Entity to Take Damage from a Munition .. 70-7
70.3. Configuring Formations 70-8
70.3.1. Formation Files 70-9
70.3.2. Creating a User-Defined Formation 70-10
70.4. Configuring Emitters 70-11
70.5. Configuring Radios 70-13
70.5.1. Publishing Radio Transmitters 70-13
70.6. Configuring Task Execution Rules 70-14
70.6.1. Configuring Action Categories 70-15

70.7. Configuring Obstruction Avoidance 70-15
70.7.1. Configuring Object Types 70-15
70.7.2. Configuring Vector Feature Types 70-16
70.8. Configuring Embarkation 70-16
70.9. Configuring the Effects of Suppressive Fire 70-17
70.10. Configuring VR-Forces Components on Remote Entities 70-18

Chapter 71. Configuring Sensors

71.1. Configuring Sensors 71-2
71.1.1. Adding a New Sensor to a Simulation Object 71-2
71.1.2. Editing a Sensor System Definition File 71-4
71.1.3. Specifying a Sensor Signature 71-5
71.1.4. Adding Detection Types or Why Doesn't My Sensor Detect Anything? 71-9
71.1.5. Adding a New Sensor Type 71-9
71.1.6. Illumination and Time of Day 71-12

Chapter 72. Configuring Joystick and Keyboard Controls

72.1. Configuring Joysticks and Keyboard Control 72-2
72.1.1. Creating a Joystick Configuration 72-3
72.1.2. Mapping the Joystick Stick 72-4
72.1.3. Mapping Joystick Buttons and Keyboard Keys 72-6
72.1.4. Configuring "Switch Between" Options 72-9
72.1.5. Specifying a Favorite Joystick Configuration 72-11

Section XIII Configuring the Graphical User Interface

Chapter 73. Display Engine and Window Management

73.1. The Display Engine 73-2
73.1.1. Window Types 73-2
73.2. Managing Display Engine Configurations 73-3
73.2.1. Adding a Window 73-3
73.2.2. Adding a Channel to a Window 73-5
73.2.3. Removing a Window 73-6
73.2.4. Removing a Channel 73-6
73.2.5. Saving a Display Engine Configuration 73-6
73.2.6. Loading a Display Engine Configuration 73-6
73.3. Changing a Window's Attributes 73-7
73.4. Changing a Channel's Attributes 73-8
73.4.1. Setting the Clipping Planes 73-10
73.4.2. Specifying the Projection Resize Policy Attribute 73-12
73.4.3. Changing a Channel's Frustum (Field of View) 73-15
73.4.4. Changing the Viewport 73-16
73.4.5. Configuring Water Visibility 73-17
73.5. Configuring Multichannel Displays 73-18

73.5.1. Changing the Camera's Position and Orientation Offset .. 73-20
73.6. Stereoscopic Displays 73-21
73.6.1. Configuring Anaglyphic Stereo 73-22
73.6.2. Configuring Polarized Stereo 73-23

Chapter 74. Common GUI Controls and Behaviors

74.1. Dockable Panels and Toolbars 74-2
74.1.1. Docking and Undocking a Panel 74-2
74.1.2. Docking and Undocking a Toolbar 74-3
74.1.3. Displaying and Hiding Panels and Toolbars 74-3
74.1.4. Restoring the Layout of Toolbars and Panels 74-4
74.2. Using Tear-Off Menus 74-5
74.3. Viewing the Window in Full Screen Mode 74-6
74.4. Using Context-Sensitive Menus 74-6
74.5. Common VR-Forces Buttons 74-7
74.6. Adding Text to the Title Bar 74-8
74.7. Specifying Display Units 74-9

Chapter 75. Configuring Toolbars, Dialog Boxes, and Menus

75.1. Introduction 75-2
75.2. Configuring Menus and Menu Commands 75-2
75.3. Configuring Dialog Box Pages 75-3
75.4. Configuring Context-Sensitive Menus 75-5
75.5. Configuring Toolbars 75-6
75.5.1. Creating Toolbars 75-6
75.5.2. Specifying an Icon for a Toolbar Item 75-7
75.6. Locking Main GUI Elements 75-7

Chapter 76. Creating and Editing Key Mappings

76.1. Introduction 76-2
76.2. The Key Mapping Editor 76-3
76.2.1. Binary Key Mappings 76-4
76.3. Editing a Key Map 76-4
76.3.1. Adding a Key Mapping 76-5
76.3.2. Changing a Key Mapping 76-7
76.3.3. Deleting a Key Mapping 76-7
76.4. Using Combined Key Mappings 76-8
76.5. Filtering the Function List 76-9
76.6. Creating a Key Map 76-9
76.7. Deleting a Key Map 76-9

Section XIV Appendixes

Appendix A. VR-Forces Configuration Files

A.1. Introduction A-2

A.2. The *vrfSim.mtl* Configuration File A-3
A.3. Configuring Communication Models A-8
A.4. MÄK Technologies Lisp (MTL) Files A-9
A.4.1. Using Environment Variables in an MTL File A-9
A.4.2. Specifying Lists of Lists A-10
A.4.3. Using Conditional Statements A-10

Appendix B. Object Parameters

B.1. The Structure of a Simulation Object Parameter B-2
B.2. The Parameter Type Hierarchy B-3
B.2.1. Vrf Base Object Parameters B-3
B.2.2. Vrf Object Parameters B-7
B.2.3. Moving Object Parameters B-8
B.2.4. Parameter Types for Entity Platforms B-8
B.3. Simulation Object Parameters B-8
B.4. Object Type Enumerations B-9
B.4.1. Object Super-type B-9
B.4.2. AggregateKind B-9
B.4.3. ObjectKind B-10
B.4.4. Unit Category B-10
B.4.5. Unit Specific B-11
B.4.6. Point Object Category B-12
B.4.7. LinearObjectKind B-12
B.4.8. LinearObjectCategory B-12
B.4.9. Interaction B-12

Appendix C. Systems and System Usage

C.1. VR-Forces Systems C-2

MÄK 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 and Ballistic Missiles*, 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.
  - Chapter 13, *Using a Communications Effects Server* explains how to configure VR-Forces to work with external communications effects servers, such as QualNet.
  - Chapter 14, *Example Entity-Level Scenarios* walks you through the process of creating two of the sample scenarios provided with VR-Forces.
- Section III, "Simulation Objects"
  - Chapter 15, *Introduction to Simulation Objects* contains conceptual details about simulation objects.
  - Chapter 16, *Creating and Placing Objects*, explains describes the common procedures for creating simulation objects, tactical graphics, and props.
  - Chapter 17, *Selecting Objects* describes procedures common to simulation objects, control objects, and tactical graphics.
  - Chapter 18, *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 19, *Creating Simulation Objects*, explains the fine points of creating simulation objects, with a particular emphasis on simulation objects in entity-level scenarios. It also covers embarkation and collision avoidance.
  - Chapter 20, *Generating Traffic and Crowds* explains how to use the Spawn Pattern Manager to create pattern of life traffic.
- 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, Sets, and Scripted Tasks"
  - Chapter 25, *Assigning Tasks*, provides general information about assigning tasks and creating behavior sets.

  - Chapter 26, *Tasks for Entity-Level Scenarios*, describes the tasks provided with VR-Forces that apply to simulation objects in entity-level scenarios.
  - Chapter 27, *Tasks for Aggregate-Level Scenarios*, describes the tasks that apply to simulation objects in aggregate-level scenarios.
  - Chapter 28, *Creating Scripted Tasks*, introduces scripted tasks and explains how to create them. It does not cover how to write scripts for scripted tasks.
  - Chapter 29, *Writing Scripts for Scripted Tasks*, introduces the Lua script API for VR-Forces and explains how to write scripts for scripted tasks.
  - Chapter 30, *Setting Simulation Object State*, describes how to change the state of simulation objects using set data requests.
- Section VI, *Plans*
  - Chapter 31, *Plan Concepts* provides conceptual background to help understand the procedures in Chapter 32, *Writing Plans*.
  - Chapter 32, *Writing Plans* explains how to write plans for simulation objects and how to write global plans.
- Section VII, "Tactical Graphics"
  - Chapter 33, *Introduction to Tactical Graphics* is a brief introduction to tactical graphics.
  - Chapter 34, *Overlays* describes the creation and use of overlays.
  - Chapter 35, *Creating and Editing Tactical Graphics* describes how to create tactical graphics other than points, lines, and areas, and how to edit tactical graphics.
  - Chapter 36, *Remote Graphics* explains how to enable display of graphics generated by remote applications.
- Section VIII, "Visual and Audio Effects"
  - Chapter 37, *Displaying Tactical Information and Effects* describes how to display a variety of simulation object effects such as trailing effects, shadows, and sensor volumes.
  - Chapter 38, *Lighting Effects* describes the lighting effects supported by VR-Forces and how to manage them.
  - Chapter 39, *Marine Effects* describes water related effects, such as displaying rain splash and water spray. It also covers wake effects and entity buoyancy.
  - Chapter 40, *Using Sensors* explains how to enable and configure infrared and night vision goggle sensor views.
  - Chapter 41, *Intervisibility* explains how to determine if simulation objects can see other simulation objects or areas of the terrain.
  - Chapter 42, *Sound Effects* explains how to enable and configure sound effects for entities.
- Section IX, "The Observer and Viewing Scenarios"

  - Chapter 43, *The Observer and Observer Modes* explains what an observer is and the different ways observers can be configured.
  - Chapter 44, *Moving the Observer* explains how to navigate the 3D and 2D views by moving the observer.
  - Chapter 45, *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 46, *Inset Views, Sensor FOVs, and HUDs* describes special secondary windows – inset views, sensor fields of view, and instrument panels, and HUDs.
- Section X, "Terrain"
  - Chapter 47, *Introduction to Terrain Usage in VR-Forces* provides conceptual background for terrain use in VR-Forces.
  - Chapter 48, *Displaying Terrain Information* describes ways to change how terrain is displayed. These options provide different types of information about the terrain.
  - Chapter 49, *Using Terrain Profiles* explains how to display and configure terrain profiles.
  - Chapter 50, *Composing Terrains* explains how to build terrain databases by importing elevation data, imagery, and feature data.
  - Chapter 51, *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 52, *Editing Earth Files* explains the syntax of earth files so that you can customize them for viewing streaming terrain from terrain servers.
  - Chapter 53, *Generating Navigation Data* explains how to generate navigation data for advanced navigation by humans and ground entities.
  - Chapter 54, *Processing MetaFlight Files* explains how to preprocess MetaFlight files so that VR-Forces can load them.
  - Chapter 55, *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 56, *Terrain Tutorials* provides step-by-step instructions for creating and saving a terrain database.
- Section XI, "Configuring Visual Models"
  - Chapter 57, *Model and Element Definitions* explains how to create and edit model schemas, model definitions, and element definitions.
  - Chapter 58, *Mapping Models and Effects* explains how map entity enumerations to the models configured in VR-Forces.
  - Chapter 59, *Configuring Specific Visual Models* explains how to configure some specific visual models, including 2d Icons, cockpits, and speedtrees.

  - Chapter 60, *Configuring Emitter Volumes* explains how to configure color, radius, and segment size for emitter volumes.
  - Chapter 61, *The WRM Specification (DIS Notes)* explains the WRM specification for 3D modeling.
  - Chapter 62, *Model Tutorials* provides step-by-step instructions for creating a model and mapping it to an entity enumeration.
- Section XII, "Creating and Editing Simulation Models"
  - Chapter 63, *The Simulation Object Editor and SMSs* explains simulation model sets (SMSs) and how to create them in the Simulation Object Editor.
  - Chapter 64, *Editing Simulation Object Models* explains how to use the Simulation Object Editor to edit simulation object parameters, systems, and models. It covers procedures that are common to entity-level and aggregate-level SMSs and specific to individual entities.
  - Chapter 65, *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 66, *Editing Aggregate-Level Objects* explains how to configure aggregates for aggregate-level scenarios in the Simulation Object Editor.
  - Chapter 67, *Adding DI-Guy Models to VR-Forces* explains where to put the files for DI-Guy models and how to configure them.
  - Chapter 68, *The Object Parameter Database* describes the object parameter database, which contains parameters for simulation objects and control objects, and how to edit it.
  - Chapter 69, *Using the OPD Editor* explains how to use the OPD Editor to edit entity and system parameters. The OPD Editor lets you edit all simulation object parameters, while the Simulation Object Editor only lets you edit some parameters.
  - Chapter 70, *SMS Configuration Files* describes additional configuration files that VR-Forces uses to manage munitions, formations, and joystick use.
  - Chapter 71, *Configuring Sensors* explains how to configure sensors and sensor signatures.
  - Chapter 72, *Configuring Joystick and Keyboard Controls* explains how to configure joysticks and keyboard keys to control entity movement.
- Section XIII, "Configuring the Graphical User Interface" GUI use and customization.
  - Chapter 73, *Display Engine and Window Management* explains how to configure windows and channels and how to save and load display configurations.
  - Chapter 74, *Common GUI Controls and Behaviors* explains how to manage toolbars and other aspects of the VR-Forces GUI.
  - Chapter 75, *Configuring Toolbars, Dialog Boxes, and Menus* explains how to configure the menus, toolbars, and dialog box pages that get displayed.

  - Chapter 76, *Creating and Editing Key Mappings* explains how to edit key mappings and create new key mappings.
- Section XIV, "Appendixes".
  - Appendix A, *VR-Forces Configuration Files* explains the syntax of configuration files and the parameters in the configuration files provided with VR-Forces.
  - Appendix B, *Object Parameters* describes the high level parameters used in OPD files.
  - Appendix C, *Systems and System Usage* lists all the systems provided with VR-Forces and provides information about them, including which entities use them.

MÄK Products Glossary defines terms common to DIS, HLA, and MÄK 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, the OPD Editor, 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.

MÄK Products

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

- MÄK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MÄK 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 MÄK'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.

- MÄK 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™. The DI-Guy product line is a set of software tools for real-time human visualization, simulation, and artificial intelligence. Every DI-Guy software offering comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy enables the easy creation of crowds and individuals who are terrain aware, autonomous, and react intelligently to ongoing events. Save time, money and create outstanding simulations with DI-Guy. The DI-Guy product line includes the following products:
  - The DI-Guy SDK. Embed the DI-Guy library in your real-time application and populate your world with lifelike human characters.
  - DI-Guy Scenario™. Author and visualize human performances in a rich, user-friendly graphical environment. Use DI-Guy Scenario as an end visualization application or save scenarios and load them into your DI-Guy SDK enabled application.

  - ECOSim. Enhanced Company Operations Simulation (ECOSim) is a company-level training simulation that teaches leaders how best to deploy troops, UAVs, convoys, and other assets. ECOSim focuses on ease-of-use, rapid scenario generation, runtime operator control, and realistic and reactive human simulation.
  - DI-Guy AI. Generate crowds of autonomous characters to quickly populate your worlds with hundreds and thousands of terrain-aware, collision avoiding DI-Guys. Used as a module on top of DI-Guy Scenario and DI-Guy SDK.
  - Expressive Faces Module. Enable DI-Guy characters to have faces that display emotion, eyes that look in directions and blink, and lips that sync to sound files.
  - DI-Guy Motion Editor. Create or customize motions to your particular needs in an easy-to-use graphical application.
- RadarFX. RadarFX is a client-server application that can 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. VR-Forces includes a sample client application.

How to Contact Us

For VR-Forces technical support, information about upgrades, and information about other MÄK 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
VR-Vantage support:	

Internet

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

Post

Send postal correspondence to:	VT MÄK 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 MÄK 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 *vrforces4.4.1/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

MÄK software products may use code from third parties. This section contains the license documentation required by these third parties.

Boost License

VR-Link, and all MÄK 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 MÄK software that uses VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MÄK Products. MÄK 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 MÄK 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

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 1-6
- Realistic Display of Vehicles and Terrain (3D View) 1-7
- Create Complex Scenarios 1-8
- Simulation Object Types Supported 1-9
- Mission Planning 1-11
- Simulation Object Tasks 1-11
- Scripted Tasks 1-12
- Tactical Graphics 1-12
- Terrain Agility and Composability 1-13
- Flexible, Intuitive Graphical User Interface 1-14
- Overlays 1-14
- Behaviors 1-14
- Observer Attach Modes 1-15
- Special Effects and Simulation Object Information Visualization 1-15
- Dynamic Ocean 1-15
- Lighting Effects 1-15
- Accurate Vehicle Positioning 1-16
- Batch Mode Operation 1-16
- Remote Control 1-16

The VR-Forces Toolkit 1-17
- Plug-in Architecture 1-17

DI-Guy 1-18

Support for External Communications Servers 1-18

Helpful Utilities 1-19
Third-Party Software and Content 1-20
SilverLining 1-20
Triton Ocean SDK 1-21
GL Studio 1-21
SpeedTree 1-22
3D Models, Terrain, and Graphical Content 1-22
OpenSceneGraph 1-23
osgEarth 1-23
Distributed Simulation Standards Supported 1-23

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 both 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 MÄK products, such as VR-Vantage Stealth and MÄK 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 MÄK 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/projects/osg/wiki/About/Introduction)

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		
Advanced lighting	X	
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	
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	
Intervisibility	X	X
Paged terrain	X	X
Range rings	X	X

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

Feature	3D	2D
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	
SpeedTrees	X	
Streaming buoys and beacons	X	X
Streaming terrain	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 the following sections:

- "Entity-Level Modeling and Aggregate-Level Modeling," on page 15-9.
- Section IV, "Modeling Units".
- Chapter 22, *How Aggregate-Level Modeling Works*.

For information specific to the different types of modeling, please see the following sections:

- Chapter 26, *Tasks for Entity-Level Scenarios*.
- Chapter 27, *Tasks for Aggregate-Level Scenarios*.
- Chapter 65, *Editing Units for Entity-Level Scenarios*.
- Chapter 66, *Editing Aggregate-Level Objects*.

Terminology

We use the following terms to refer to the different types of objects that VR-Forces simulates:

- Entity. An individual player in a scenario, such as a car, a helicopter, or a human.
- Unit. A hierarchical grouping of entities, such as a platoon or a company.
- Simulation object. A collective term for entities and units.
- Cultural feature. A man-made object, such as a bench or a building that is simulated as an entity.
- Tactical graphic. Graphic objects such as points, lines, and areas that are drawn on overlays. Simulation objects are aware of some types of tactical graphics, such as control objects and some kinds of lines and points. Others are simply drawings on the screen.
- Combat engineering objects. In aggregate-level scenarios, objects such as fortified lines, minefields, and bridges that are visually similar to tactical graphics, but which the back-end simulates and simulation objects can interact with.
- Props. Terrain features that can be individually manipulated.

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 50, *Composing Terrains*. For details about models, please see Chapter 58, *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.
  - An order of battle.
  - Tactical graphics.
  - A plan for each simulation object in the order of battle.
- 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 Chapter 30, *Setting Simulation Object State*.

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 5-17.

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 `When` statements and `While` 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 the order of battle and tactical graphics in response to events.

For more information, please see Chapter 32, *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

You can create new tasks by writing scripts using the Lua scripting language. Input dialog boxes are created automatically and tasks are added to the Task 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 28, *Creating Scripted Tasks*.

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 29-29.

You can organize scripted tasks into groups called Behavior Sets. By assigning Behavior Sets to forces you can control whether or not tasks are available to simulation objects in those forces. For more information, please see "Using Behavior Sets to Manage Scripted Tasks," 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 34, *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 (MÄK 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 50, *Composing Terrains*. For a list of supported file formats, please see "Supported File Formats," on page 47-2.

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. 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 74, *Common GUI Controls and Behaviors*. For descriptions of individual windows and dialog boxes, please see online help.

1.1.11. 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 34-1.

1.1.12. 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.

1.1.13. 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 44, *Moving the Observer*.

1.1.14. 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.

1.1.15. 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.

1.1.16. Lighting Effects

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

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

Shader-based Effects Maps

VR-Forces supports several types of shader-based effects 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 effects 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 38, *Lighting Effects*.

1.1.17. 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 6-16 and "Using Ground Clamping," on page 19-21.

1.1.18. 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 MÄK Data Logger, for later review. For more information, please see "Running VR-Forces in Batch Mode," on page 7-36.

1.1.19. Remote Control

You can control the VR-Forces observer remotely through view control messages, which can be generated by MÄK applications such as MÄK Data Logger, and by programs that use the VR-Vantage Control Toolkit. For details, please see "Controlling the Observer from Other Applications," on page 44-32.

You can also send view control messages from plans. For details, please see "Sending View Control Messages," on page 32-19.

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 MÄK 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.

1.4. Support for External Communications Servers

You can configure VR-Forces to use an external communications effects server to control the transmission of radio messages between VR-Forces simulation objects. The communications effects server determines which simulation objects are able to communicate with each other, and how long it takes a message to be transmitted from the originator of the message to the receiver of the message.

Any communications effects server that meets the defined interface can be used. For information about the interface, please contact support@mak.com. For more information, please see Chapter 13, *Using a Communications Effects Server*.

1.5. 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 63, *The Simulation Object Editor and SMSs*.
- The Object Parameter Database (OPD) Editor is a graphical user interface for editing the object parameter database. Using the OPD Editor can speed up editing of the object parameter database and prevent syntactical errors that can occur when you use a text editor to edit parameters. For details, please see Chapter 69, *Using the OPD Editor*.
- The vrfMsgDump utility displays the contents of VR-Forces messages discovered on the network.
- 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 52-14.
- The medfTool lets you encrypt 3D model files. For details, please see "Compressing Model Files," on page 59-23.
- The mftTool lets you process MetaFlight terrains so that they will work in VR-Forces. For details, please see "The mftTool," on page 54-3.
- osgearth_cache. This utility lets you cache terrain data from earth files offline. For details, please see "Caching osgEarth Terrain Data Offline," on page 50-32.

1.6. Third-Party Software and Content

The VR-Forces front-end 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, Terrex (now Presagis), 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.

1.6.1. 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 MÄK 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.

1.6.2. 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 MÄK 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.

1.6.3. 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.

1.6.4. 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 MÄK 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.

1.6.5. 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.

1.6.6. 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/projects/osg. It is distributed under the OpenSceneGraph Public License (OSGPL), which is based on the GNU Lesser General Public License (LGPL).

MÄK 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.

1.6.7. 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).

MÄK 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.

1.7. 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 5-17. 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 2-2
Uninstalling VR-Forces on Windows 2-3
The VR-Forces Directory Structure 2-4
Installing and Setting Up the MÄK License Manager 2-6
Specifying the License Server 2-7
Installing an RTI 2-9
Installing the MÄK 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
Merging Translation Files 2-16

2.1. Installing VR-Forces

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

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

VR-Forces has two installers, an application installer and a data installer. You must run both installers. You must run the application installer first.

To install the VR-Forces toolkit, you must enter an installation code. Be sure you have received your code from your MÄK 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.

2.1.1. Installing VR-Forces on Windows

Note the following:

- You must have administrator privileges to install MÄK products on Windows Vista.
- When you install large applications on Windows Vista or later, 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 delay can also take place between the time that VR-Forces or VR-Vantage unpack compressed data files and the time the data installer starts. 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 and whether it is a 32 bit or 64 bit version. (Some installers have both 32 bit and 64 bit libraries and are not labeled for the number of bits.)

➤ To install VR-Forces, run the installers. Follow the instructions in the installation wizards. (You must run the application installer and the data installer.)

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. Uncompress and untar the data package into the same directory as the application.

2.1.3. Uninstalling VR-Forces on Windows

Because installation of VR-Forces is a two step process, uninstallation is also a two-step process.

To uninstall VR-Forces on Windows:

1. In the VR-Forces root directory, run *unins001.exe*.
2. In the VR-Forces root directory, run *unins000.exe*.

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\vrforces4.4.1*.)

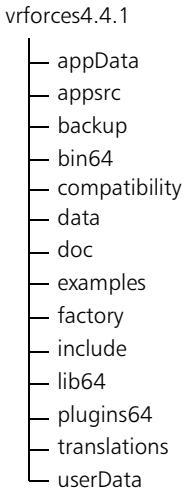


Figure 2-1. VR-Forces directory structure

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	Product executables and DLLs.
./compatibility	Header files for compatibility with releases before VR-Forces 4.3.
./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 MÄK.
./include	Header files for use by API developers.
./lib64 or */lib*	Library files.

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

Directory	Directory contents
./plugins64 or *./plugins*	Plug-in DLLs.
./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 MÄK License Manager

Before you can use a MÄK product, you must obtain a valid license file, install the MÄK 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 MÄK 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 MÄK 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 MÄK installation media. It is separate from the product installers. You can download the installers from our web site at http://www.mak.com/license-support.html or you can download directly from:

- Windows:
  ftp://ftp.mak.com/out/MAKLicenseManager-win-setup.exe (for 32-bit systems)
  ftp://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe
- Linux:
  ftp://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz (for 32-bit systems)
  ftp://ftp.mak.com/out/MAKLicenseManager-linux64-setup.tar.gz

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 MÄK 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.

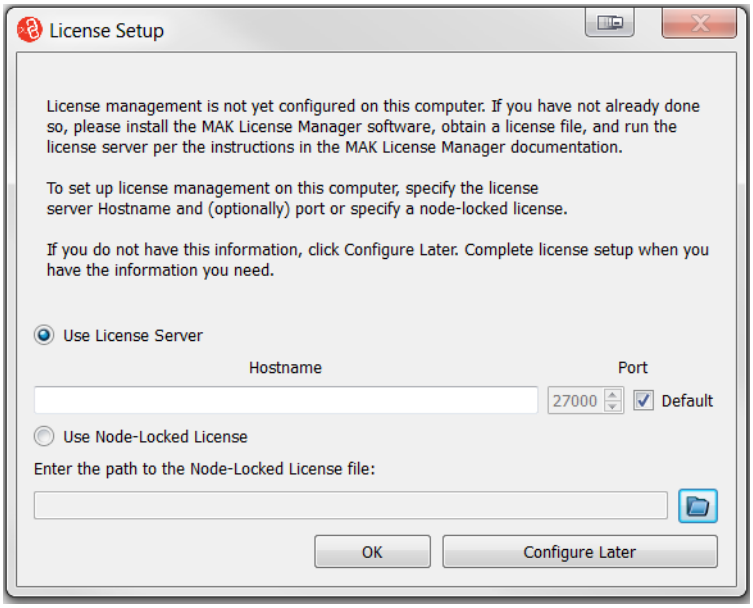


Figure 2-2. License Setup dialog box

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 MÄK applications until you set up license management.

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

Under limited circumstances, MÄK 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 MÄK 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 MÄK application.
- You can clear the saved license configuration by deleting the cache file. On Windows 7, 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 MÄK products run on.

Windows

To add the MAKLMGRD_LICENSE_FILE in Windows:

1. Open the Control Panel.
2. On Windows 7, click System and Security.
3. Click System.
4. In the sidebar menu, click the Advanced System Settings. The System Properties dialog box opens.
5. Click Environment Variables. The Environment Variables dialog box opens.
6. Click New. The New System Variable dialog box opens.
7. In the Variable Name field, enter MAKLMGRD_LICENSE_FILE.
8. In the Variable Value field, enter `@server_name`, where `server_name` is the name of the license server.
9. Click OK to back out of each dialog box and set the variable.



Different versions of Windows have slightly different wording of the various Control Panel options.

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 MÄK 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 MÄK products, please see the release notes for your MÄK application.

2.4.1. Installing the MÄK RTI

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

Configuring Your System to Use the MÄK 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 MÄK 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 MÄK RTI

To run a MÄK application with the MÄK 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 MÄK RTI, however you can run the rtiexec if you want to. (It is required to use certain features of the MÄK 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 MÄK 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 MÄK 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).

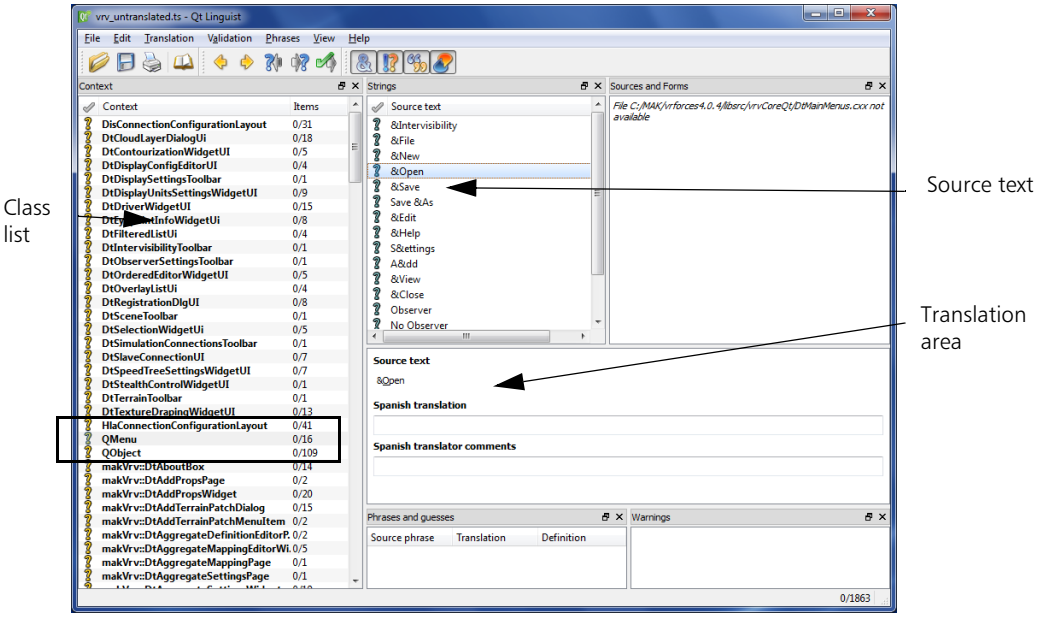


Figure 2-4. Linguist window

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.

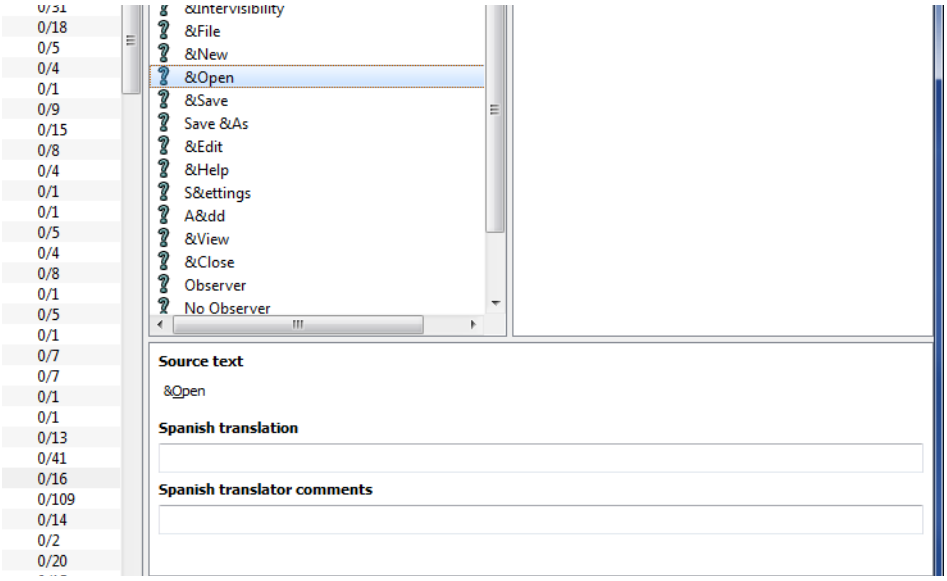


Figure 2-5. Text selected for translation

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.

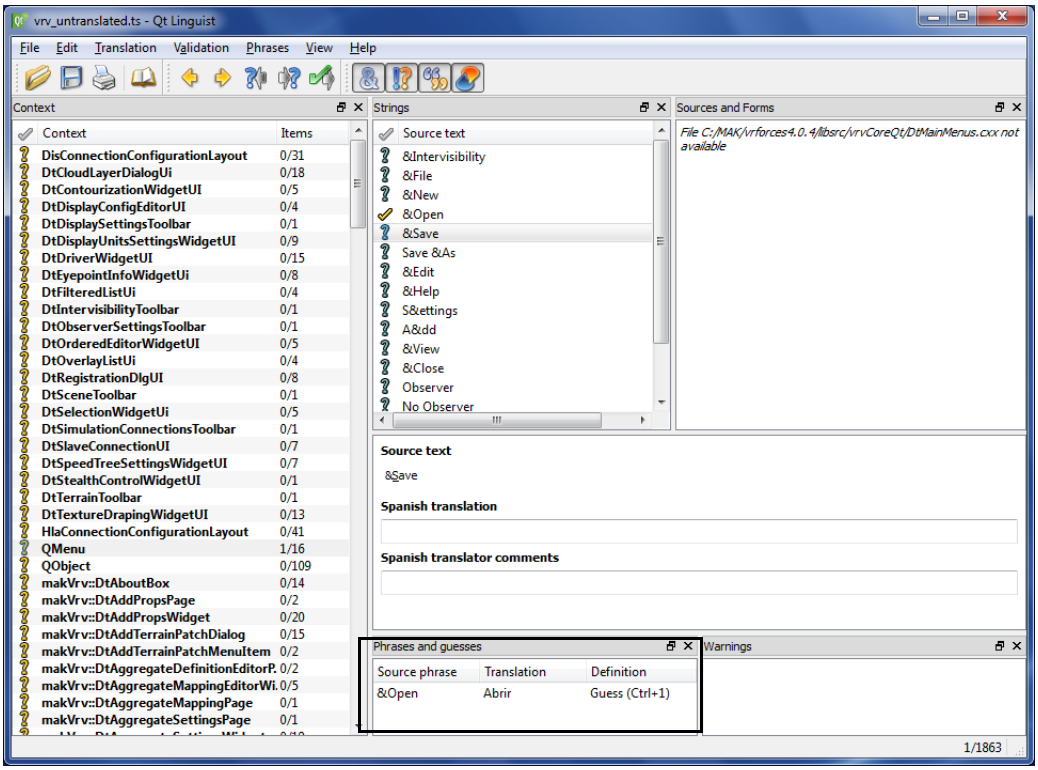


Figure 2-6. Completed translation for Open

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 *product_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 scripted tasks 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 29-21. 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.

2.5.4. Merging Translation Files

If you localize a version of VR-Forces and upgrade to a newer version, you can merge your localized *.ts* file with the *.ts* file from the new version. Then you only have to translate new or changed text in the new version. You merge translation files with the *translationMerge* utility, which is in the *./bin64* directory. The syntax is as follows:

```
translationMerge [-o] [-m] previous.ts current.ts new.ts
```

where:

- *`previous.ts`* is the name of the previously localized translation file.
- *`current.ts`* is the name of the translation file provided with the current release of VR-Forces.
- *`new.ts`* is the name to give to the merged file.
- `-o` specifies that the merged file removes obsolete messages.
- `-m` merges modules that have equal text strings.



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 3-11
Interactions 3-12
Advanced Terrain Navigation 3-12
How Navigation Data is Generated 3-12
Path Finding 3-14

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 32-22 and "Configuring VR-Forces Components on Remote Entities," on page 70-18.)

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.

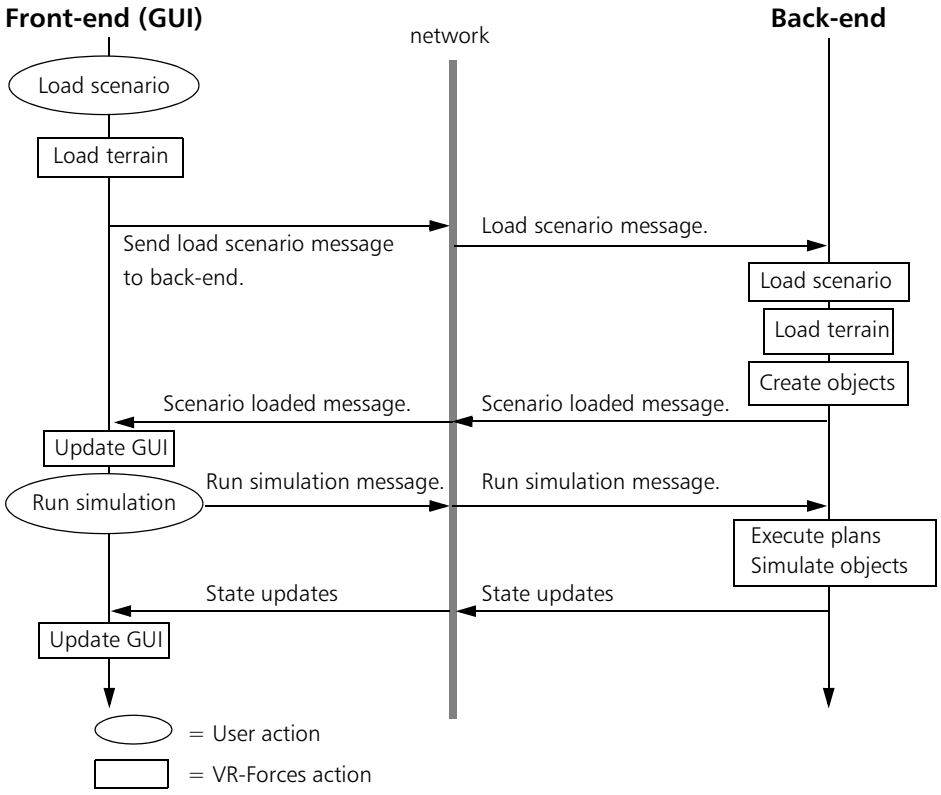


Figure 3-1. Front-end and back-end interaction

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.

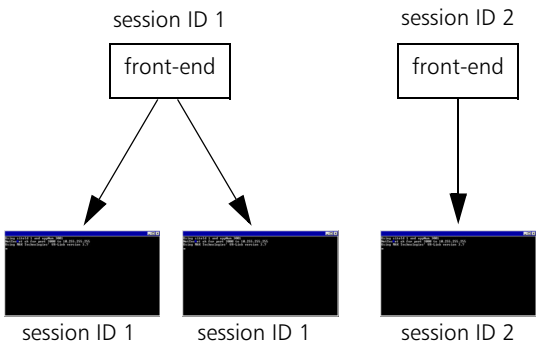


Figure 3-2. Using session ID to control back-ends

- 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.

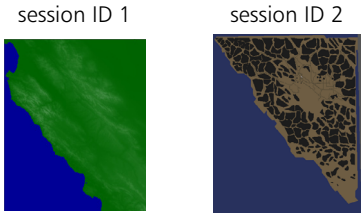


Figure 3-3. Using session IDs to load different terrain databases

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 16-4.)

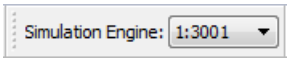


Figure 3-4. Selected Simulation Engine toolbar

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 `-T` command-line option. The back-end will not receive any order of battle data, 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 a Scenario," on page 7-19.

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 Enumerations for Simulation Interoperability*). 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 an SMS for entity-level scenarios, *EntityLevel.sms*, and one for aggregate-level scenarios, *AggregateLevel.sms*.

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 32-22 and "Configuring VR-Forces Components on Remote Entities," on page 70-18.

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.

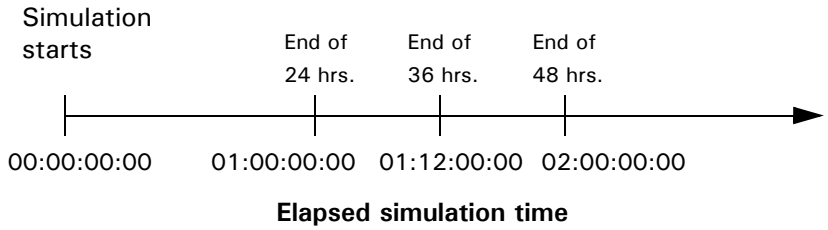


Figure 3-5. Simulation time

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 11-2.

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, is starts over at 00:00.

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 then 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.



- 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.

3.6.1. 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 life-form can move in spaces that a ground vehicle cannot move in. Figure 3-6 shows a NavMesh for a terrain that has several areas in which entities cannot move.

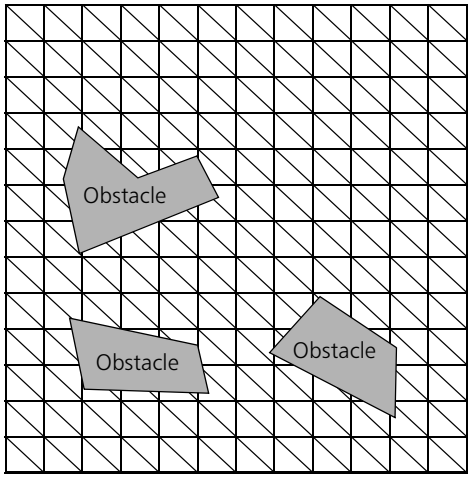


Figure 3-6. NavMesh

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 3-7 shows a hypothetical graph of the paths that an entity can follow in the NavMesh shown in Figure 3-6.

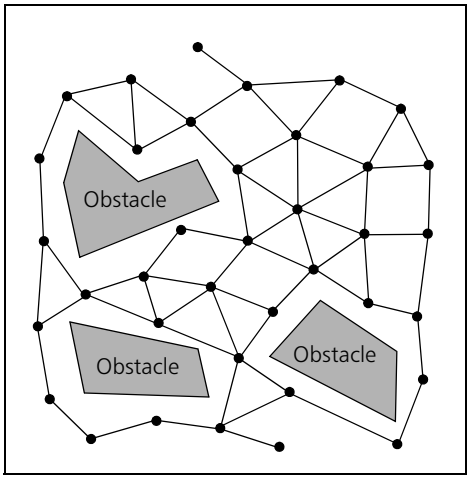


Figure 3-7. Graph

For details about generating navigation data, please see Chapter 53, *Generating Navigation Data*.

3.6.2. 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.

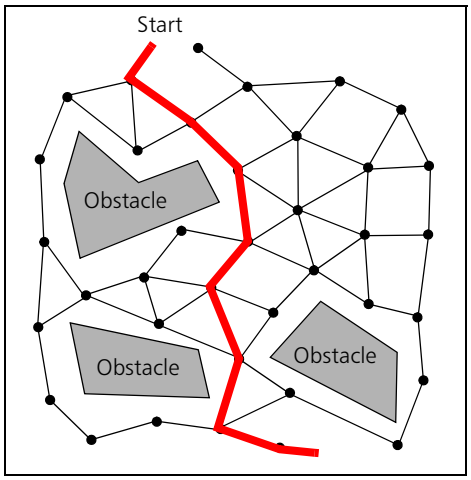


Figure 3-8. Planning a path

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 3-8, 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 3-9 and 3-10 illustrate this process. Using the prospective path defined in Figure 3-8, the entity computes its actual path. In Figure 3-9, view 1, it tests the first three vertices (dashed lines) and finds that it can reach all of them directly. In Figure 3-9, 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 3-10 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.

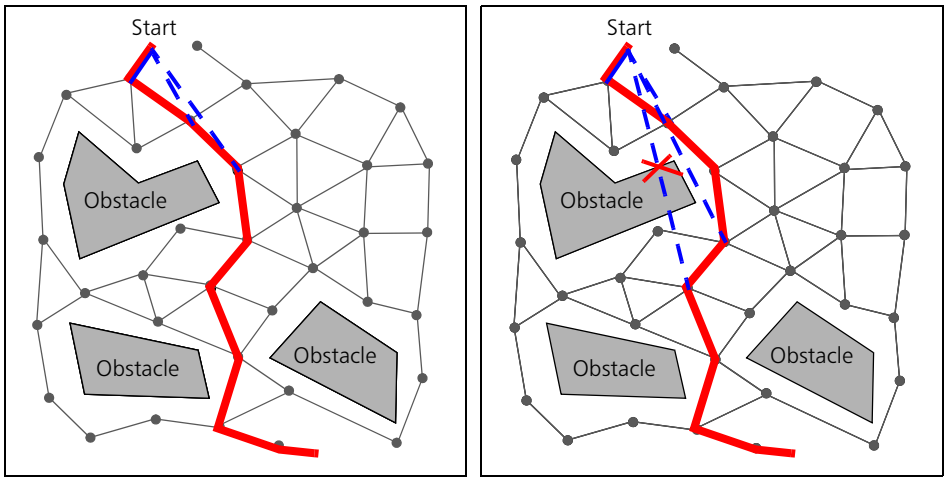


Figure 3-9. Checking next vertex

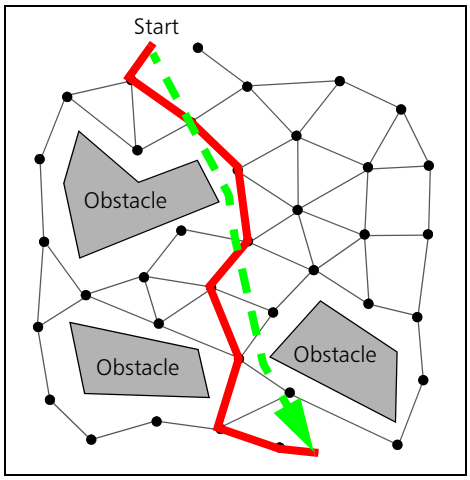


Figure 3-10. Final calculated path

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.

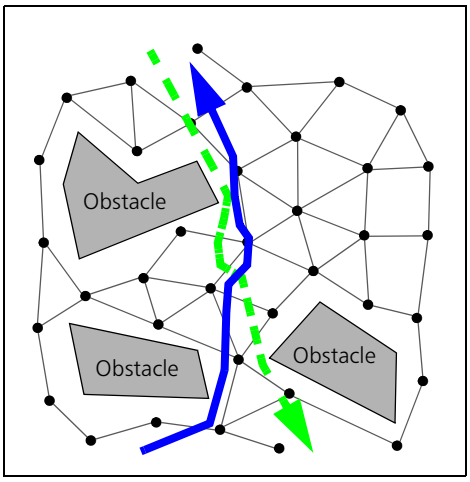


Figure 3-11. Collision avoidance

In Figure 3-11, two entities traveling in opposite directions avoid each other, then resume their routes. Compare the track of the dashed line to that in Figure 3-10.



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 4-10

The VR-Forces Window 4-12
- Opening New Windows 4-14
- Printing the VR-Forces Display 4-14

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

Using HLA Time Management 4-18
- 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
- Closing a Terrain 4-23

Managing VR-Forces Settings 4-23
- Synchronizing Settings Among VR-Forces Installations 4-25
- Global Settings and Observer-Specific Settings 4-25

Exiting VR-Forces 4-25

Configuring Simulation Connections 4-26
- Opening the Simulation Connections Configuration Dialog Box 4-26

Adding a Simulation Connection 4-27
Editing a Simulation Connection 4-28
Simulation Connection Parameters 4-28
Deleting a Simulation Connection 4-30
Configuring Auto Connect 4-31
Displaying Connection Information 4-32
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 Windows 7, on the Start menu, choose **Programs** → **MAK Technologies** → **VR-Forces 4.4.1** → **VR-Forces** *configuration*, where *configuration* is:

   – VR-Forces GUI

   – VR-Forces Simulation Engine

   – VR-Forces GUI + Simulation Engine.

   On Windows 10, on the Start menu, choose **All apps** → **MAK Technologies** → **VR-Forces** *configuration*.

   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 5-13. 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 DIS
   ```

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).

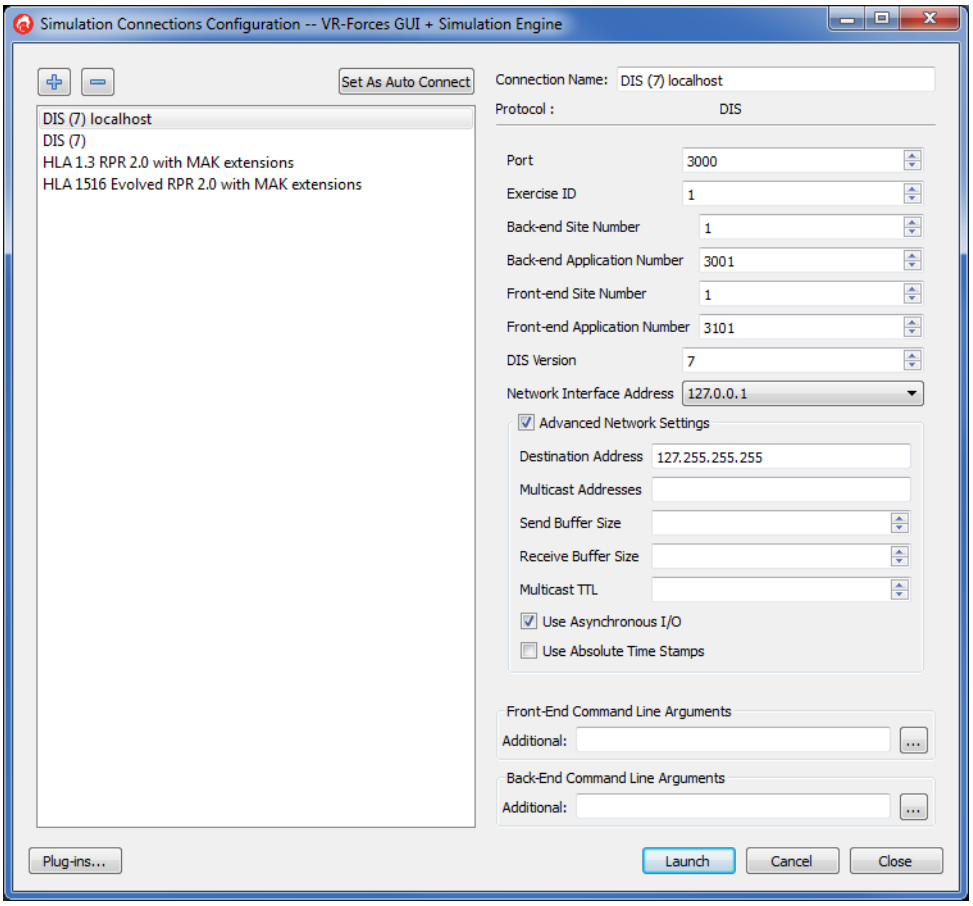


Figure 4-1. Simulation Connections Configuration dialog box (DIS)

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 4-26.)
3. Optionally, edit the connection's parameters. If you change a parameter, it is saved automatically.
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 line, 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.

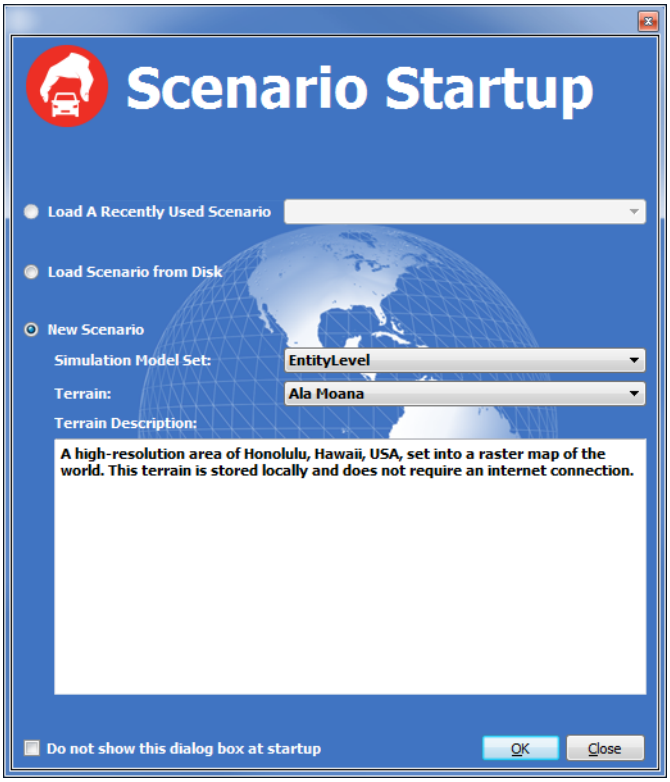


Figure 4-2. Scenario Startup dialog box

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 Session Options page (Figure 4-3).

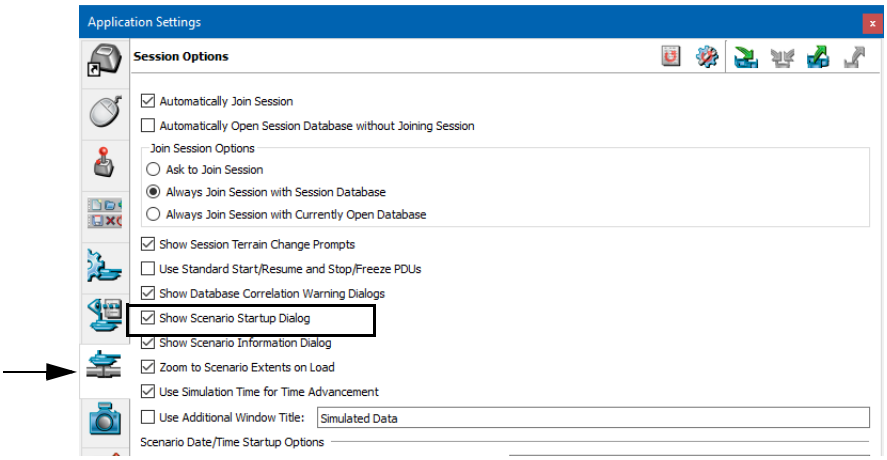


Figure 4-3. Session Options page

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

To start independent VR-Forces executables:

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
   vrfSimprotocol -s 2 -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 4-10.

4.1.3. Specifying a Session ID

➤ To assign a session ID, 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.

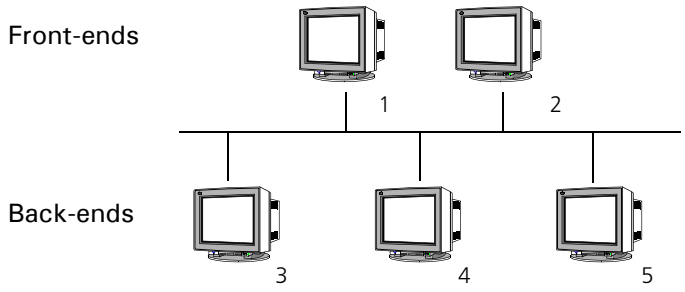


Figure 4-4. VR-Forces multicomputer configuration



As an alternative to starting individual front-ends and back-ends from the command line and specifying their protocol information in the command line, if you do not need to specify other command-line options, 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 5-13.

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.

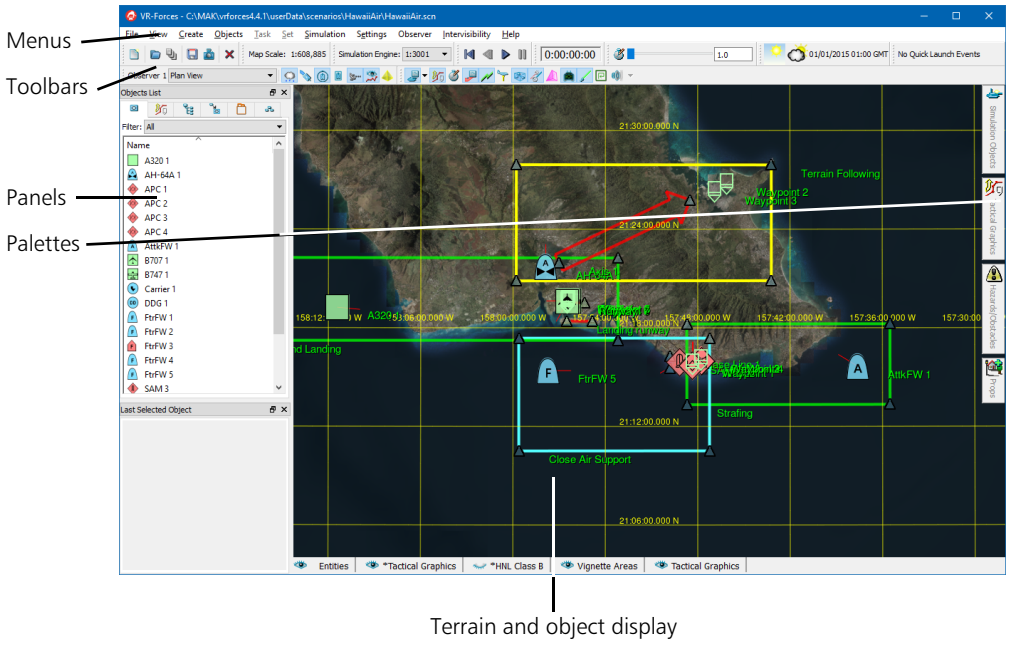


Figure 4-5. VR-Forces window and toolbars (2D)

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

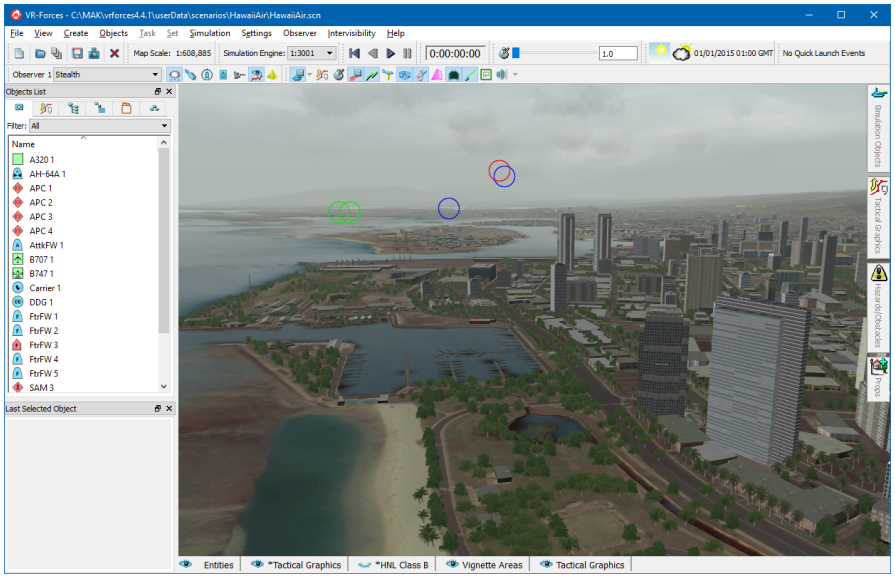


Figure 4-6. VR-Forces window (3D)

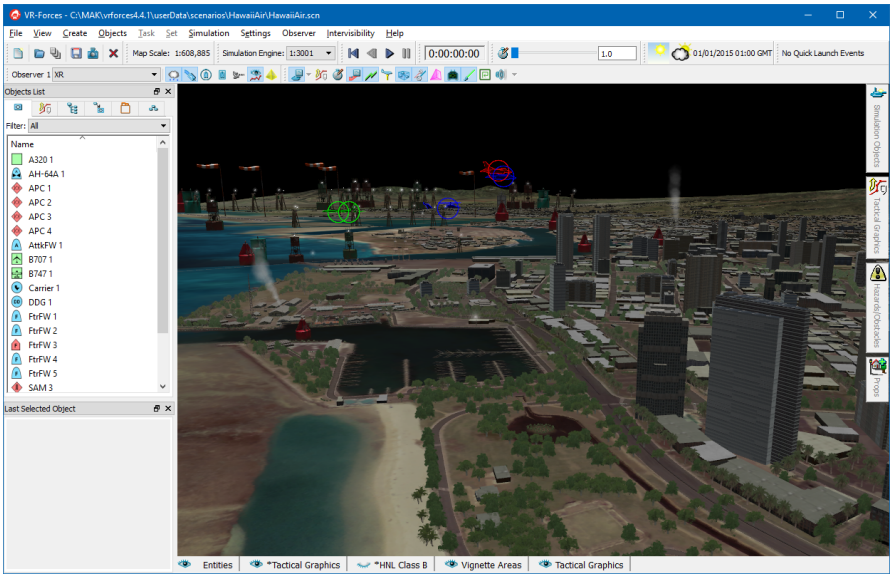


Figure 4-7. VR-Forces window (XR)

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 74, *Common GUI Controls 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 46-2.

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** → **Print**. The Print dialog box opens.
2. Select a printer.
3. Click Print.

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 options set in the Application Settings dialog box, Session Options 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.

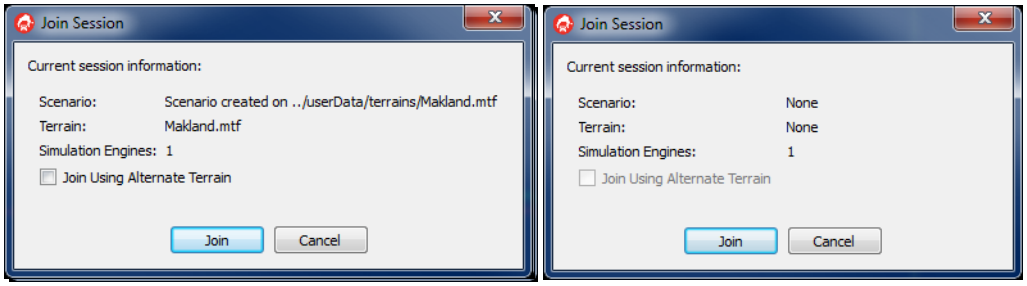
! If you do not load the terrain specified by the scenario, it is possible that visual and simulation terrains will not be synchronized.

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 4-8). If a scenario is running, its details are listed.



Scenario running | No scenario running

Figure 4-8. 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 Options 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 follows:

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 checkbox 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 4-8) 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 Database Correlation Warning Dialogs. If enabled, VR-Forces displays a warning if the terrain in the front-end might not correlate with the terrain loaded in the back-end.
- 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 MÄK 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 using HLA 1.3, the sample FED file configured for use with time management, and the federation name VR-LinkTM:

```
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 48-18.)

You can only "open" terrains in MTF format. To open any other terrain format, you must compose a terrain, as described in Chapter 50, *Composing Terrains*, or use an earth file, as described in Chapter 51, *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 59-23.) If you want to disable runtime caching, you can do so as described in "Configuring File Caching," on page 50-31.

! 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** → **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 4-9).

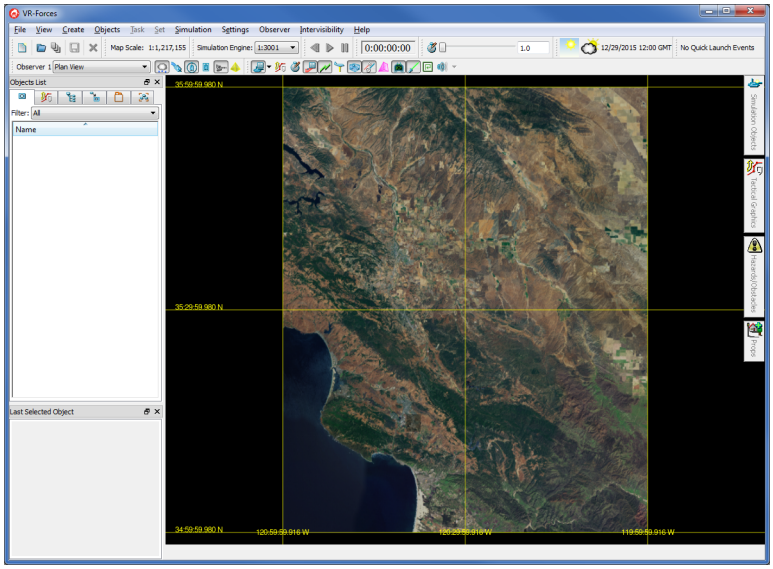


Figure 4-9. VR-Forces window with a database loaded

- 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 spacebar (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. Closing a Terrain

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

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

4.6. 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 4-25.

- 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.
- 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 a 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. However, the way that VR-Forces saves or restores settings depends on the type of setting, as follows:

- When you initiate a save or restore operation on independent settings, it only affects 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 save or restore operation on dependent settings, all of the changes to all of the related settings get saved or restored. This ensures that the relationships do not get broken.

(Figure 4-10) illustrates the settings management buttons.



Figure 4-10. Settings buttons

- Revert to the default settings. Override any changes made to settings during this session.
- Restore the factory settings. Change settings to the settings used when VR-Forces was installed. (The factory settings and default settings may be different.)
- Import settings from a file. Replace the current settings with those stored in the selected file.
- Merge in settings from a file. (Only supported for dependent settings.) 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 4-11 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.

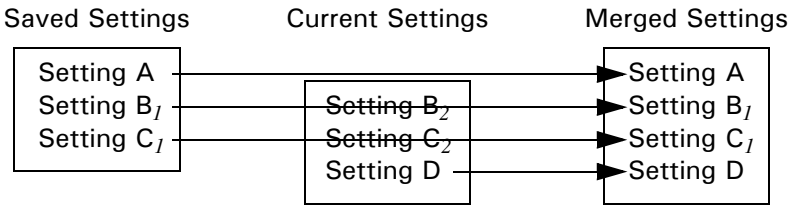


Figure 4-11. Merged settings

- Export settings to a file. Save the current settings to the selected file.
- Save the selected setting to a file. (Only supported for dependent settings.)



- 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 settings to *./appData/settings/vrfGui*.
- If you load an alternative settings file at startup, it is treated like an imported settings file. If you save current settings as default, they get saved to the default files in *./appData/settings/*. They do not get saved back to the alternative file unless you explicitly export them to that file.

4.6.1. 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.

4.6.2. 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.

4.7. Exiting VR-Forces

When you exit VR-Forces from the GUI, all GUI windows close. In combined mode, the simulation engine also closes. If you are not running in combined mode, you must also close all simulation engines.

➤ To exit the VR-Forces GUI, choose **File** → **Exit**.

➤ To exit the simulation engine, type **`q`** and press **`Enter`**.

4.8. 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 MÄK extensions.
- HLA 1516 Evolved RPR FOM 2.0 with MÄK 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.

4.8.1. 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, 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, on the Windows 7 Start menu, choose **Programs** → **MAK Technologies** → **VR-Forces 4.4.1** → **Tools** → **Configure Connections**. On Windows 10, on the Start menu, choose **All apps** → **MAK Technologies** → **Configure 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 5-13.

4.8.2. Adding a Simulation Connection

To add a simulation connection:

1. Open the Simulation Connections Configuration dialog box, as described in "Opening the Simulation Connections Configuration Dialog Box," on page 4-26.

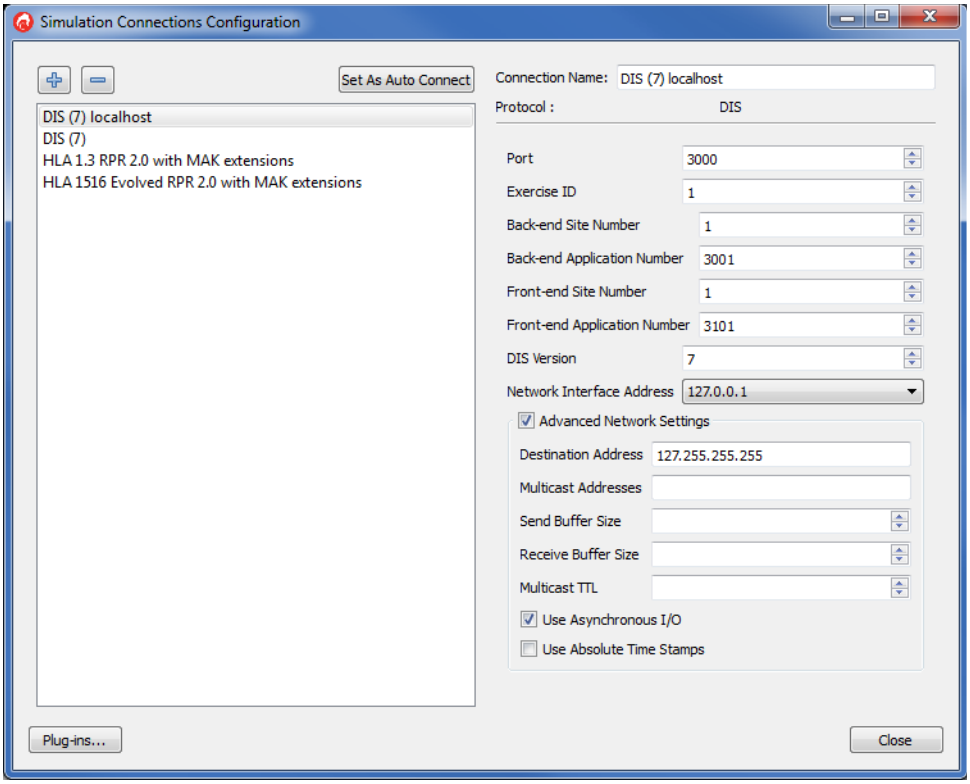


Figure 4-12. Simulation Connections Configuration dialog box

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

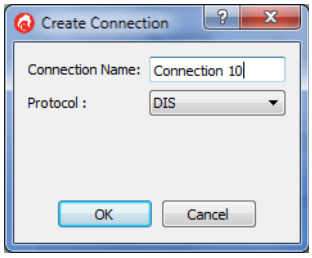


Figure 4-13. Create Connection dialog box

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.

4.8.3. Editing a Simulation Connection

To edit a simulation connection:

1. Open the Simulation Connections Configuration dialog box, as described in "Opening the Simulation Connections Configuration Dialog Box," on page 4-26.
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.

4.8.4. Simulation Connection Parameters

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

Table 4-2: Connection configuration parameters

Parameter	Command-line equivalent	Description
DIS		
Port	`-P`	Specifies the DIS port. "Specifying the Port Number," on page 5-19.
Exercise ID	`-x`	Specifies the DIS exercise ID. "Specifying the Exercise ID," on page 5-20.
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 5-14.
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 5-14.

Table 4-2: Connection configuration parameters

Parameter	Command-line equivalent	Description
DIS Version	`--disVersion`	Specifies the DIS version: 4, 5, or 6. Default: 6. "Specifying the DIS Version," on page 5-20.
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 5-19.
Multicast Addresses	`-S`	Specifies one or more multicast addresses. "Specifying Point-to-Point or Multicast Operation," on page 5-19.
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.
Use Absolute Time Stamps	`--useAbsolute-TimeStamps`	Specifies use of absolute time stamps instead of relative time stamps.
HLA		
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 5-16.
FED File Name	`-F`	Specifies the FED file to use. "Specifying the FED File," on page 5-17.
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 5-14.

> **!** 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 4-2: Connection configuration parameters

Parameter	Command-line equivalent	Description
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 5-14.
		! 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.
RPR FOM Version	`--rprFomVersion`	"Specifying the RPR FOM Version," on page 5-17.
FOM Mapper Library	`-f`	Specifies the VR-Link FOM Mapper to use. "Specifying a FOM Mapper," on page 5-17.
Initialization String	`--fomMapperInitData`	Specifies initialization data required by the FOM Mapper, if any. "Specifying FOM Mapper Initialization Data," on page 5-18.
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.
Use Absolute Time Stamps	`--useAbsoluteTimeStamps`	Specifies use of absolute time stamps instead of relative time stamps.

4.8.5. Deleting a Simulation Connection

To delete a simulation connection:

1. Open the Simulation Connections Configuration dialog box, as described in "Opening the Simulation Connections Configuration Dialog Box," on page 4-26.
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.

4.8.6. 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. Open the Simulation Connections Configuration dialog box, as described in "Opening the Simulation Connections Configuration Dialog Box," on page 4-26.
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 **Programs** → **MAK Technologies** → **VR-Forces 4.4.1** → **Tools** → **Configure 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.

4.8.7. 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).

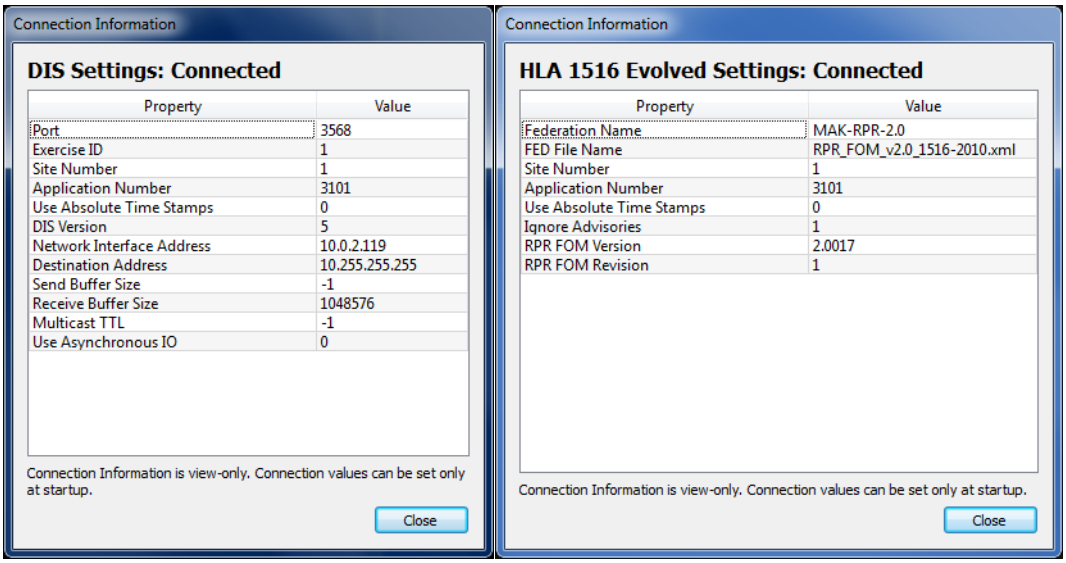


Figure 4-14. Connection Information dialog box

4.9. Managing Plug-ins

One way to extend the functionality of VR-Forces is by loading plug-ins that have been written for it by MÄK 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.

4.9.1. 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).

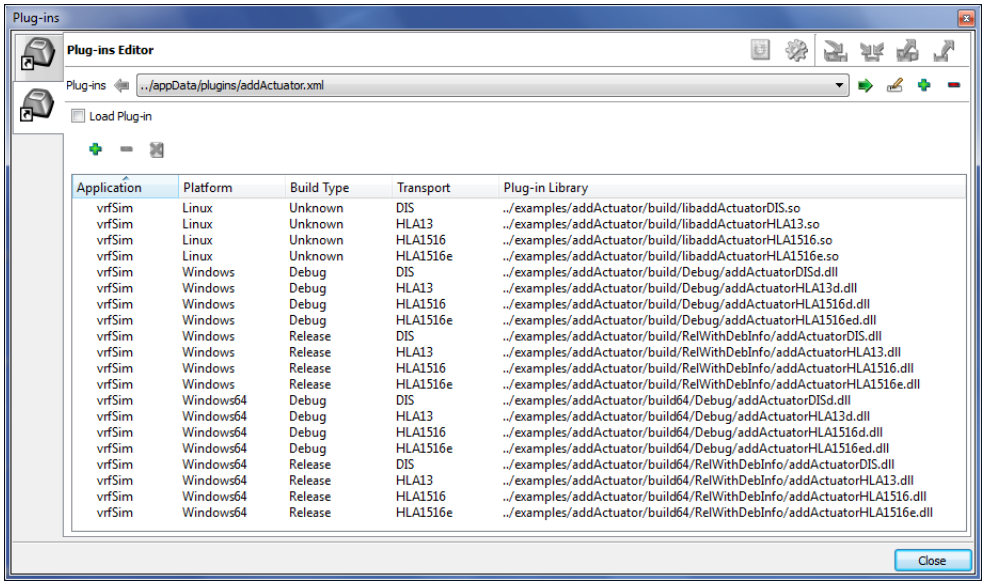


Figure 4-15. Plug-ins dialog box, Plug-ins Editor page

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.

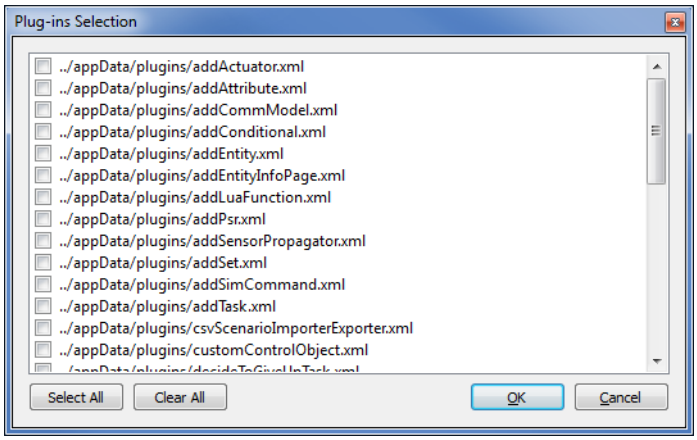


Figure 4-16. Plug-ins Selection dialog box

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.

4.9.2. 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).

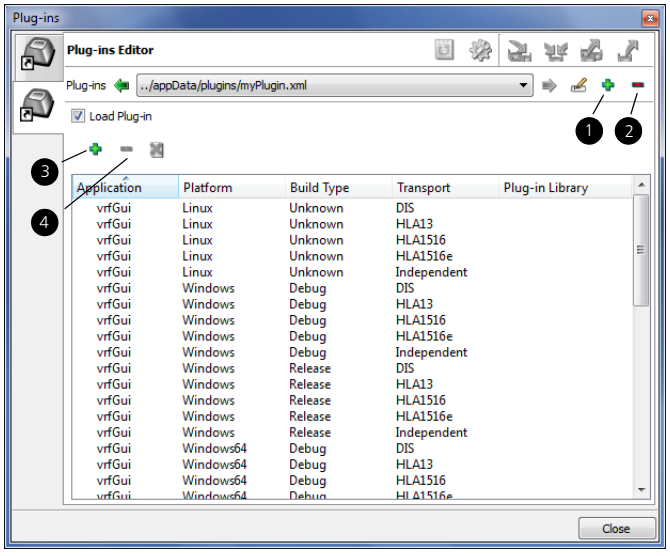


Figure 4-17. New plug-in

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.

4.9.3. 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).

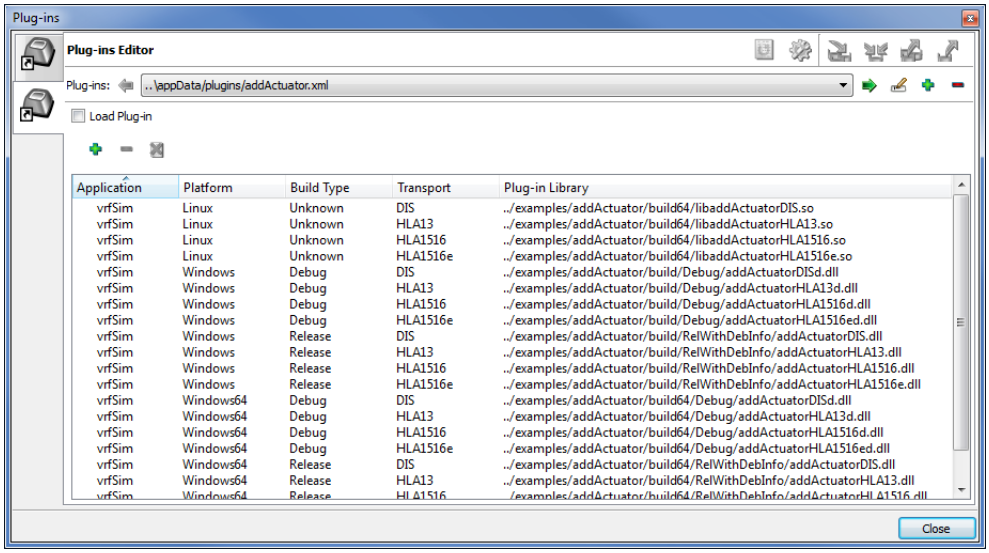


Figure 4-18. Plug-ins Editor

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 ( ).

4.9.4. 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).

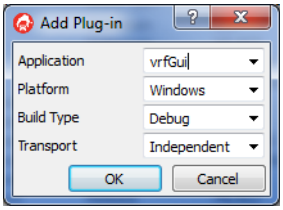


Figure 4-19. Add Plug-in dialog box

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.

4.9.5. 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).

4.9.6. 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).

4.9.7. 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).

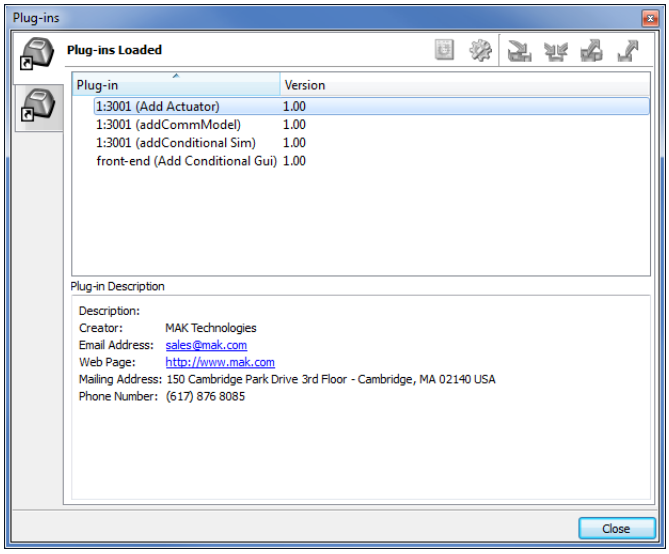


Figure 4-20. Plug-ins dialog box, Plug-ins Loaded page

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

4.10. 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 5-3
Command-line Options for vrfSim 5-8
Command-line Options for vrfLauncher 5-13
Running in Combined Mode from the Command Line 5-14
Using the -- Command-line Argument 5-14
Protocol-Independent Command-line Options 5-14
Specifying the Site ID and Application Number 5-14
Specifying the Language to Use in the GUI 5-15
Setting the Notification Level 5-15
Loading Plug-ins 5-16
Command-line Options for HLA Federations 5-16
Specifying the Federation Execution 5-16
Specifying the FED File 5-17
Specifying the RPR FOM Version 5-17
Specifying a FOM Mapper 5-17
Specifying the RPR FOM Revision 5-17
Specifying FOM Mapper Initialization Data 5-18
Specifying a FED File That is Appropriate for the FOM Mapper 5-18
Enabling Time Management 5-18
Command-line Options for DIS Exercises 5-19
Specifying the Port Number 5-19
Specifying Point-to-Point or Multicast Operation 5-19
Specifying the Multicast Time-to-Live 5-19
Using Asynchronous I/O 5-19

Specifying the Exercise ID 5-20
Specifying the DIS Version 5-20

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 5)

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.
--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).
--defaultSimulationModel-Set *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.
--dispSetting *filename*	Specifies the display engine configuration to load.
--doNotCheckPluginVersions	Do not check the version in a plug-in to make sure it can load in VR-Forces. A plug-in gets built against a particular version of VR-Forces. Ordinarily, when VR-Forces loads a plug-in, it checks to be sure it was created with that version of VR-Forces. This option prevents that check.
--doNotLoadVrfPlugins	Prevents loading of plug-ins in the *./plugins* directory.
--entDispSetting *filename*	Specifies the entity display setting to load.
--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	Set the root directory used to restore settings to factory defaults.

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

Option	Description
`--fileTransporterReceive-Port` *port*	Port for the file transporter. This is the port used to receive order of battle and plan files.
`--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).
`--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 *./bin64* 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 | --batchedRendering)`	Disables instance rendering. Default: Disabled.
`(-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.

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

Option	Description
`--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 4-33.
`--loadSimulationObject-Group` *group*	Creates a new scenario on the simulation object group editing terrain and creates an instance of the simulation object group on that terrain.
`(-n | --notifyLevel)` *notify_level*	Specifies the notification level for application messages, 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 5-15.
`--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.
`--pvd`	Start the VR-Forces front-end with only 2D modes available.
`(-S | --suppressHatInter-sectOnAttach)`	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-Forces 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	Specifies a terrain (MTF) file to load.
`--unplug` *filename*	Do not load the specified plug-in. Can be used multiple times.

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

Option	Description
`--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-Forces 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 | --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.
`(-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)

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

Option	Description
(-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 MÄK 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.
--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.
(-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 5)

Option	Description
(-a | --appNumber) *application_number*	Specifies the application number. For details, please see "Specifying the Site ID and Application Number," on page 5-14.
--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 7-40.
(-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.
--disableCallbackQueue	Disables all main thread parallel algorithms.
--disableParallelTick	Execute all tick code serially.
--doNotLoadVrfPlugins	Do not load any VR-Forces plug-ins from the *./plugins* directory.

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

Option	Description
`--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 *./bin64* directory. Mutually exclusive with the `-T` option. For details, please see "Loading a Scenario from the Command Line," on page 7-18.
`--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 SISO 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 5-15.
`--noRtiCompilerCheck`	Disable RTI Compiler version check.
`--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.

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

Option	Description
(-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 4-40.
(-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 5-14.
--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.
--startMinimized	Minimizes the vrfSim console at startup.
(-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 3-7 and "Loading a Terrain Database at Startup," on page 4-22.
--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.

! 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 4 of 5)

Option	Description
`--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.
`(-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 | --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.

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

Option	Description
DIS Only	
`(-A | --destAddrString)` *address*	Destination address.
`--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.
`--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.
`--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.
`(-h | --help)`	Displays help information in a console window.
`--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 MÄK 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 connection. The command would be:

```
vrfLauncher --usePredefinedConnection DIS
```

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 MÄK 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.

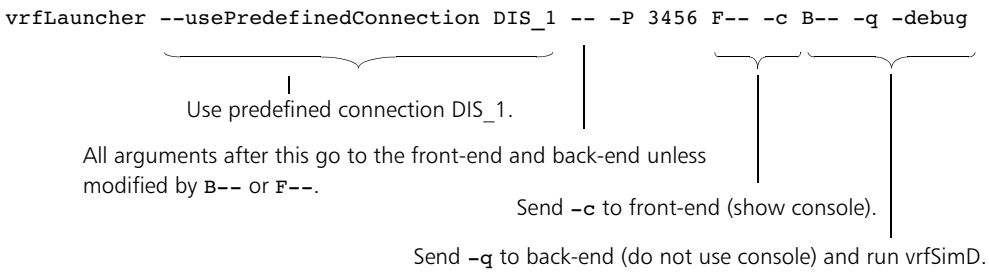


Figure 5-1. Using the -- command-line argument

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 3000; 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 2-11.

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 4-33.)

➤ 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 4-39.

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 MÄK RTI and are running multiple, concurrent federation executions, you must run the rtiexec. In other words, you cannot use the MÄK RTI in lightweight mode if you are running multiple federations. Please see the MÄK 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 1.0.

You can specify a different RPR FOM version (0.5, 0.7, 0.8, 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

RPR FOM revision 2 for version 2.0017 fixes an encoding and decoding size issue with object ID lists to match RPR FOM 2 draft 17.

➤ 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.

By default VR-Forces uses a FED file called *MAK-RPR2-1-1.fed*, but you can specify an alternate 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.

➤ 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 6-5
Displaying Performance Statistics 6-6
Displaying the VR-Forces Function Profiler 6-7
Displaying OSG Statistics 6-8
Filtering Simulation Objects Using Interest Management 6-8
Enabling Interest Management 6-8
Configuring Interest Management 6-9
Miscellaneous Performance Configuration Options 6-11
Limiting Use of Spot Reports 6-11
Using Asynchronous I/O for DIS Exercises 6-11
Using Incremental Compiling with Streamed Data 6-11
Setting the Tick Rate 6-12
Specifying the Tick Rate for Components 6-12
Tuning the Network Interface 6-13
Tuning the State Repository Tick Rate 6-13
Configuring Graphics Quality 6-13
Balancing Visual Quality Against Network Performance 6-14
Enabling Configuration of Performance Options 6-15
Trajectory Smoothing 6-16
Configuring Trajectory Smoothing 6-17

Tuning the Target Frame Rate 6-17
Coloring Draw Calls 6-18
Configuring VSync 6-19

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 6-14.)
- Turning off processor-intensive features. (For details, please see "Miscellaneous Performance Configuration Options," on page 6-11.)
- Adjusting the network processing time.
- Specifying the tick rate for a simulation object or component. (For details, please see "Setting the Tick Rate," on page 6-12.)
- Specifying a tick rate for the network interface. (For details, please see "Tuning the Network Interface," on page 6-13.)
- Specifying the VR-Forces sleep time. (For details, please see "Tuning the Target Frame Rate," on page 6-17.)
- Specifying the number of objects that you expect to simulate in the exercise.
- Optimizing the terrain.
- Using interest management. (For details, please see "Filtering Simulation Objects Using Interest Management," on page 6-8.)

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 for VR-Forces," on page 59-22.
- 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 6-6.

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.
- Anti-aliasing settings. For details, please see "Configuring Graphics Quality," on page 6-13.

The following sections describe options for optimizing front-end performance:

- "Configuring SpeedTree Trees," on page 59-18.
- "Configuring File Caching," on page 50-31.
- "Clearing the Model Instancing Cache," on page 2-30.
- "Setting the Clipping Planes," on page 73-10.
- "Enabling Texture Compression," on page 50-33.
- "Considerations and Limitations for Building Terrains," on page 50-4.
- "Filtering Simulation Objects Using Interest Management," on page 6-8.
- "Setting the Ocean LOD Elevation," on page 50-16.
- "Configuring the Ocean Height Map," on page 50-17.
- "Displaying Wakes and Spray Effects," on page 39-5.
- "Configuring the Skybox Cube Map," on page 38-12.
- "Improving Performance when Attaching to Ground Objects," on page 45-6.
- "Enabling Buoyancy for Surface Entities," on page 39-6.
- Chapter 38, *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 lists statistics for each back-end (Figure 6-1).

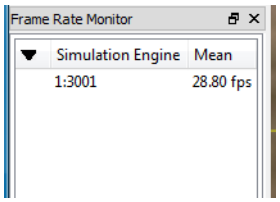


Figure 6-1. Frame Rate Monitor

➤ To view the Frame Rate Monitor, choose **View** → **Frame Rate Monitor Panel**.

6.3. Displaying Performance Statistics

VR-Forces can display two kinds of performance statistics — OSG statistics (Figure 6-2), and VR-Forces statistics (Figure 6-3). VR-Forces also has a function profiler that developers can use to assess the performance of new code.

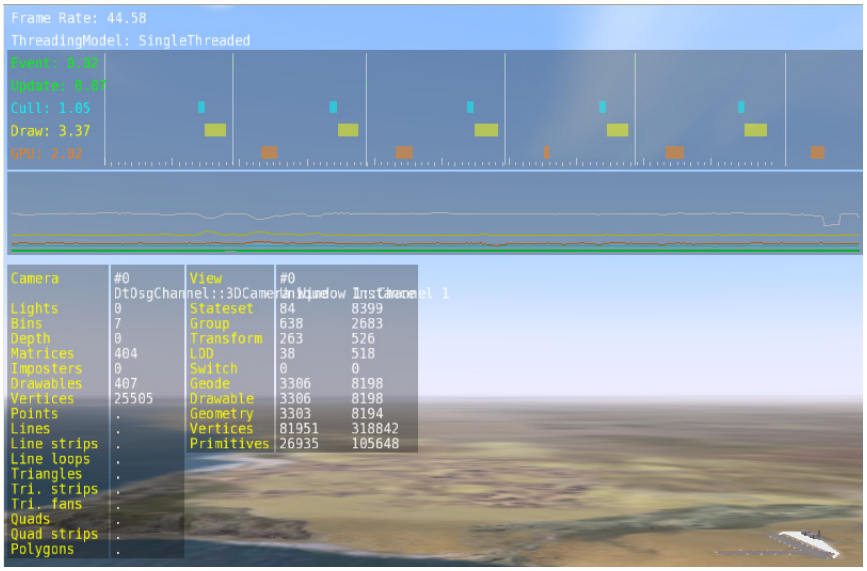


Figure 6-2. OSG performance statistics

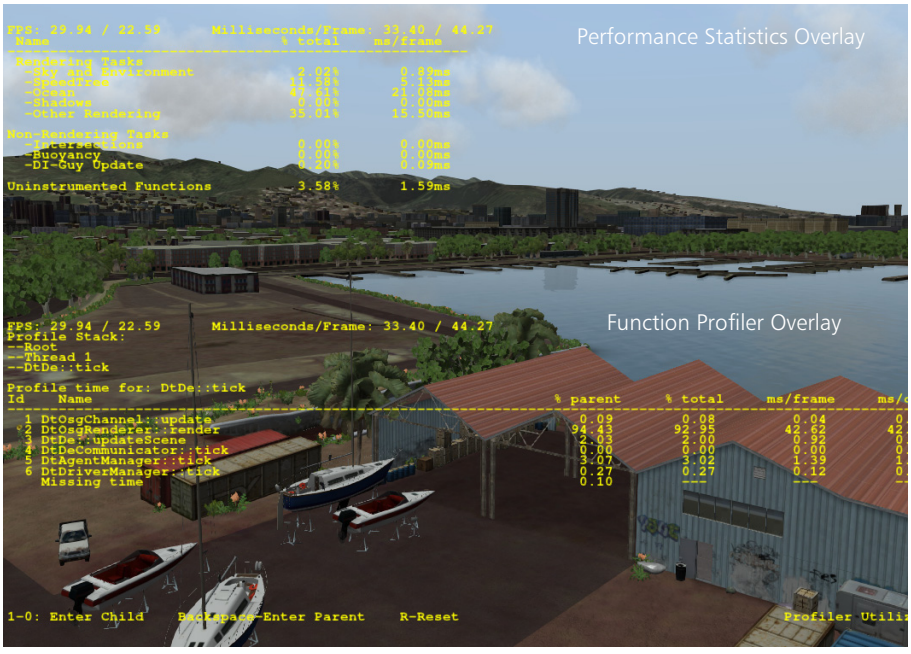


Figure 6-3. VR-Forces performance statistics and function profiler



Displaying performance statistics may affect performance. Displaying OSG statistics may affect performance more than VR-Forces statistics.

To display VR-Forces performance statistics:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-4).

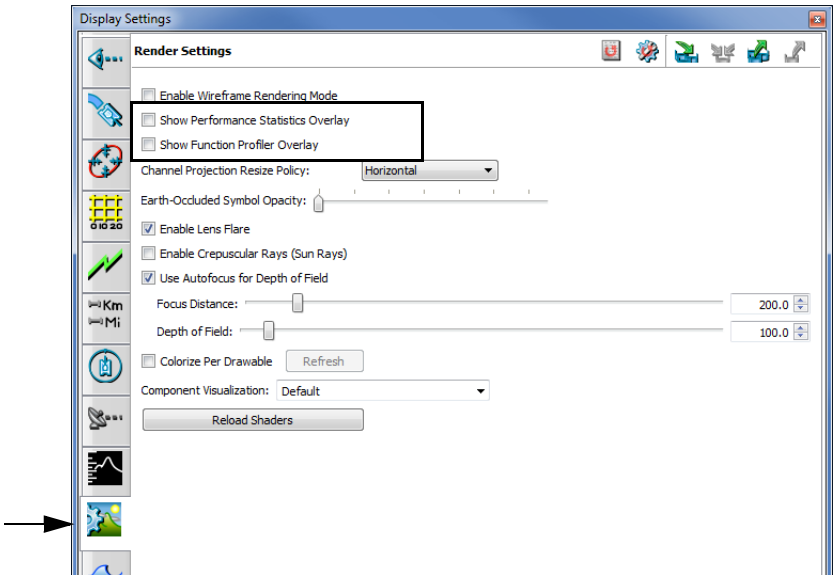


Figure 6-4. Render Settings page

3. Select the Show Performance Statistics Overlay check box. A statistics display is added to the window (Figure 6-3).

6.3.1. 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*.

To run the VR-Forces Profiler:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 6-4).
3. Select the Show Function Profiler Overlay check box. A statistics display is added to the window (Figure 6-3).

6.3.2. 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. 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.

6.4.1. Enabling Interest Management

To use interest management:

- You must use HLA (DIS does not support interest management).
- If you are using the MÄK RTI, you must use MÄK RTI 3.4a or later.
- You must configure your RTI to support DDM. (If you are using the MÄK 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
```

6.4.2. 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 6-5).

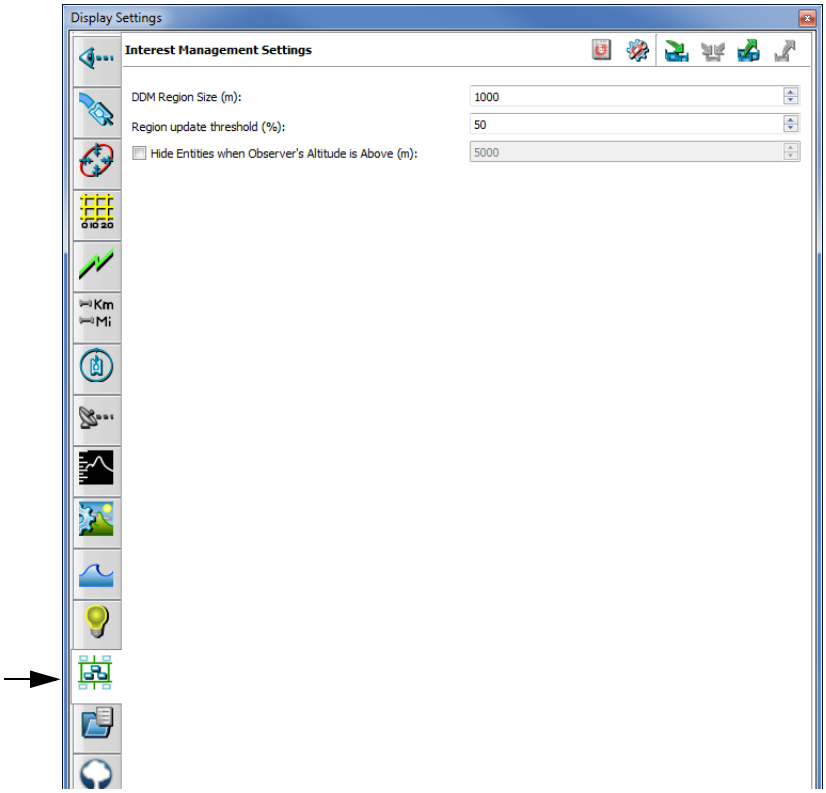


Figure 6-5. Interest Management Settings page

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.

6.5. 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, using the OSG_ICO environment variable.

6.5.1. 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 Simulation Objects Based on Spot Reports," on page 9-2.

6.5.2. 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.

6.5.3. Using Incremental Compiling with Streamed Data

Users of streamed terrains might want to consider using 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. It is only recommended for streamed terrains.

6.6. 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 entity or a command post, 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:

$$t_{next} = t_0 + \text{min-tick-period} +/- \text{rand(min-tick-period-variance/2)}$$

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.

6.6.1. 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. You can update these parameters in the OPD Editor.

6.6.2. 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 which which you tick network interface components could delay the discovery of objects in remote front-ends and back-ends.

6.6.3. 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.

6.7. Configuring Graphics Quality

VR-Forces has command-line options that let you configure graphics quality. Depending on the capabilities of your graphics card, you may need to adjust the default settings for anti-aliasing and graphics depth and stencil buffers. The command-line options are:

- `--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.
- `--depthBits` *`depth`*. The number of bits in an image determines its quality and the number of colors it supports. You can set the depth to 24 or 32. Default: 24.
- `--stencilBits` *`bits`*. The stencil buffer can be set to 0 or 8. Default: 8.
- `--alphaBits` *`bits`*. Specifies the number of alpha bits – 0 or 8.

On older graphics cards, you may need to set anti-aliasing to 0 to improve performance.

Depending on your graphics card and driver, some combinations of these settings 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.

6.8. 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 options:

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

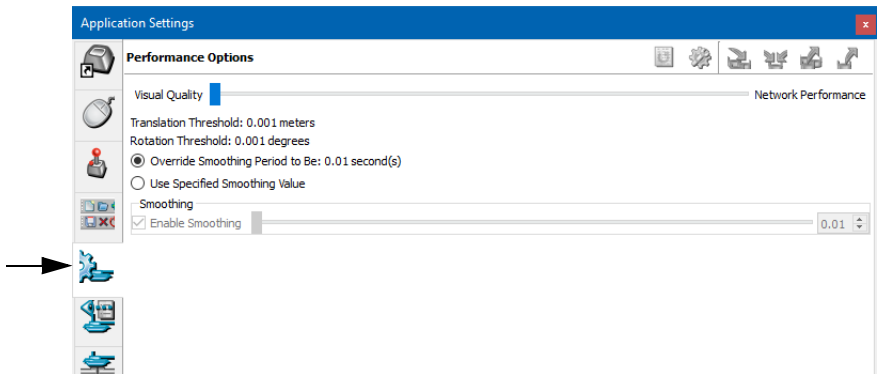


Figure 6-6. Performance Options page

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.

6.8.1. Enabling Configuration of Performance Options

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 Options page.

6.9. 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 6-7 illustrates dead-reckoning and smoothing.

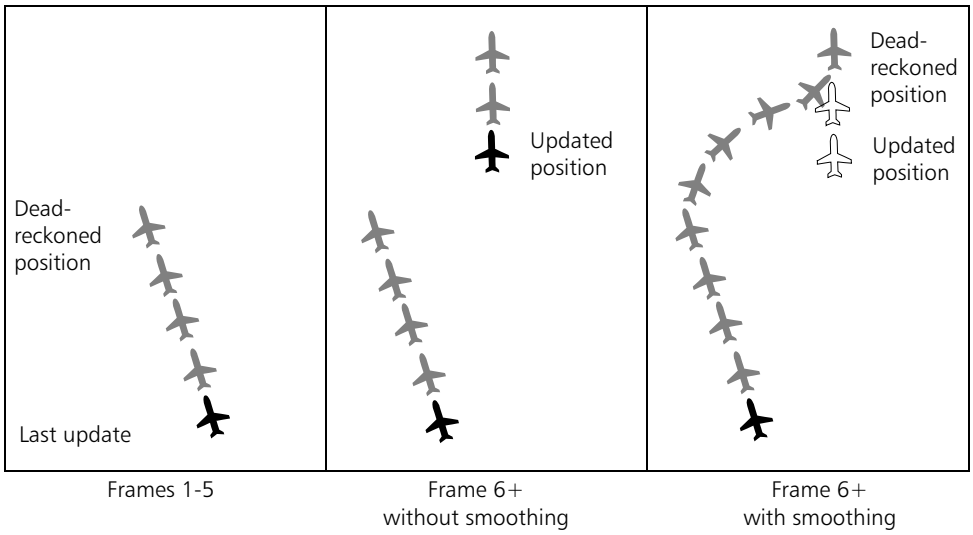


Figure 6-7. Dead-reckoning and smoothing

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.

6.9.1. Configuring Trajectory Smoothing

To configure trajectory smoothing:

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

6.10. Tuning the Target Frame Rate

The `--targetFrameRate` command line option and *vrfSim.mtl* parameter allow you to specify the mean 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 will only prevent the back-end from sleeping if it cannot achieve that rate – ultimately it is up to the operating system to determine how much CPU time each process receives.

6.11. 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 user fewer of them.

Figure 6-8 illustrates a tank with its drawables colorized.

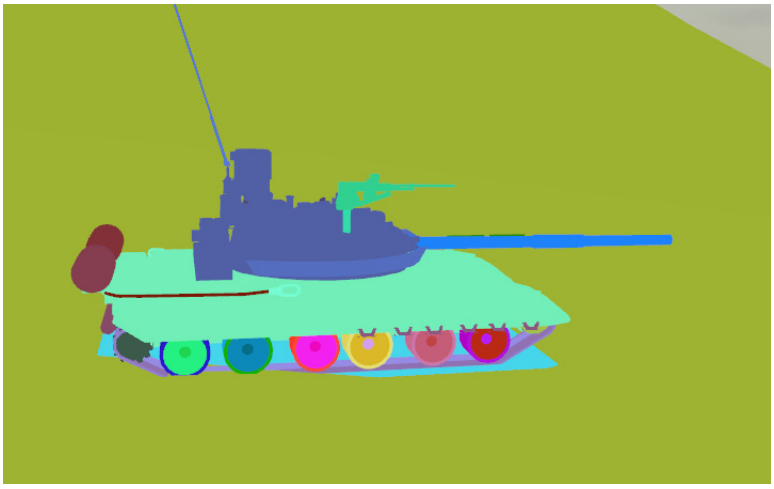


Figure 6-8. Colorized drawables on tank

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 6-4).
3. Select the Colorize Per Drawable check box.
4. To refresh the display, click Refresh.

6.12. Configuring VSync

By default, VR-Forces enables VSync (vertical synchronization), which 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. Use of VSync can also significantly improve terrain and feature data streaming performance.

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.

If you prefer not to use VSync, you can disable it at startup with the `--noVSync` command-line option, or you can configure your video card to force it to be off.



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:

- An order of battle 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 scripted tasks, 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*).

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 X, "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 (Merging) Scenarios 7-13
- Importing and Exporting MSDL 7-14
- Scenario Building Blocks 7-16

Loading a Scenario 7-17
- Loading a Recently Loaded Scenario 7-18
- Loading a Scenario from the Command Line 7-18
- Load Balancing a Scenario 7-19
- Displaying Scenario Information 7-21
- Editing the Scenario Description 7-23
- Sample Scenarios 7-23

Running a Scenario 7-24
- Changing the Simulation Speed 7-24
- Pausing a Scenario 7-25
- Rewinding a Scenario 7-25
- Closing a Scenario 7-27

Saving a Scenario 7-27
- Saving a Previously Saved Scenario 7-29
- Saving a Scenario to a New Name 7-29

Checkpoints and Snapshots 7-29
- Saving Checkpoints 7-30

Deleting Checkpoints 7-33
Creating Snapshots 7-34
Clearing the Snapshot List 7-35
Pausing a Scenario Automatically 7-35
Setting the Scenario End Time for a New Scenario 7-35
Setting the Scenario End Time for an Open Scenario 7-36
Running VR-Forces in Batch Mode 7-36
The Batch File 7-37
Creating a Batch File 7-37
Editing a Batch File 7-38
Running VR-Forces in Batch Mode 7-40
Recording Batch Scenarios 7-41
Sending Run and Pause Messages to Simulation Participants 7-42
Recording VR-Forces Simulations with the MÄK Data Logger 7-42

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 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 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 48-18. For details about terrain files, please see Section X, "Terrain".)

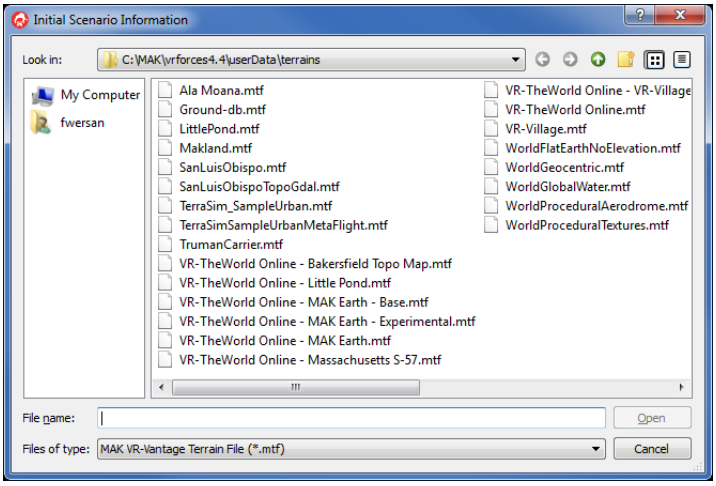


Figure 7-1. Initial Scenario Information dialog box

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).

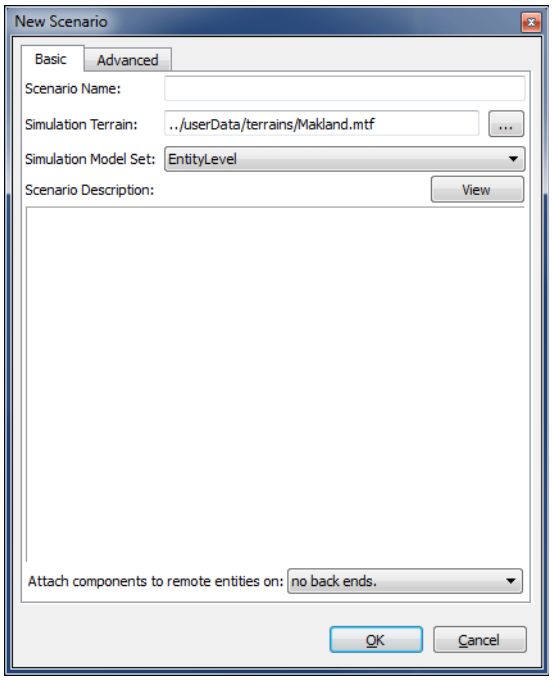


Figure 7-2. New Scenario dialog box

4. Select the 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 Information 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 "Configuring VR-Forces Components on Remote Entities," on page 70-18.

> **!** 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).

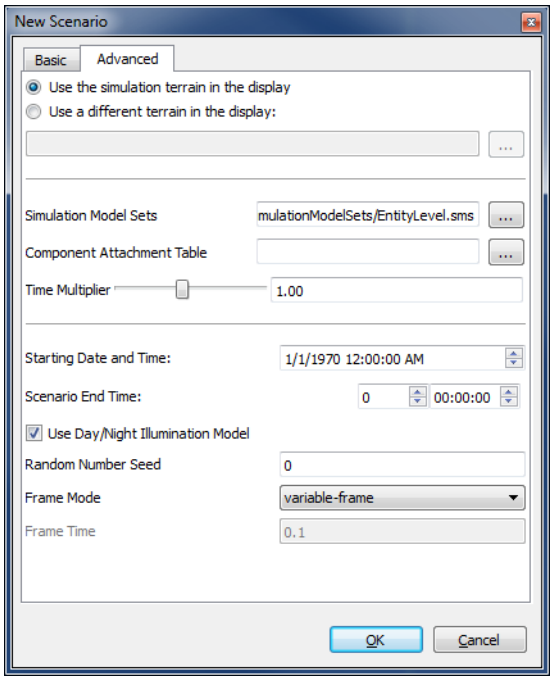


Figure 7-3. New Scenario dialog box, Advanced tab

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 "Configuring VR-Forces Components on Remote Entities," on page 70-18.
Time Multiplier	Specify 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
Starting Date and Time	Specify the date and time of day at which the simulation should start. As time advances, the Environment Settings Toolbar daylight icon changes coloration. In 3D, the sky coloration also changes as time advances. This is a back-end setting. Default: current date at 10:00 A.M.
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 7-35.
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 11-4.
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 3-11.
- "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 16, *Creating and Placing Objects*.
9. Save the scenario. For details about saving scenarios, please see "Saving a Scenario," on page 7-27.

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 63-5.)

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.

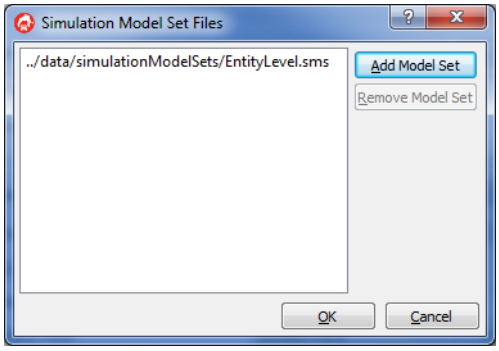


Figure 7-4. Simulation Model Set Files dialog box

2. In the Simulation Model Set Files dialog box, click Add Model Set. A blank line is added to the window (Figure 7-5).

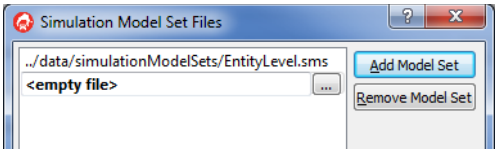


Figure 7-5. Add simulation model set

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 (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 Session Options page (Figure 7-6).

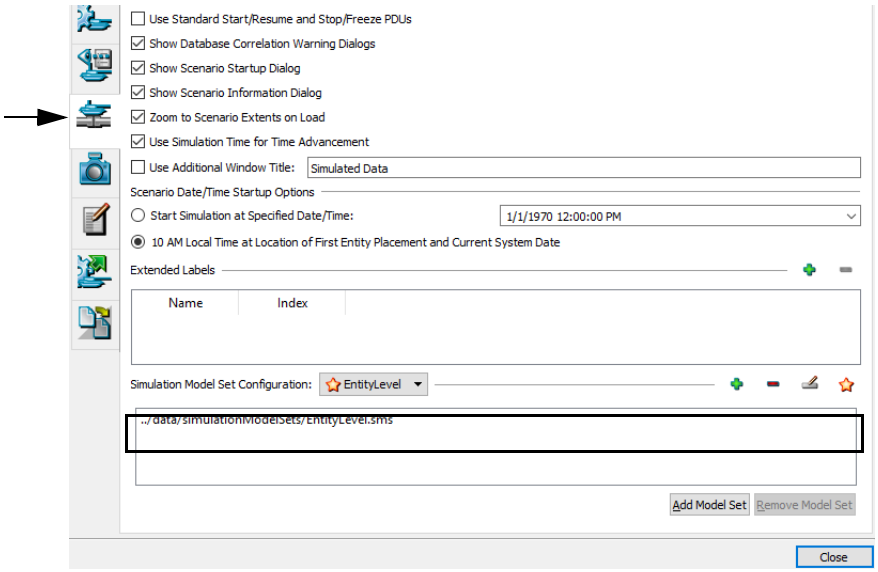


Figure 7-6. Session Options page

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 Session Options 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 entity that gets created in the scenario.



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 and you can change the default time used for new scenarios. When you set the scenario time when you create a scenario, or the default scenario start time used for new scenarios, that time is in the time zone that is currently set in the environment settings.

For more information about simulation time, please see "Setting the Date and Time of Day," on page 11-2.

To set the scenario start date and time:

1. Begin the scenario creation process, as described in "Creating a Scenario," on page 7-3.
2. When the New Scenario dialog box opens, select the Advanced tab (Figure 7-3).
3. In the Starting Date and Time box, specify the starting date and time.
4. Continue creating the scenario.

Setting the Default Scenario Starting Date and Time

You can change the default starting date and time for scenarios.

To set the default starting date and time:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Session Options page (Figure 7-6).
3. Select the Start Simulation at Specified Date/Time option.
4. Specify the starting date and time.

7.1.4. 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 7-7).

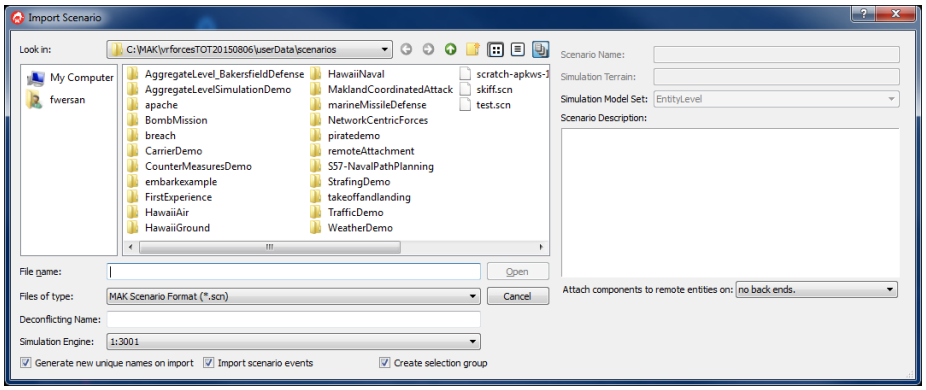


Figure 7-7. Import Scenario dialog box

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.

7.1.5. 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 Laydown**. The Import Scenario File dialog box opens.
3. Select the MSDL file that you want to import.
4. 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. Type a name for the file.
4. Click Save. If VR-Forces is not sure how to export an entity type to MSDL, the Export MSDL Scenario dialog box opens (Figure 7-8).

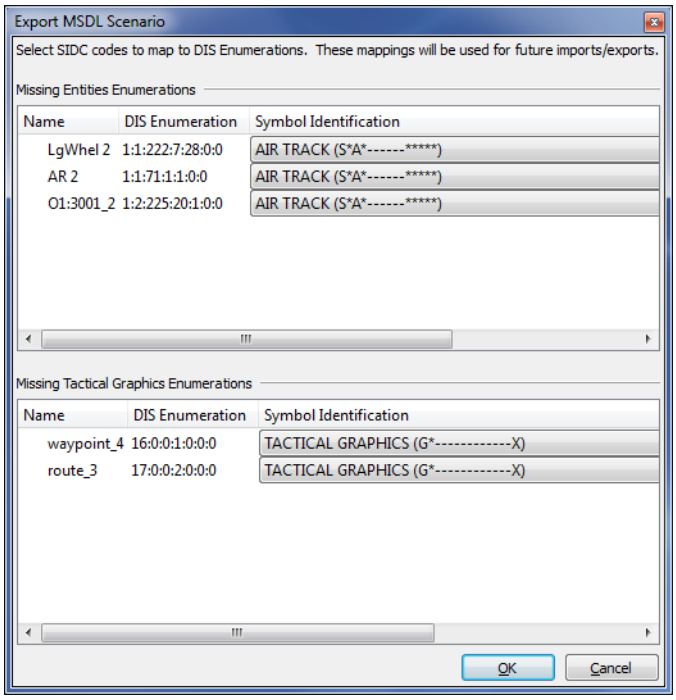


Figure 7-8. Export MSDL Scenario dialog box

5. 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.
6. Click OK.

7.1.6. Scenario Building Blocks

If you frequently need to create scenarios with the same order of battle, scenario-level scripts, or other scenario components, consider using one or more of the following strategies:

- Create new units. If the preconfigured units provided with VR-Forces do not match the units you need, create new ones in the Simulation Object Editor.
- 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 16-25.
- 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 scripted tasks that you do not want to promote to system tasks. 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 7-13.
- 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 7-14.

7.2. 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 7-9).

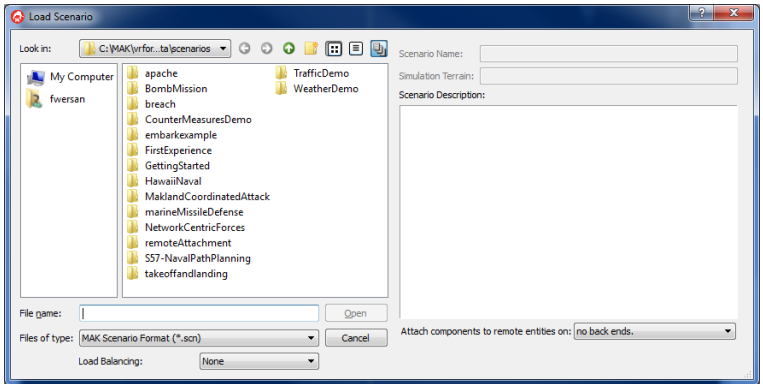


Figure 7-9. Load Scenario dialog box

2. Select a scenario to load. Basic information about the scenario is displayed (Figure 7-10). Scenarios are files with the suffix *.scn*. Sample scenarios are located in the *./userData/scenarios* directory. (Batch scenarios, described in "Running VR-Forces in Batch Mode," on page 7-36, have the suffix *.bsn*.)

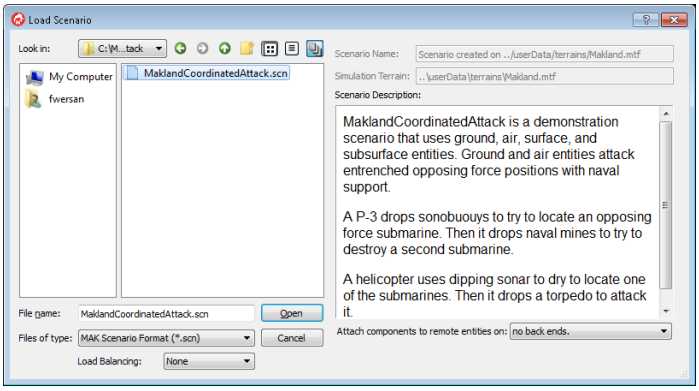


Figure 7-10. Load Scenario dialog box

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 "Configuring VR-Forces Components on Remote Entities," on page 70-18.
4. Optionally select a load balancing option.
5. Click Open.

7.2.1. 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*.

7.2.2. 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.scn"
```

To load a scenario when you start the front-end, use the `--scenarioFile` command-line option, for example:

```
vrfGui --scenarioFile "../userData/scenarios/mySce-
  nario.scn" --dis -s 1 -a 3000
```

7.2.3. Load Balancing a Scenario

After you create a scenario, you may decide that you want to spread simulation of objects across more back-ends than you used to create it. You would typically increase the number of back-ends to improve performance. The process of spreading the simulation work across multiple simulation engines is called load balancing. VR-Forces supports even distribution (done automatically by VR-Forces) and manual load balancing.

! 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 scenario:

1. Choose **File** → **Load Scenario**. The Load Scenario dialog box opens.
2. In the Load Scenario dialog box, select an option from the Load Balancing list (Figure 7-11).

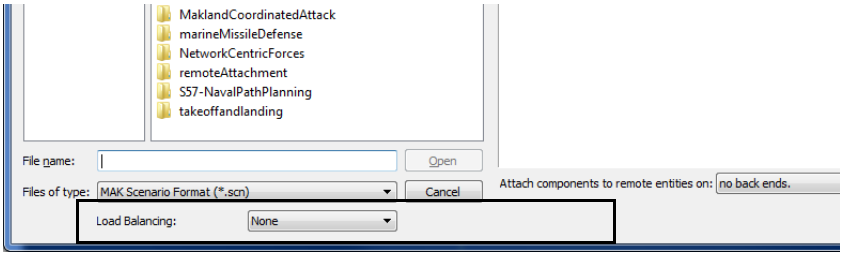


Figure 7-11. Load Balancing list

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 7-12).

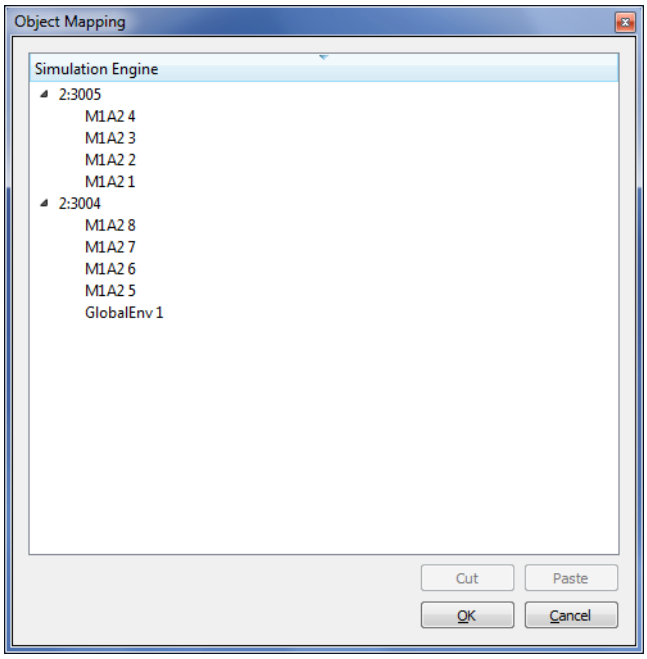


Figure 7-12. Object Mapping dialog box

3. In the Object Mapping dialog box, expand the list for the back-end from which you want to distribute objects.
4. 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.
5. 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.
6. 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.

7.2.4. Displaying Scenario Information

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. (You can disable this behavior.)

You can display scenario information at any time while a scenario is loaded. You can edit the scenario description. Other values are read only. (You can edit them directly in the scenario file. For details, please see "The Scenario File," on page 12-3.)

To display scenario information:

1. Choose **File** → **Scenario Information**. The Scenario Information dialog box opens (Figure 7-13).

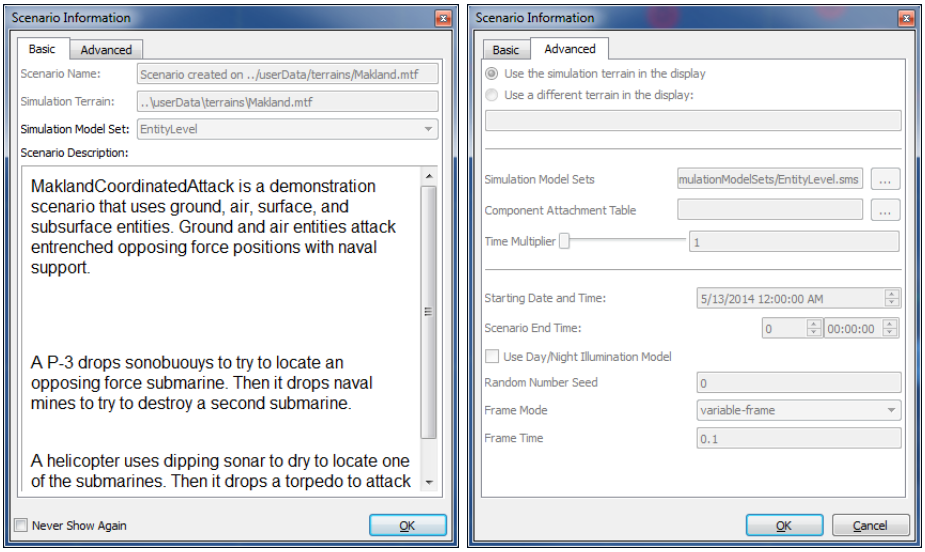


Figure 7-13. Scenario Information dialog box

2. To view advanced parameters, select the Advanced tab.

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, Session Options page, clear the Show Scenario Information Dialog check box.

➤ To enable the Scenario Information dialog box, on the Application Settings dialog box, Session Options page, select the Show Scenario Information Dialog check box.

7.2.5. Editing the Scenario Description

You can edit a scenario's description in the Scenario Information dialog box.

To edit a scenario's description:

1. Choose **File** → **Scenario Information**. The Scenario Information dialog box opens (Figure 7-14).

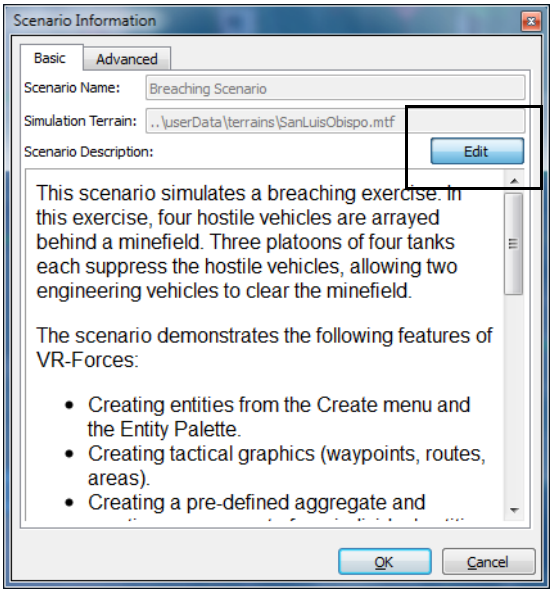


Figure 7-14. Scenario Information (editable)

2. Click Edit. The description window changes to edit mode and the button text changes to View.
3. Edit the text.
4. To see how the text will be displayed, click View.
5. Click OK.

7.2.6. Sample Scenarios

VR-Forces includes several sample scenarios. Appendix 14, *Example Entity-Level Scenarios* has the step-by-step process for creating the breaching scenario and the embarkexample scenario.

7.3. 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 7-15).

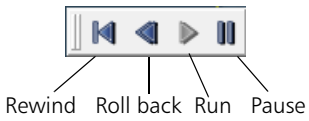


Figure 7-15. Simulation Control Toolbar

7.3.1. Changing the Simulation Speed

You can set the speed of a simulation to run faster 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, drag the indicator on the Simulation Time Scale Toolbar (Figure 7-16), or type a value in the text box.



Figure 7-16. Simulation Time Scale Toolbar

7.3.2. 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 7-15).

7.3.3. 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).



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 7-15).

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 when you clicked Run Scenario.

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, they are also deleted from the list in the Scenario Snapshots dialog box. So you cannot roll back to them. However, since they are saved to disk, you can always load them using the Load Scenario command on the File menu.

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 7-17).

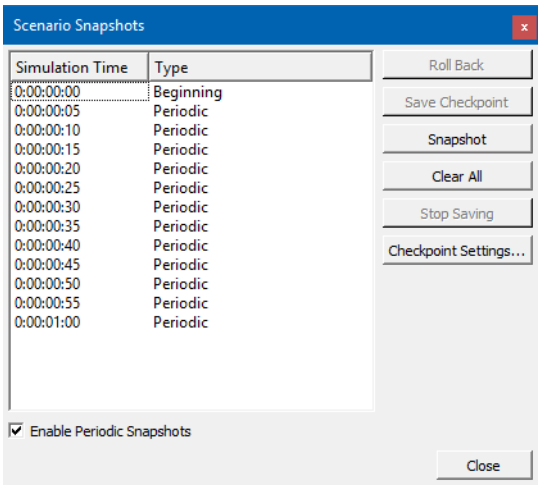


Figure 7-17. Scenario Snapshots dialog box

2. Select the snapshot or checkpoint you want to roll back to.
3. Click Roll Back.

You can also display a list of snapshots by clicking the Snapshots button (◀) on the Simulation Control Toolbar (Figure 7-18).

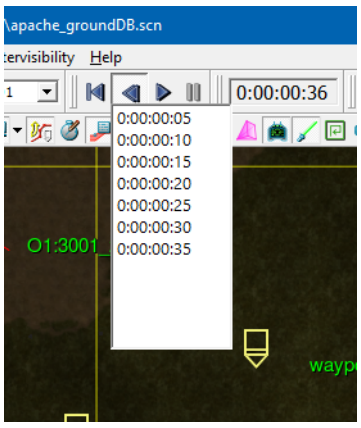


Figure 7-18. Snapshot list

7.3.4. Closing a Scenario

When you close a scenario, the terrain stays loaded.

➤ To close a scenario, choose **File** → **Close Scenario**, or click the Close Scenario button (✖) on the File Toolbar.

7.4. 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 order of battle, 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).

- 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 checkpoint, 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 Scripted Tasks," on page 29-12.

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 7-30 and "Saving a Previously Saved Scenario," on page 7-29.
- 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.



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. Click Save. All scenario files are saved with the new name.



If you save a remapped scenario, you are prompted to save the scenario with the original mappings or with the remappings.

7.4.1. 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**.

7.4.2. Saving a Scenario to a New Name

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. Click Save. The scenario is saved to the new name.

7.5. Checkpoints and Snapshots

As noted in "Saving a Scenario," on page 7-27, 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.

Sometimes VR-Forces users want to be able to quickly return to an earlier point in a simulation without the overhead of checkpoints. Snapshots save the state of a scenario to memory. Snapshots are ephemeral. They are discarded when you load a scenario. When you roll back to a snapshot, when you run the scenario, all snapshots that were taken at a later time than the rollback point get discarded and new snapshots get saved.

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.

7.5.1. 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 Scripted Tasks," on page 29-12.

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 7-19). It lists all of the snapshots available.

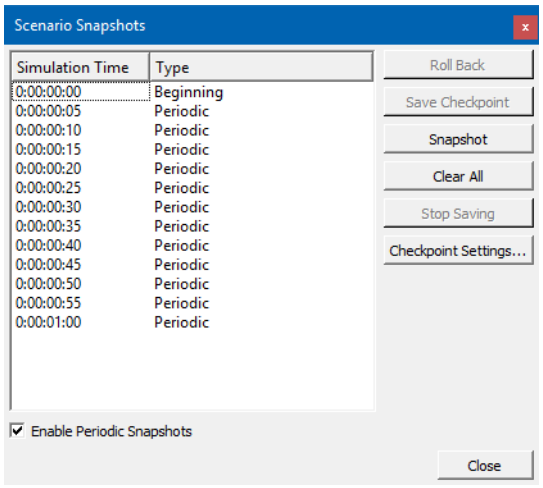


Figure 7-19. Scenario Snapshots dialog box

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 7-20).

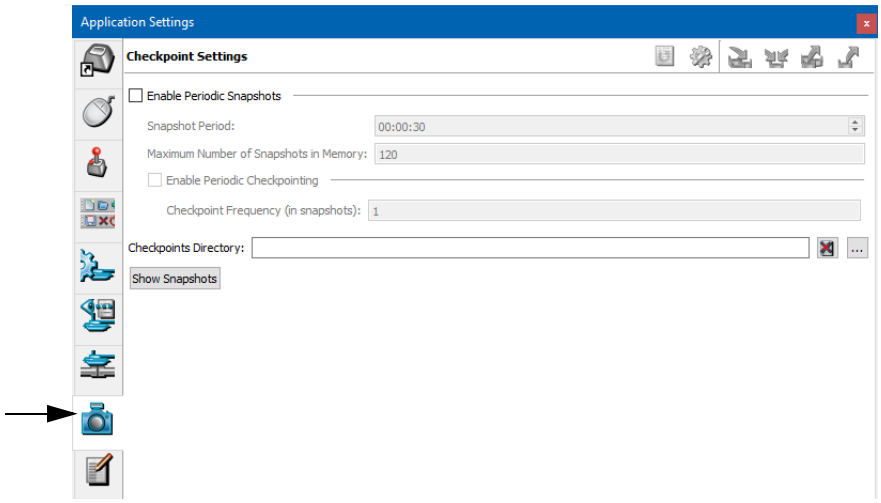


Figure 7-20. Checkpoint Settings page

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 7-20).
3. Clear the Enable Periodic Checkpointing check box.

7.5.2. 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 7-21).

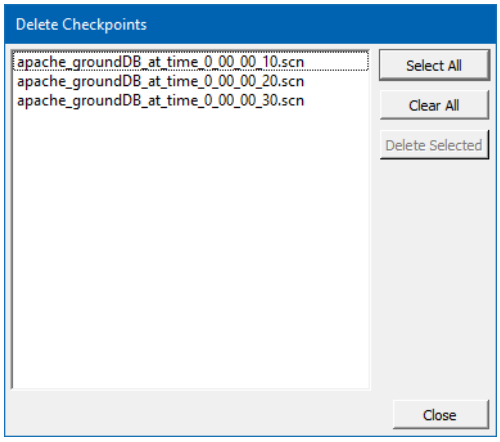


Figure 7-21. Delete Checkpoints dialog box

2. Select the checkpoints that you want to delete.
3. Click Delete Selected.

7.5.3. 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 7-19).
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 7-20).
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.

7.5.4. 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 7-19).
2. Click Clear All.

7.6. 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 7-22). When the scenario reaches this time it stops. You can resume the scenario by clicking the Play button.

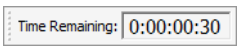


Figure 7-22. Simulation Time Remaining Toolbar

You can set the scenario end time when you create a scenario or after you have loaded it.

7.6.1. 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. Specify an end time in days, hours, minutes, and seconds.
4. Continue the scenario creation procedure.

7.6.2. 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 **Simulation** → **Set Scenario End Time**. The Set Scenario End Time dialog box opens.
2. Specify how long you want the scenario to run.
3. Click OK.
4. 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.

7.7. 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.

7.7.1. 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 MÄK 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*.

7.7.2. 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 7-38.
5. Save the new batch file.

7.7.3. Editing a Batch File

A batch file is written in MÄK Technologies Lisp format (MTL). (For information about the MTL format, please see "MÄK Technologies Lisp (MTL) Files," on page A-9.) 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\breach\breaching.scn")
      (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\apache_groundDB.scn")
      (sms-filename
        "..\data\simulationModelSets\EntityLevel.sms")
      (run-duration 0 : 0 : 0 : 20.000000) ;; real time
    )
  )
)
```

Table 7-3: Batch file parameters (Sheet 1 of 2)

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.

Table 7-3: Batch file parameters (Sheet 2 of 2)

Parameter	Description
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.

7.7.4. 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 MÄK Logger.
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.

7.7.5. 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 MÄK 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.

7.8. 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 Session Options page (Figure 7-6).
3. Select the Use Standard Start/Resume and Stop/Freeze PDUs check box.

7.9. Recording VR-Forces Simulations with the MÄK Data Logger

You can use the MÄK 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 MÄK 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
Starting a Scenario Event from a Plan 8-11
Ending a Scenario Event 8-12
Sorting the Event List 8-13
Deleting a Scenario Event 8-13
Showing and Hiding Scenario Events 8-14

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 affect simulation objects. 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).

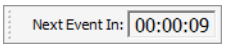


Figure 8-1. Scenario Event Toolbar

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).

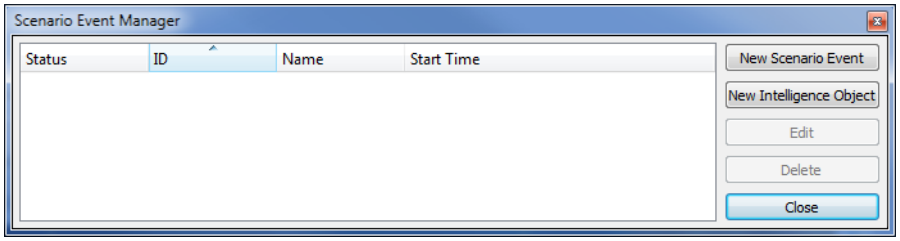


Figure 8-2. Scenario Event Manager

2. Click New Scenario Event. The Create Scenario Event dialog box opens (Figure 8-3).

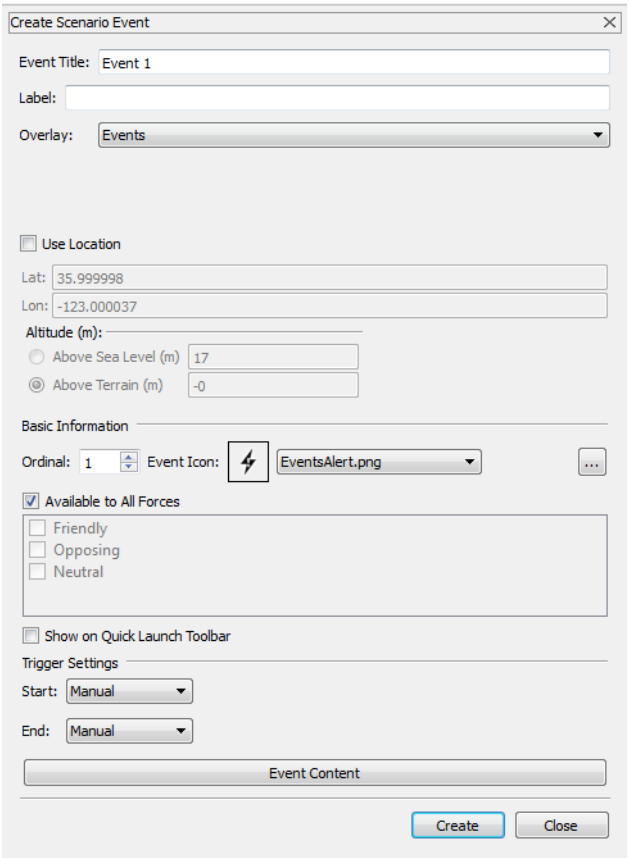


Figure 8-3. Create Scenario Event dialog box

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. ◆ Linked. Specify the linking events as described in "Adding Linked Events," on page 8-6. ◆ Simulation Time. Specify the simulation time. ◆ 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. ◆ 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. Events may be sorted 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 (in 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).

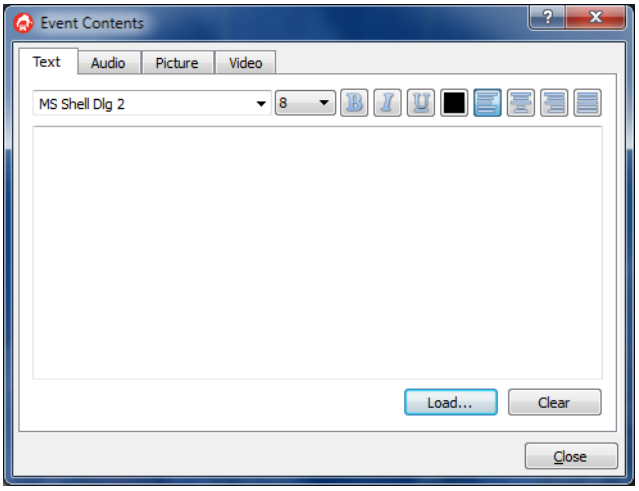


Figure 8-4. Event Contents dialog box

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).

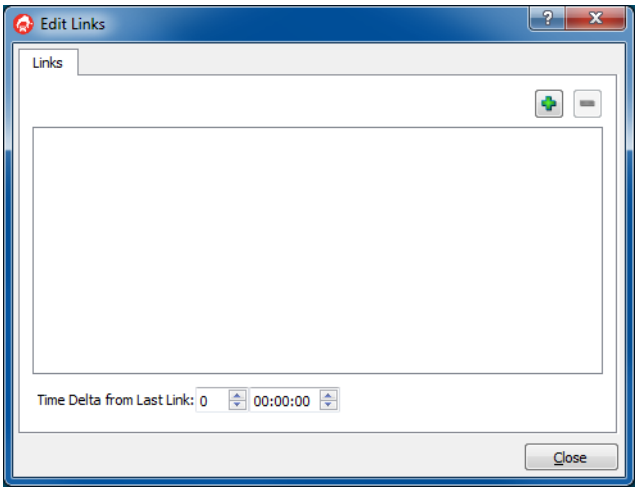


Figure 8-5. Edit Links dialog box

4. Click the Add button (+). The Link to Scenario Event dialog box opens (Figure 8-6).

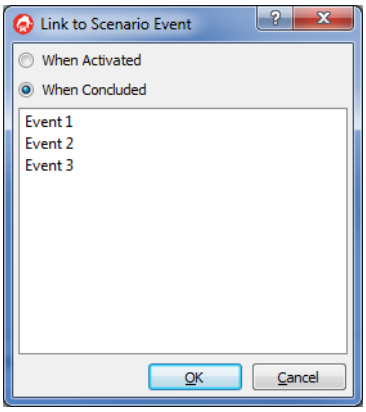


Figure 8-6. Link to Scenario Event dialog box

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.

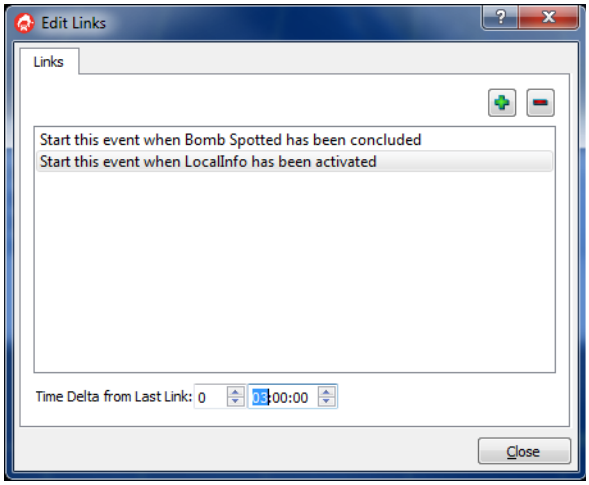


Figure 8-7. Edit Links dialog box with links

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).

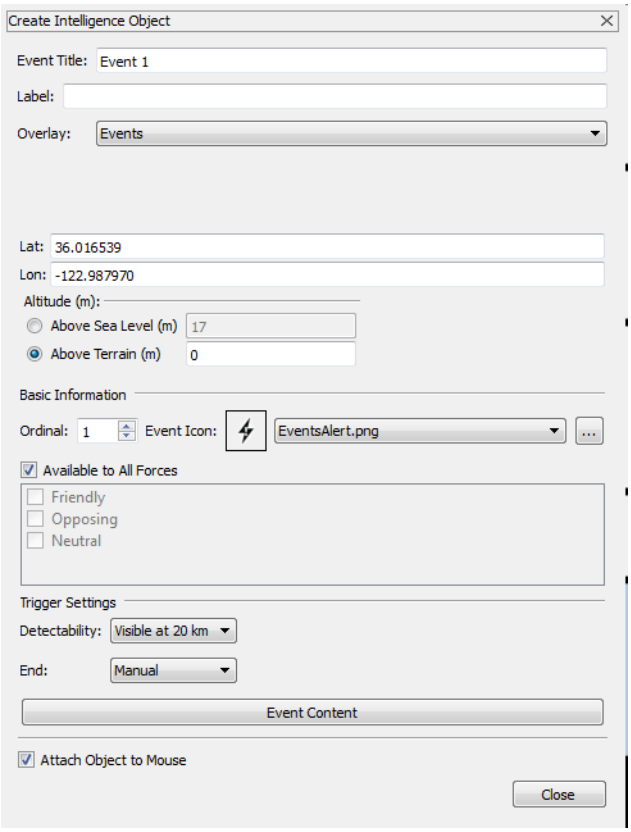


Figure 8-8. Create Intelligence Object dialog box

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 Set Sensor Signatures dialog box opens (Figure 8-9).

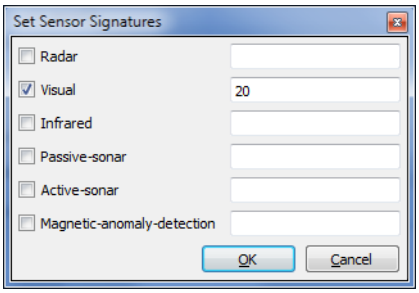


Figure 8-9. Set Sensor Signatures

   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).

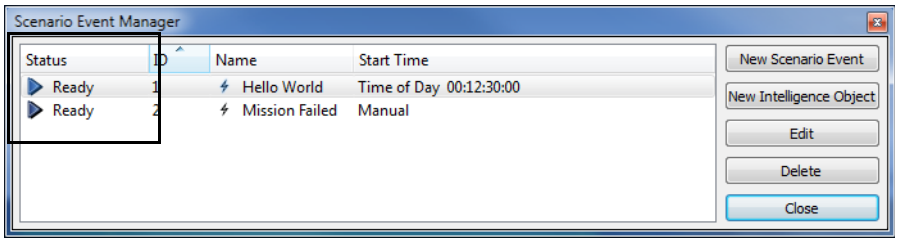


Figure 8-10. Scenario Event Manager, Play button

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).

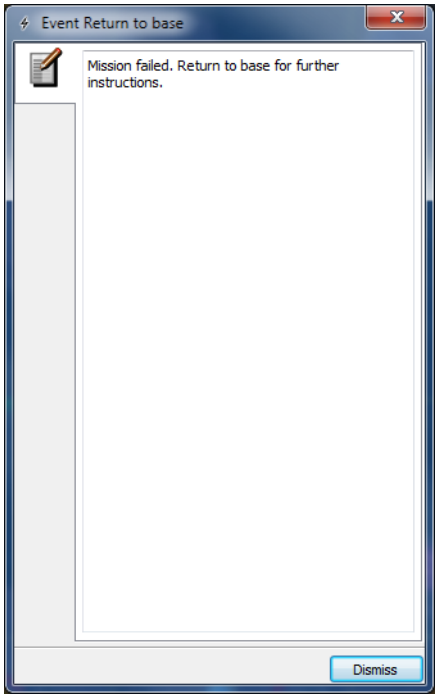


Figure 8-11. Event window

8.4.1. Starting a Scenario Event from a Plan

You can include scenario events in 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!"

To send a scenario event in a plan:

1. In a plan window, choose **Commands** → **Scenario Event**. The Scenario Event dialog box opens.
2. Select the event you want to send.
3. In the Operation list, select Start or Stop.
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 8-12).

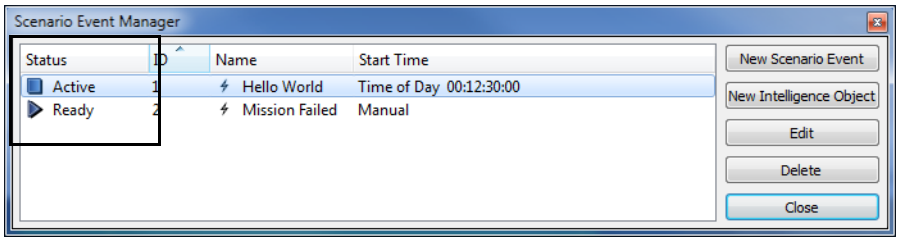


Figure 8-12. Scenario Event Manager, Stop button



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. 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.

8.7. Deleting a Scenario Event

To delete a scenario event:

1. Choose **Simulation** → **Scenario Events**. The Scenario Event Manager opens (Figure 8-12).
2. Select the event you want to delete.
3. Click Delete.

8.8. 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 8-13).

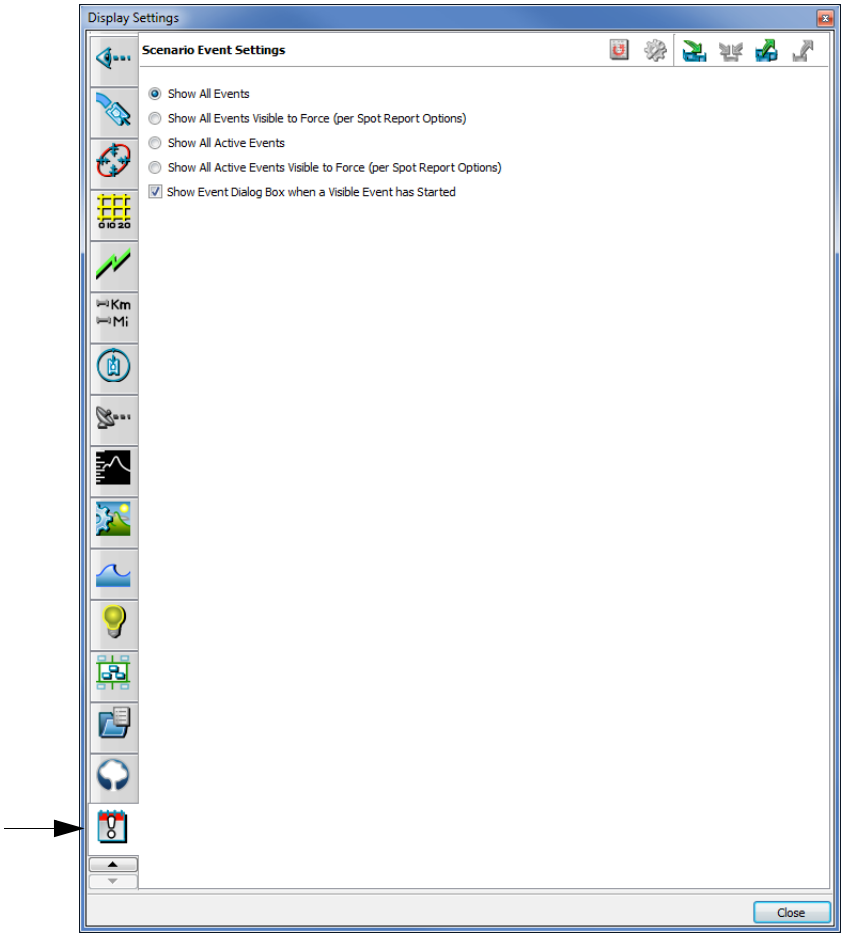


Figure 8-13. Scenario Event Settings page

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 Simulation Objects Based on Spot Reports 9-2
- Enabling or Disabling Spot Reports 9-3
- Configuring the Spot Reports Viewpoint 9-4
- Configuring the Spot Reports Certainty Level 9-8
- Applying Spot Reports to Tactical Graphics 9-9
- Displaying Labels for Spot Reports 9-9
- Using Spot Reports in Tasks 9-9

Managing Force Hostility Relationships 9-10
- Changing a Force's Hostility in a Plan 9-12

Detecting Targets 9-13
- Target Detection and Spot Reports 9-15
- Lasing Targets 9-15

Using Sonar 9-17
- Propulsion Noise and Sonar 9-18

Launching Counter Measures (Chaff and Flare) 9-19

Modeling Artillery Munitions 9-21

Taking Damage from Munitions 9-22

9.1. Displaying Simulation Objects Based on 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 simulation objects 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 simulation objects move to different locations, VR-Forces maintains a certainty level for the location of spot reports. If a detected simulation object 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 simulation object 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 simulation objects.
- When you rewind a scenario, the global spot report setting persists. Spot report settings for individual simulation objects revert to the settings saved with the scenario.

You can view a video that demonstrates simulation object classification and spot reports at *./doc/vrforces_tutorialvideos.htm*, 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 30-40.

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 Options page.
3. Select the Enable/Disable Spot Reports tab (Figure 9-1).

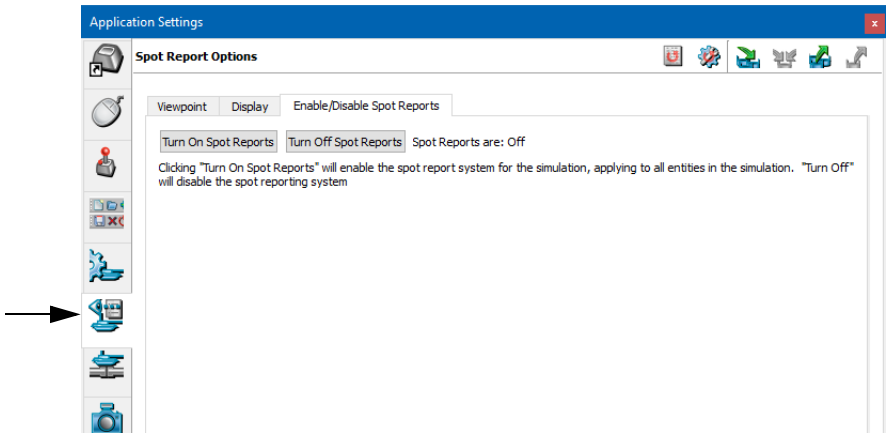


Figure 9-1. Spot Report Options page, Enable/Disable Spot Reports tab

4. Click Turn on Spot Reports or Turn Off Spot Reports. 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 Options page.
3. Select the Viewpoint tab (Figure 9-2).

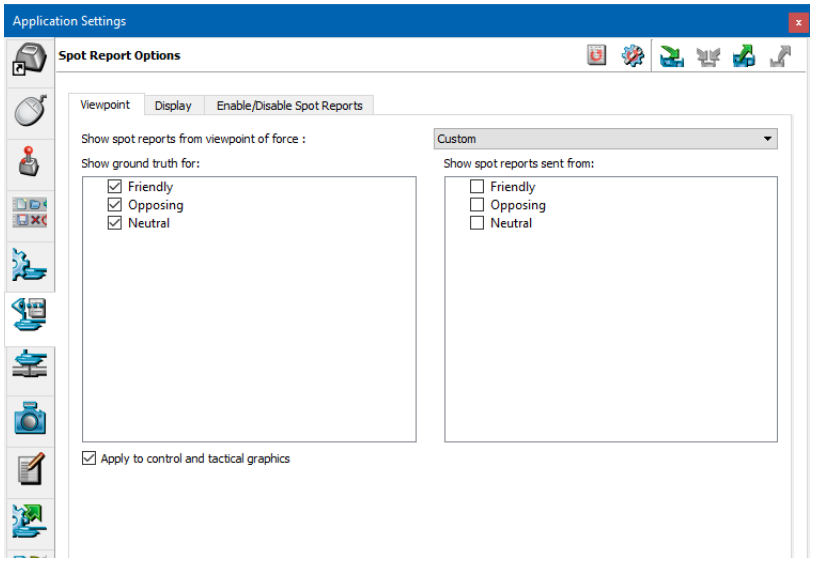


Figure 9-2. Spot Report Options page, Viewpoint tab

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 Options page.
3. Select the Viewpoint tab (Figure 9-2).
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)
    )
  )
  ...
)
```

You can edit the system definition file by hand or in the OPD Editor. For details about the OPD Editor and the Simulation Object Editor, please see Section XII, *Creating and Editing Simulation Models*.

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 Options page.
3. Select the Display tab (Figure 9-3).
4. Specify a value, in seconds, in the Time After Spot Report Expiration Before Spot Report Is Removed box.

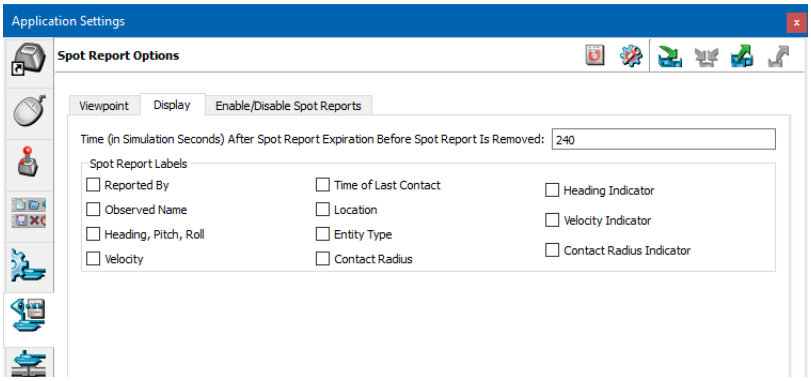


Figure 9-3. Spot Reports dialog box, Display tab

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 35-2.

To apply spot reports to tactical graphics:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Spot Report Options page.
3. Select the Viewpoint tab (Figure 9-2).
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 Options page.
3. Select the Display tab (Figure 9-3).
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.

9.2. 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 64-50. 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 9-12.

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 opens (Figure 9-4). 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).

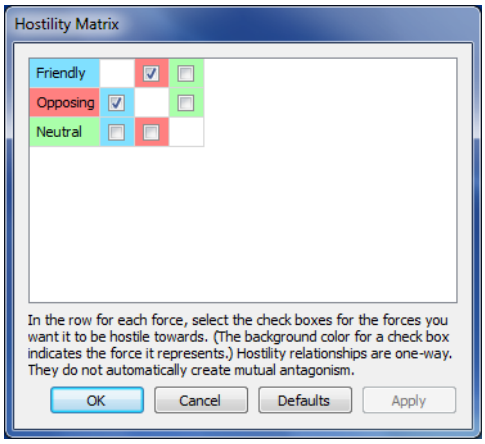


Figure 9-4. Hostility Matrix

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.

9.2.1. 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 9-5).

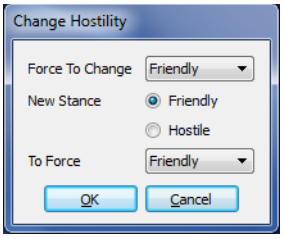


Figure 9-5. Change Hostility dialog box

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.

9.3. Detecting Targets

Simulation objects can have one or more sensors, for example, visual, sonar, or radar, for detecting and identifying other simulation objects. 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 70-2.

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 9-6 shows a series of 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 view videos that demonstrates simulation object classification and spot reports at *./doc/vrforces_tutorialvideos.htm*, or click the camera icons in previous paragraphs.

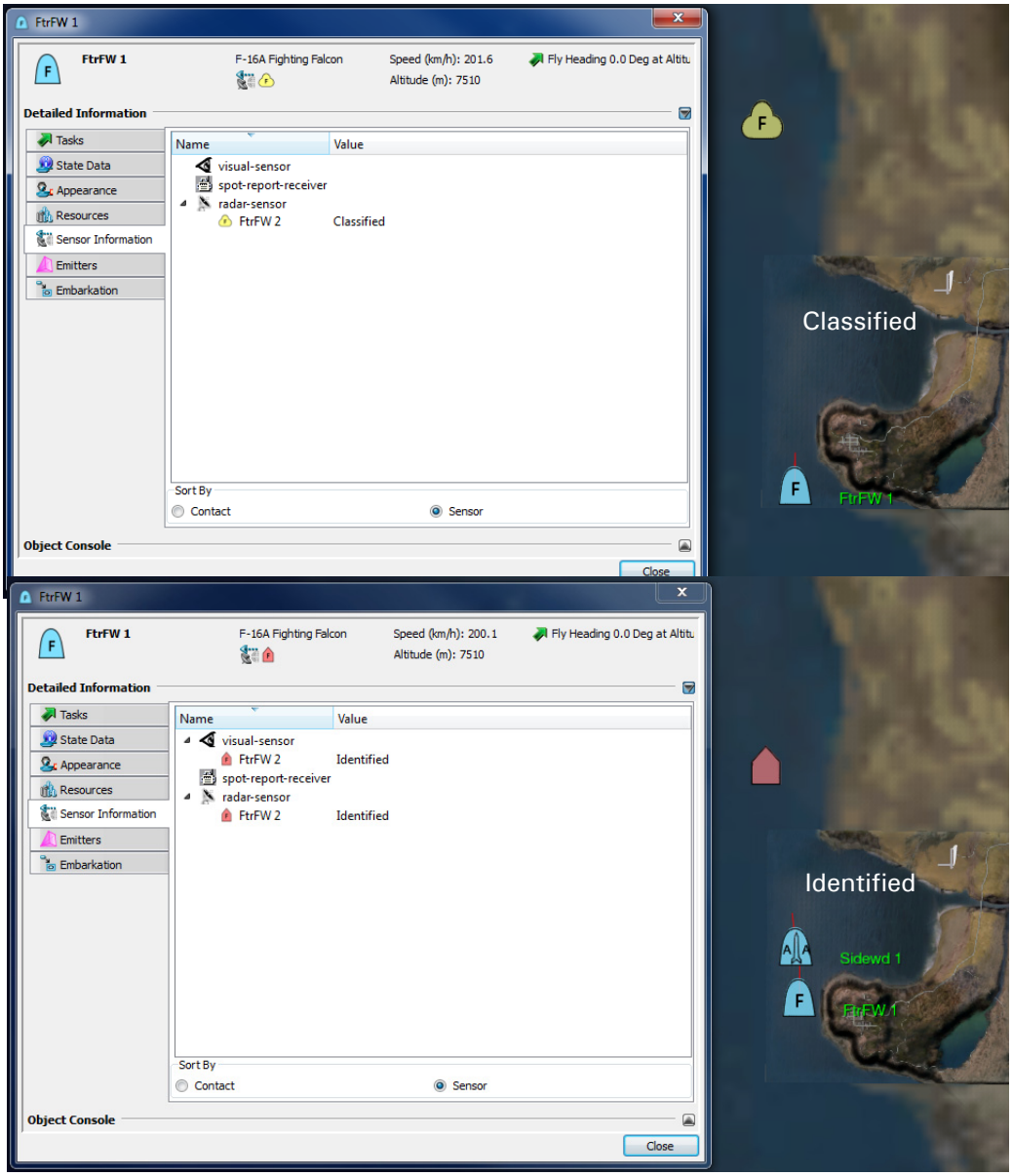


Figure 9-6. Simulation object classification and display when spot reports are enabled

9.3.1. 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.

9.3.2. 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 111-8888, using only the digits 1-8. 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 30-26 and "Synchronize Laser Code," on page 30-42.

9.4. 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 11-20.
- 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 70-11.
- 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.

9.4.1. 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 30-9) or set the **power-plant-active** parameter in the propulsion-noise-generator component of the entity's movement system in the OPD Editor.

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.

9.5. 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 *CISCounterMeasuresTable.asl* table.

Figure 9-7 shows a friendly force fighter launching counter measures against an aphid missile.



Figure 9-7. 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 26-33 and "Counter Measures Auto Launch," on page 30-12.

You can configure the behavior of automatic counter measures in the *counter-measures-launcher.sysdef* and *cis-counter-measures-launcher.sysdef* files (in *./data/simulationModel-Sets/EntityLevel/vrfSim/systems/other*). Table 9-1 describes the parameters that you might want to change.

Table 9-1: 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.

9.6. 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 and Ballistic Missiles*.

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. For example, in the OPD Editor:

1. Select the Systems List tab.
2. Select the M284 155mm Cannon.
3. Select the Components tab.
4. Expand the Actuators list.
5. Select M284-cannon.
6. Change the value for the **simulate-munition** parameter.

9.7. Taking Damage from Munitions

Entities can be damaged by munitions fired by other entities. They can be damaged by direct fire – a munition that hits the entity or within a configurable distance of the entity, and by indirect fire – a munition that explodes within a configurable distance of the entity. An entity can also suffer collateral damage. This can occur if an entity is hit by direct fire of sufficient power to damage nearby entities.

The types of munitions to which an entity is vulnerable are determined by its damage system. Each damage system has a damage actuator, which specifies damage files. These damage files specify how entities are damaged by direct and indirect fire from specific munitions. The important point to remember as a VR-Forces user is that an entity will not be harmed by a munition unless it has a damage system that knows about that munition. And, if you introduce a new munition type, it will not harm any entities unless their damage systems have a damage file for the new munition.



The default radar signature for ground vehicles is 500 meters. This is too short a distance for most fixed-wing aircraft to detect and fire on a ground vehicle. To make ground vehicles vulnerable to aircraft, increase the radar signature to a distance that is appropriate for the aircraft in your simulation.



10. Indirect Fire and Ballistic Missiles

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 10-8
- Editing Missile Target Events 10-10
- Deleting Missile Target Events 10-10

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. Simulation objects do not take damage from indirect fire unless they are configured to do so. Figure 10-1 illustrates indirect fire within an indirect fire area.

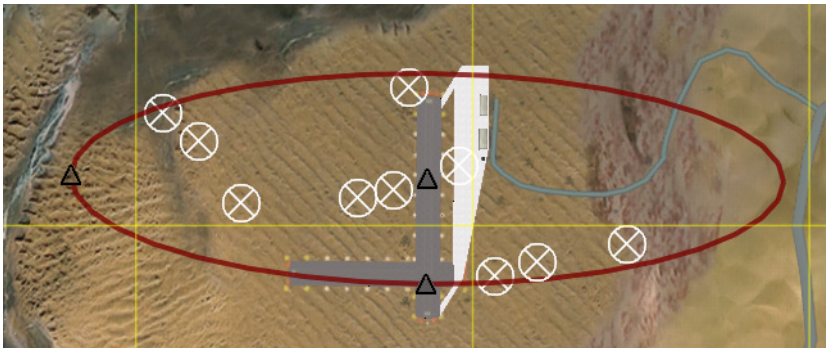


Figure 10-1. Indirect fire event

Indirect fire types are configured in *./data/simulationModelSets/EntityLevel/indirectFireTypes.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 Environmentals tab 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). You can put other objects on this layer. You can put indirect fire objects on other layers by selecting a layer in the Create Indirect Fire dialog box or by dragging them on the Overlay tab of the Objects List Panel.

To create an indirect fire event:

1. Choose **Simulation** → **Munition Targets**. The Munition Target Settings dialog box opens (Figure 10-2).

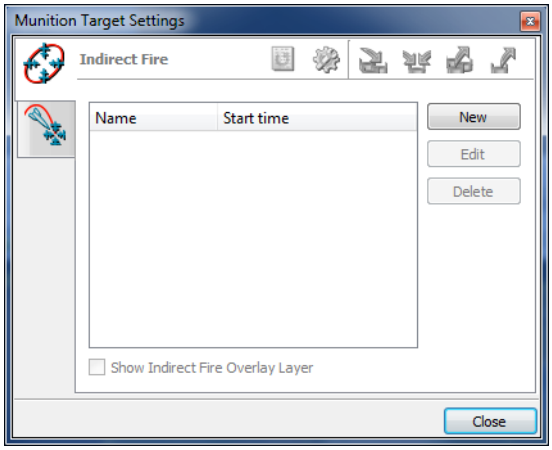


Figure 10-2. Munition Target Settings dialog box, Indirect Fire page

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.

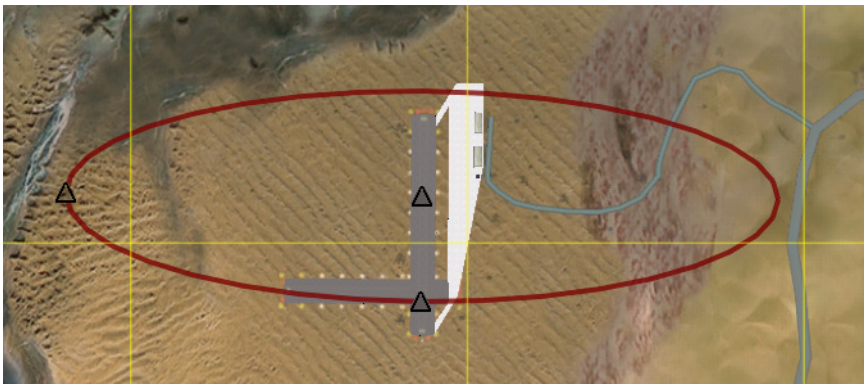


Figure 10-3. Indirect fire event area

5. Click the tab to display the Create Indirect Fire dialog box (Figure 10-4).

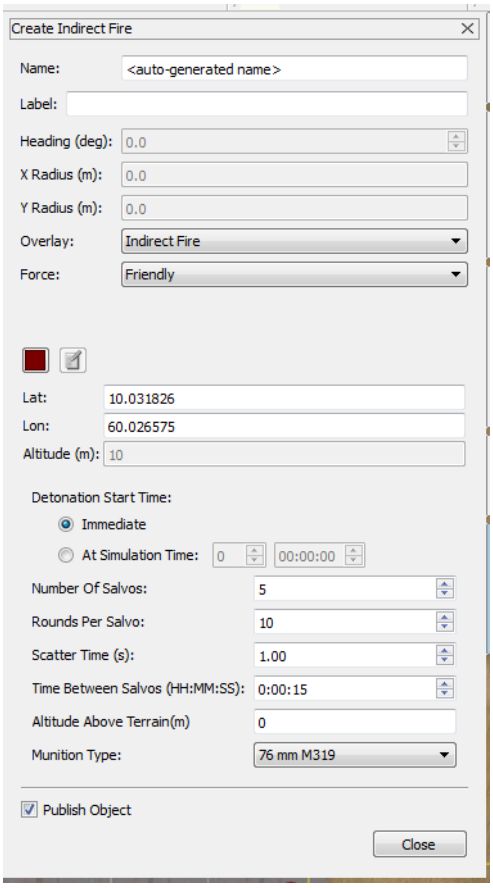


Figure 10-4. Create Indirect Fire dialog box

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 will be created 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 or At Simulation Time. If you select At Simulation Time, enter 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 tab of the Objects List Panel (unless you chose a different layer).

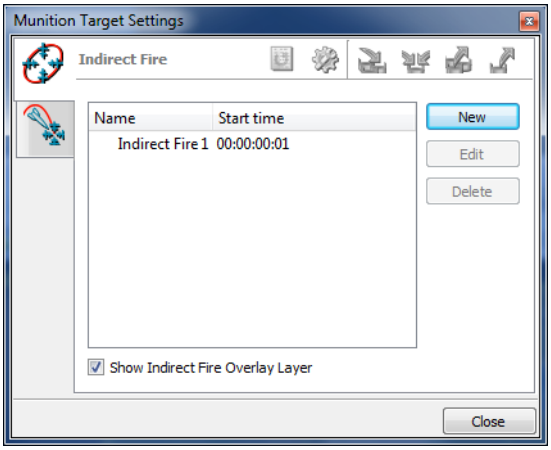


Figure 10-5. Indirect Fire page with indirect fire event

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).

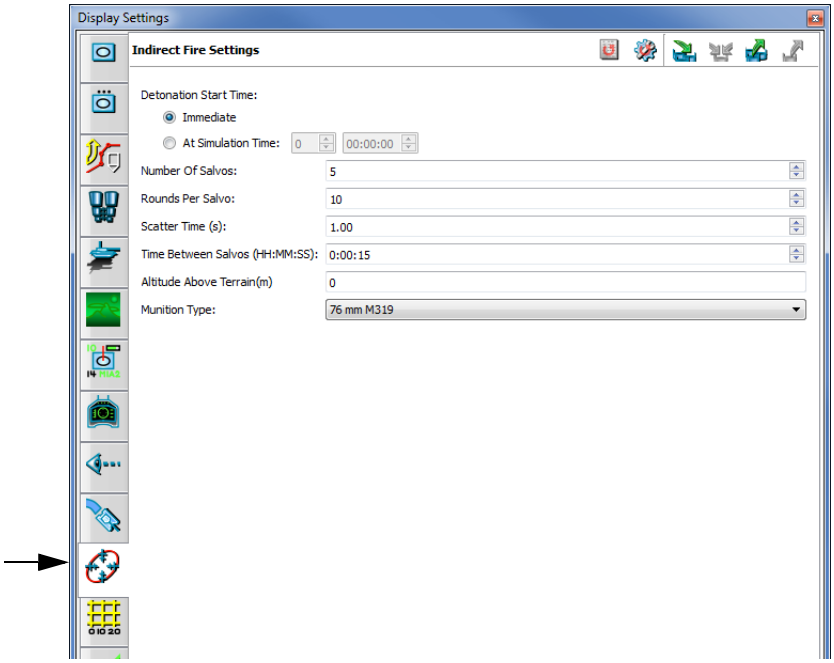


Figure 10-6. Indirect Fire Settings page

3. Set the parameters to the desired default values.
4. Click Close.

10.2. Ballistic Missiles

VR-Forces supports firing of ballistic missiles, including multistage 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.

You fire a 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 64-27.



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 (Figure 10-2).
2. Select the Missile Target page (Figure 10-7).

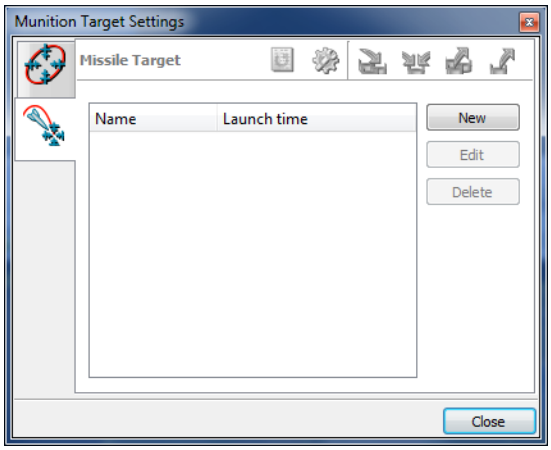


Figure 10-7. Munition Target Settings dialog box, Missile Target page

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).

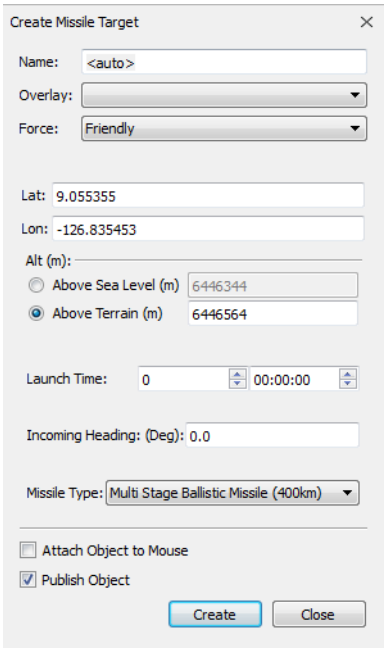


Figure 10-8. Create Missile Target dialog box

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 simply delete the target point from the terrain.



11. Setting Environment Conditions

This chapter explains how to set the time of day and weather conditions.

Introduction to Environment Conditions 11-2
Setting the Date and Time of Day 11-2
- Setting the Date 11-3
- Setting the Time of Day 11-4
- Choosing the Illumination Model 11-4

Specifying the Environment Conditions 11-5
- Adding an Environment Condition 11-7
- Editing an Environment Condition 11-7
- Deleting an Environment Condition 11-8

Setting Weather Conditions 11-8
- Setting the Wind Speed and Direction 11-9
- Setting Visibility (Fog) 11-10
- Specifying Precipitation Type and Intensity 11-11
- Specifying Cloud Cover 11-11

Creating Local Weather Zones 11-12
- Editing a Weather Zone's Environment 11-14

Configuring Marine Conditions 11-15
- Enabling Marine Effects 11-16
- Configuring Marine Conditions 11-16
- Enabling Tidal Stream Wakes 11-18
- Experimental Ocean Features 11-19

Setting the Thermocline 11-20

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 Settings**, or click the date on the Environment Settings Toolbar. The Environment Settings dialog box opens.
2. Select the Date and Time page (Figure 11-1).

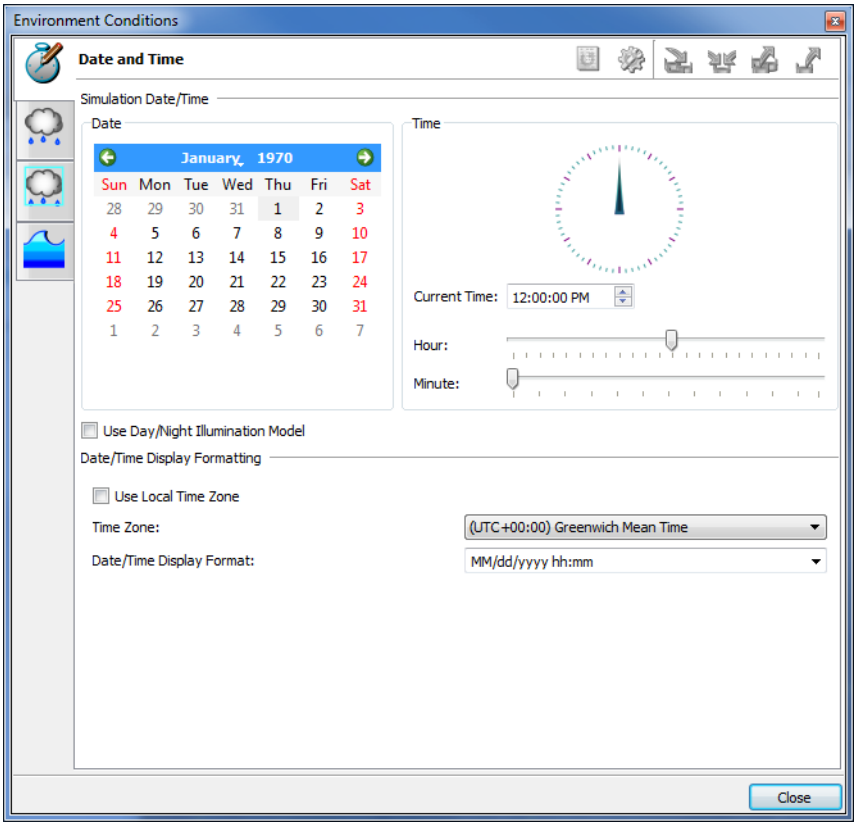


Figure 11-1. Date and Time page

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 Settings**, or click the date on the Environment Settings Toolbar. The Environment Settings 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, 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 Settings**, or click the date on the Environment Settings Toolbar. The Environment Settings dialog box opens.
2. Select the Date and Time page (Figure 11-1).
3. Select or clear the Use Day/Night Illumination Model check box.

11.3. Specifying the Environment Conditions

VR-Forces has a set of preconfigured environment conditions. Each environment condition has settings for:

- Weather:
  - 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 4-23.

You can add your own 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).

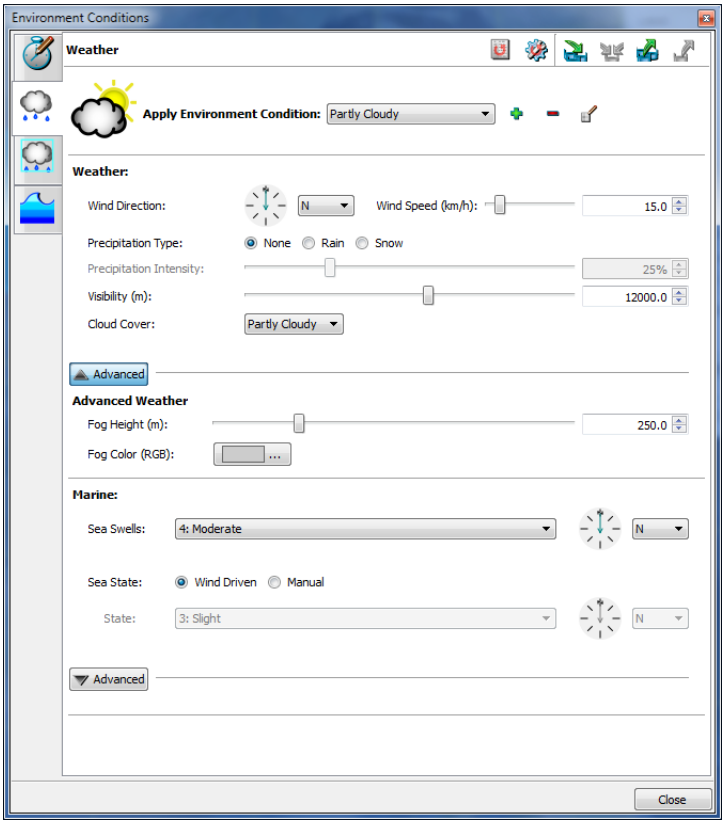


Figure 11-2. Weather page

3. In the Apply Environment Condition list, select the condition you want.

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.

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).

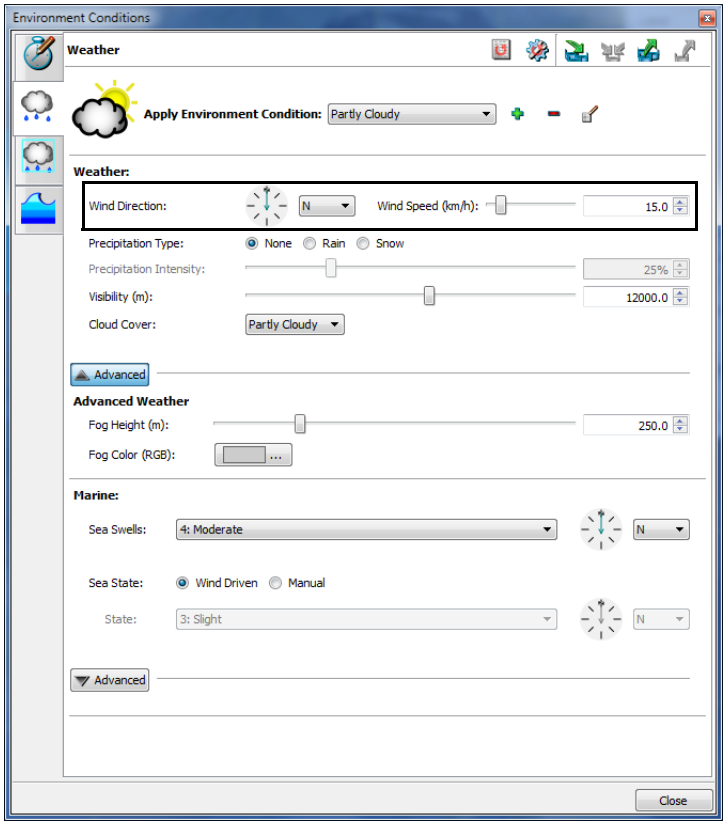


Figure 11-3. Weather page

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 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.

11.4.3. 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%.

11.4.4. 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 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.

Weather zones are rectangular areas on the terrain. By default, they 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.

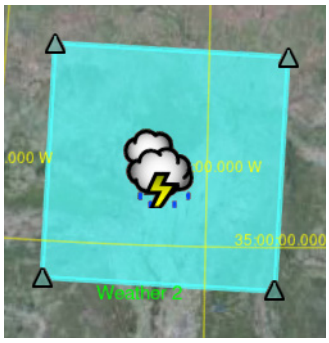


Figure 11-4. Weather zone

To create a local weather zone:

1. Choose **Simulation** → **Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Local Weather Zones page (Figure 11-5).

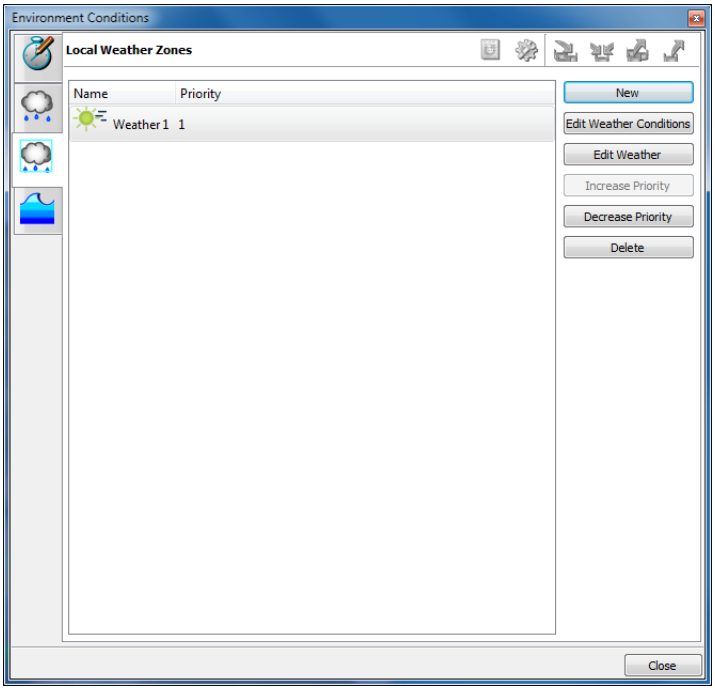


Figure 11-5. Local Weather Zones page

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 11-14.
5. Click Create.



You can also create a weather zone by selecting Weather on the Hazards/Obstacles Palette.

11.5.1. 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. For details about preconfigured environmental conditions, please see "Specifying the Environment Conditions," on page 11-5.

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.

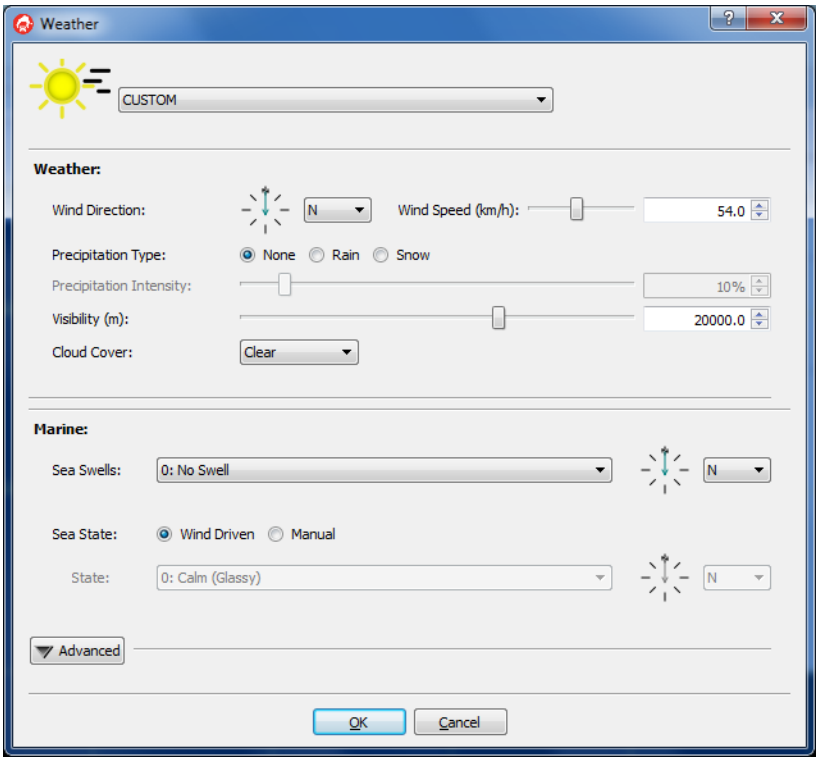


Figure 11-6. Set Weather dialog box

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 11-8.
3. Click OK.

11.6. Configuring Marine Conditions

VR-Forces supports a variety of dynamic ocean effects (Figure 11-7), 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 11-7. Fishing boat

Enabling dynamic ocean effects can affect performance.

11.6.1. Enabling Marine Effects

Marine effects can reduce performance. Therefore, they are disabled by default.

➤ To enable marine effects, choose **Observer** → **Dynamic Ocean**.

11.6.2. 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 11-3).

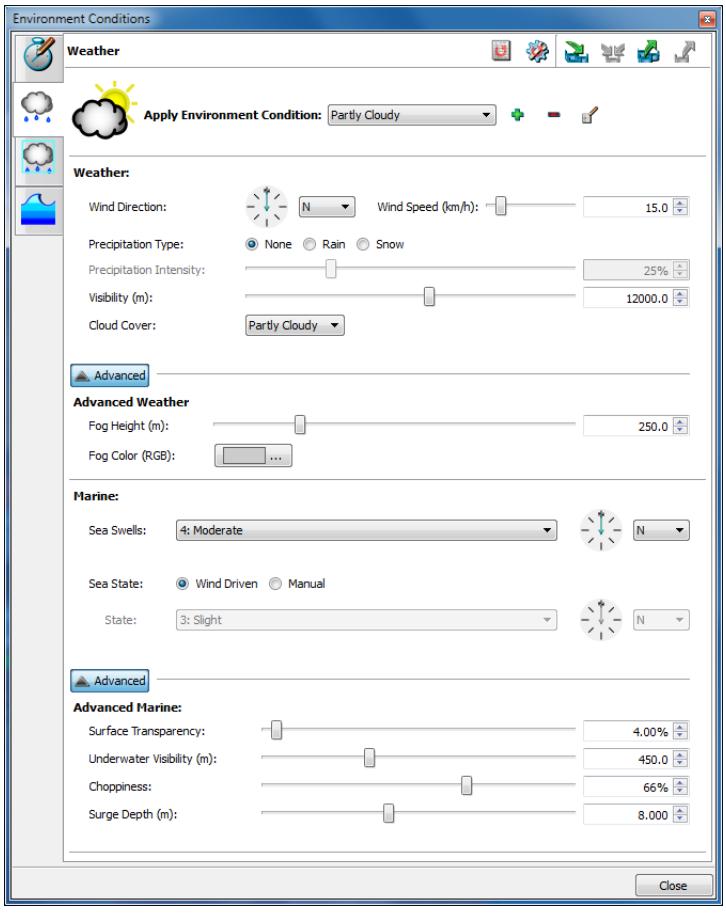


Figure 11-8. Weather page

3. Change settings as desired. The settings are described in Table 11-1.

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 Douglass 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.

11.6.3. 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 11-16.

To enable or disable tidal stream wakes:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Render Settings page (Figure 11-9).

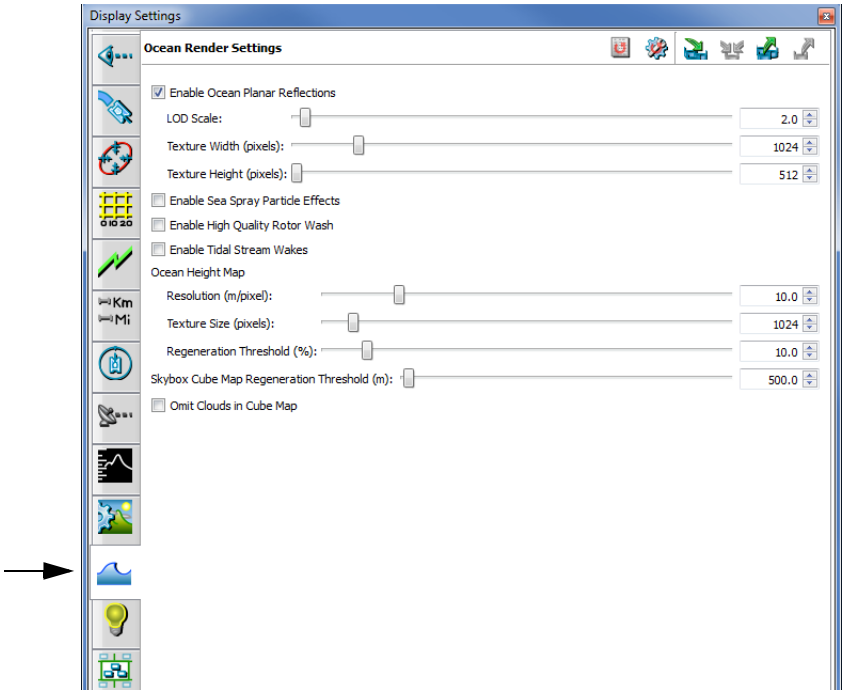


Figure 11-9. Ocean Render Settings

3. Select or clear Enable Tidal Stream Wakes.

11.6.4. Experimental Ocean Features

VR-Forces has some marine features that are considered experimental. In some cases they improve the scene. In others, they may not be realistic or may degrade performance.

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 *VR-TheWorld Online - MAK Earth.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.

To configure breaking waves:

1. Choose **Settings** → **Experimental Features**. The Experimental Features dialog box opens (Figure 11-10).

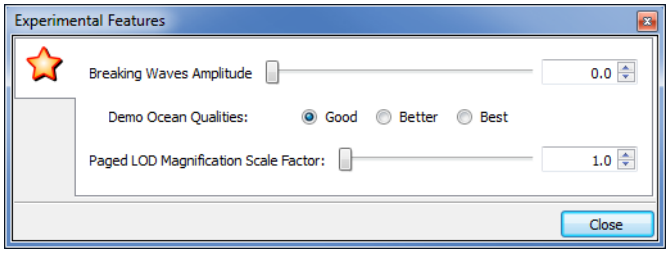


Figure 11-10. Experimental Ocean Utilities dialog box

2. Adjust the Breaking Waves Amplitude slider or type a value in the box. The maximum value is 10.0.

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** → **Experimental Features**. The Experimental Features dialog box opens (Figure 11-10).
2. Select one of the Demo Ocean Qualities options.

11.7. 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 11-11).

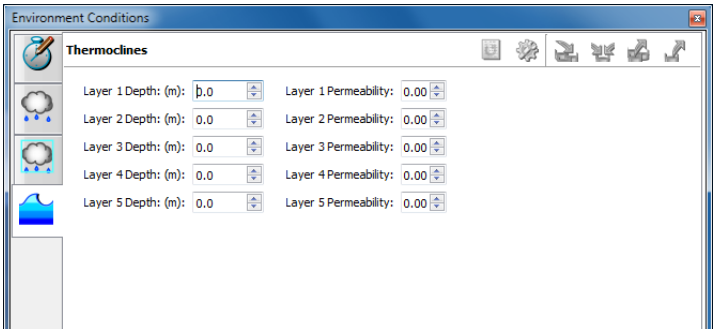


Figure 11-11. Thermoclines page

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 12-8
The Object Map File 12-9
Editing the Order of Battle File 12-10
Saving Default Parameters to the Order of Battle 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 order of battle 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 scripted tasks.
- The overlay file (*scenario_name.ovl*) lists tactical graphics that are not published. Published tactical graphics are in the order of battle 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.

For the most part, you will not want to edit these 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). Thereafter, you cannot edit scenario parameters in the front-end except for the scenario description and the option to attach components to remote simulation objects. However, you can edit them in a text editor.

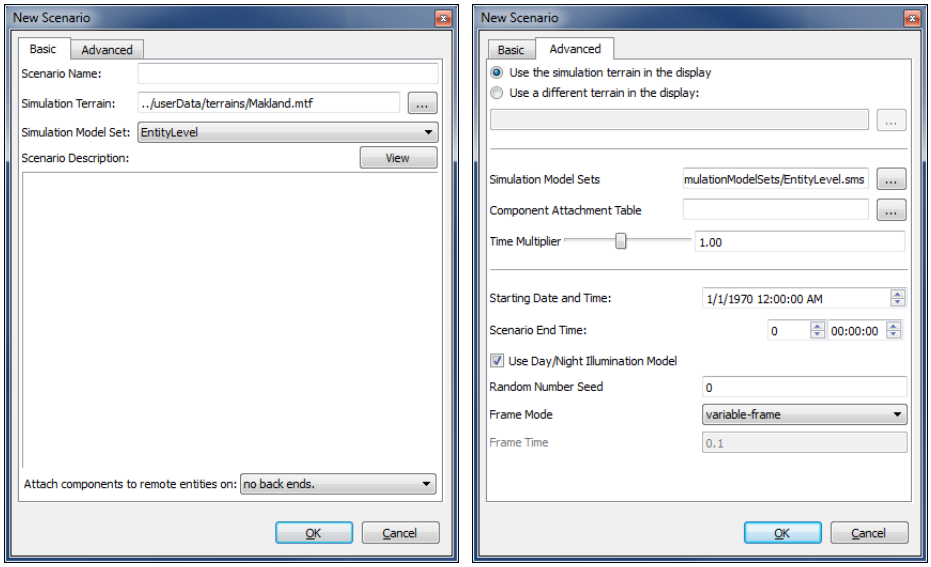


Figure 12-1. New Scenario dialog box

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")
  (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)
  (sim-time  0.000000)
  (exercise-start-time  0.000000)
  (additional-opds )
  (Simulation-Model-Set-Files
 "../data/simulationModelSets/EntityLevel.sms")
  (scenario-name  "myscenario"
  (scenario-data
    (GuiObserverViews  "myscenario.osrx")
    (GuiScenarioSettings  "myscenario.gui_settings")
    (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  3.100000)
)
```



Paths in the scenario files are relative to the application directory, for example, *./bin64*.

The Starting Time of Day is stored in the order of battle file, not the scenario file.

The parameters in a scenario file are as follows:

- **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 order of battle file.
- **Scenario-Scripts** specifies the file that has scenario-specific Lua scripts.
- **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 Managment, 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.
- **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 3-11.
- **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.
- **additional-opds** specifies custom OPD files to load (for backwards compatibility).
- **Simulation-Model-Set-Files** specifies a list of SMS files to load.
- **scenario-name** is 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.
- **scenario-data** is the name of a block of data with the following additional parameters:
  - **GuiObserverViews** specifies the saved views file for the scenario.
  - **GuiScenarioSettings** specifies the *.gui_settings* file. It records scenario-specific GUI settings.
  - **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.



The **sim-time** and **exercise-start-time** parameters are in the scenario file for backwards compatibility. They are not supported in scenarios created after VR-Forces 4.0.3.

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 directory specified by **defaultTerrainDatabasePath**.
4. If defined, look in the user-defined environmental variable VRF_TERRAIN_-PATH.

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.

Once you have chosen the terrain database, you cannot change it from the graphical user interface. You can edit the pathname of a terrain database in the scenario file. However, unless the new terrain database covers exactly the same geographic area as the old one, the order of battle and overlay information associated with the scenario may be confusing or useless.

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 a Scenario," on page 7-19.

12.5. Editing the Order of Battle File

The order of battle (OOB) is a description of the local objects in a simulation. 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 order of battle 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 order of battle 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 order of battle file.

For more information, please see Section VII, "Tactical Graphics," and Chapter 34, *Overlays*.

12.5.1. Saving Default Parameters to the Order of Battle File

When a simulation object in a scenario is saved to the OOB 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 OOB 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 order of battle files.

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.



13. Using a Communications Effects Server

This chapter explains how to configure VR-Forces to interoperate with an external communications effects server.

Introduction 13-2
Configuring VR-Forces 13-2
Enabling the External Communication Model 13-2
Configuring the External Communication Model 13-3
Using VR-Forces with QualNet 13-4
Running VR-Forces with QualNet 13-4
Example Scenario 13-5
Running the Scenario 13-5

13.1. Introduction

You can configure VR-Forces to use an external communications effects server to control the transmission of radio messages between VR-Forces simulation objects. The communications effects server determines which simulation objects are able to communicate with each other, and how long it takes a message to be transmitted from the originator of the message to the receiver of the message.

VR-Forces supports the interface specified in the file *./doc/QualNet_HLA_interface_RPR-FOM 1.0_extensions_ICD3.pdf*. This Interface Control Document (ICD) was created by Scalable Network Technologies and is distributed with their permission. Any communications effects server that conforms to this interface can be used with VR-Forces.

13.2. Configuring VR-Forces

By default, VR-Forces is configured to use a simple internal communications effects model that allows all messages to be passed to their recipients. To use an external communications effects server, you must:

- Enable the publishing of radio transmitters for any entity types that will be sending messages though the communications effects server. For details, please see "Publishing Radio Transmitters," on page 70-13.
- Set the external communication model as the communication model to be used by VR-Forces.
- Configure the external communication model.

13.2.1. Enabling the External Communication Model

You can enable external communication in the front-end or by editing *./data/simulationModelSets/<model_set>/commModelParams.mtl*. When you enable the external communication model in the front-end, VR-Forces also enables publishing of transmitters on all simulation objects that have radios. When you change the communication model setting, it gets saved. The next time you run VR-Forces, the saved setting is used.

When you change the communication model setting, existing transmitters are destroyed and new transmitters are created. Therefore, changing this option while running a scenario with QualNet will cause QualNet to no longer recognize the transmitters being published.

➤ To enable or disable the external communication model, choose **Settings** → **Use External Communication Model**.

Enabling the External Communication Model in *commModelParams.mtl*

To enable the External Communication Model, edit *./data/simulationModel-Sets/<model_set>/commModelParams.mtl*. This file defines the communication models available to the scenarios. By default, VR-Forces is configured to use the **simple-radio-comm-model**, which allows all communication between simulation objects to be received.

➤ To use an external server for all radio communication, comment out the **simple-radio-comm-model** block, and uncomment the **external-comm-model** block.

This causes the external server to be used for all radio communication.

13.2.2. Configuring the External Communication Model

If you enable the external communication model, you can set the following configuration options in *commModelParams.mtl*:

- **use-actual-message-size** specifies how the size of the messages will be modeled. When a message is sent using the external communication model, a message size is sent to the external communications effects server, and that number may affect how long it takes for the message to reach its recipient. If this parameter is set to True (the default), the actual number of bytes of the VR-Forces message is used for this size value. This includes the VR-Forces message header of the sender or recipient and the contents of the message. If this parameter is set to False, the size specified by **substitute-message-size** is used for all messages.
- **substitute-message-size** specifies the message size, in bytes, to send to the external communications effects server. It is used only if **use-actual-message-size** is False.
- **message-timeout** specifies the timeout value, in seconds, for messages sent to the communications effects server. Messages that are not delivered to the server by the end of the timeout period are dropped.

13.3. Using VR-Forces with QualNet

To use VR-Forces with QualNet, it must be configured to use the external communications effects server, as described in "Configuring VR-Forces," on page 13-2. Please note the following requirements:

- QualNet only supports particular network configurations. Please refer to the QualNet documentation for details.
- If you are using the MÄK RTI, you must run the rtiexec.
- You must create a QualNet scenario that matches the VR-Forces scenario to be run using QualNet as the communications effects server. You can do this by hand, or using the QualNet HLA Extractor tool. Please see the documentation provided with the HLA Extractor for instructions on its use.



A version of QualNet is available that is specially configured to work well with VR-Forces. Please contact your QualNet or MÄK salesperson for details about this version of QualNet. Other versions of QualNet are also compatible with VR-Forces, but require additional configuration to work with it. Please see QualNet documentation and or contact QualNet support to configure it to work with VR-Forces.

13.3.1. Running VR-Forces with QualNet

To use VR-Forces with QualNet, the two programs must be loaded and scenarios started in the following sequence:

1. If you are using the MÄK RTI, start the rtiexec. If you are using a different RTI, run its equivalent server application.
2. Start QualNet and load the desired QualNet scenario. (Right click the *.config* file, and choose Run.)
3. Start VR-Forces, and load the VR-Forces scenario.
4. After all simulation objects have been displayed in the VR-Forces GUI, click the Play button on the QualNet interface.
5. Run the VR-Forces scenario.

- If you want to load a new VR-Forces scenario, you must stop the QualNet scenario and load a new QualNet scenario using the procedure described in this section.
- The VR-Forces rewind functionality does not work correctly with QualNet. To replay a scenario you must restart QualNet and VR-Forces as described in the procedure.

13.4. Example Scenario

VR-Forces includes an example VR-Forces scenario and a matching QualNet scenario to demonstrate the interoperability of VR-Forces and QualNet.

In the scenario, the M1A2 1 tank commands the UAV to fly over the area where the opposing forces are and send back spot reports. When the first tank receives spot reports, it commands the second one to move to the waypoint, and also moves there itself. All commands and spot reports are sent though the radio, which is passed though QualNet. View the plans for M1A2 1 and UAV 1 to see the details of how they are communicating.

If none of the simulation objects start moving after 10 seconds of running, there is probably an error communicating with QualNet.

13.4.1. Running the Scenario

To run the scenario, use the following steps:

1. Start your RTI. If you are using the MÄK RTI, run the rtiexec.
2. Run QualNet.
3. In the QualNet window, select the Scenario Designer tab.
4. Expand the file system tree for the *scenarios* folder.
5. Expand the tree for the *vrforces-CommsDemo* scenario.
6. Right-click *vrforces-CommsDemo.config*. A menu is displayed.
7. Choose **Run Scenario**. QualNet loads the scenario and creates a federation.
8. Start VR-Forces. Be sure that you use a simulation connection that matches the protocol that QualNet is using. If you are using HLA, be sure that you connect to the same rtiexec as you did for QualNet.
9. Choose **Settings** → **Use External Communication Model**.
10. Click OK.
11. Open the *NetworkCentricForces/CommsDemo* scenario. When the scenario is fully loaded, the Run button on the QualNet toolbar should become enabled. If it does not, there may be a communication problem between the two applications.
12. Click the Play button on the QualNet toolbar.
13. Run the scenario in VR-Forces.



14. Example Entity-Level Scenarios

VR-Forces includes several sample scenarios. This chapter explains how to create the breaching scenario and the embarkexample scenario. It assumes some basic knowledge of VR-Forces concepts and the VR-Forces interface, such as how to select objects, how to pan and zoom the terrain, and so on. If you have not yet reviewed the entity-level sample tutorial in *VR-Forces First Experience*, you may want to try that tutorial before trying these example scenarios. *VR-Forces First Experience* also contains an aggregate-level scenario tutorial.

The Breaching Scenario 14-2
- Create the Scenario 14-3
- Create the Minefield 14-4
- Create the Opposing Forces 14-5
- Create the Tactical Graphics 14-7
- Create the Friendly Simulation Objects 14-8
- Write the Plans 14-10
- Save the Scenario 14-17
- Running the Scenario 14-18

The Embarkexample Scenario 14-19
- Create the Simulation Objects 14-20
- Create the Tactical Graphics 14-23
- Write the Plans for the Entities 14-25

14.1. The Breaching Scenario

This scenario simulates a breaching exercise. In this exercise, four hostile vehicles are arrayed behind a minefield. Three platoons of four tanks each suppress the hostile vehicles, allowing two engineering vehicles to clear the minefield. This scenario uses *EntityLevel.sms*.

Figure 14-1 illustrates the order of battle and tactical graphics in the scenario.

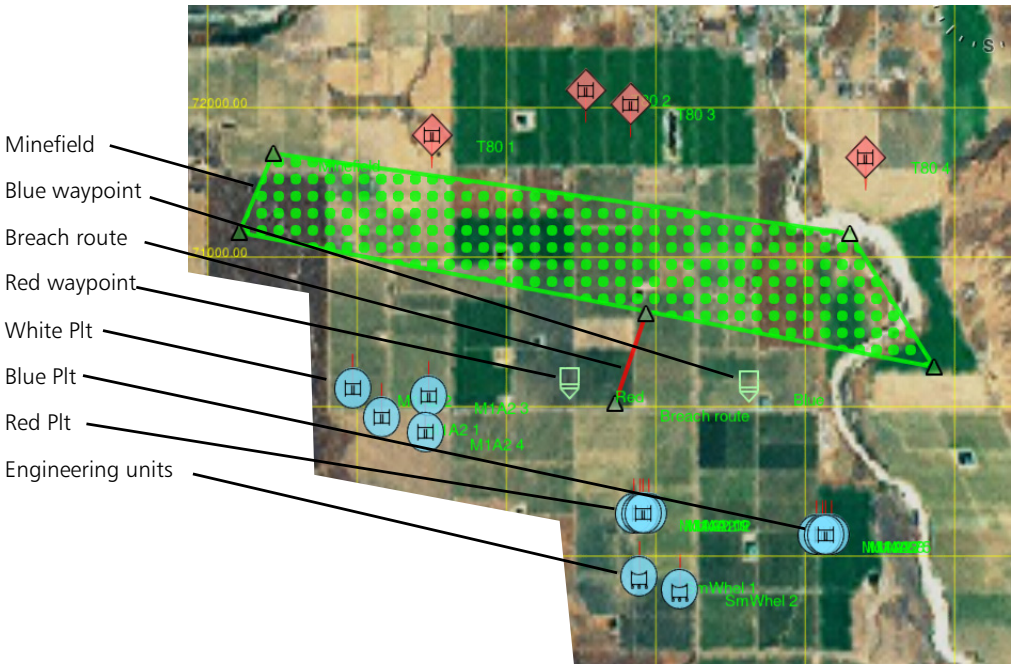


Figure 14-1. Breaching scenario order of battle

The scenario demonstrates the following features of VR-Forces:

- Creating simulation objects.
- Creating tactical graphics (waypoints, routes, areas).
- Creating a predefined unit and creating a unit from individual entities.
- Creating plans, using:
  - Move To.
  - Follow Entity.
  - Follow Route.
  - Conditions.

The scenario plays out as follows for the Blue force simulation objects:

- White Platoon stays in position and fires on hostile forces.
- Red Platoon advances to waypoint Red and provides cover for the engineers.
- Blue Platoon advances to waypoint Blue and provides cover for the engineers.
- When the hostile forces are destroyed, the engineering entities advance on the minefield to clear it.



The scenario has entities that represent engineering entities, but *EntityLevel.sms* does not actually have models for engineering entities.

You can view videos that demonstrate this tutorial at *./doc/vrforces_tutorialvideos.htm*, or click the camera icon next to the procedure whose video you want to view.

14.1.1. Create the Scenario

To create the scenario:



1. Start VR-Forces. The VR-Forces window opens.
2. Choose **File** → **New Scenario**. The Select Simulation Database dialog box opens.
3. Select a terrain database. We used the *SanLuisObispo.mtf* database.
4. Click Open. The New Scenario dialog box opens.
5. Type a name in the Scenario Name box, such as myBreachingScenario. Accept all of the default values for the scenario parameters.
6. Click OK. The San Luis Obispo database is displayed. This database covers a portion of California, U.S.A.

Save the Scenario

Save the scenario periodically as you work on it.

To save the scenario:

1. Choose **File** → **Save Scenario**. The Save Scenario dialog box opens. By default, the name you gave to the scenario when you created it is automatically entered.
2. Click the New Folder button. A new folder is added to the window.
3. Type a name for the new folder, such as myBreaching.
4. Select the folder.
5. Accept the default name or type a name of your choice.
6. Click OK.

14.1.2. Create the Minefield

We will create the minefield first, since it becomes a good reference around which to create the simulation objects and other objects.

To create the minefield:



1. Zoom in on a relatively flat portion of the terrain. This is to minimize the impact of hilly terrain and intervisibility issues on how the simulation runs. For this scenario, we used an area just to the northeast of the center of the terrain (Figure 14-2).



Figure 14-2. San Luis Obispo terrain

2. Click the Tactical Graphics tab on the right side of the window. The Tactical Graphics Palette opens.
3. In the Categories list, select Engineering.
4. In the list of tactical graphics, select Minefield. The Create Minefield tab gets added to the window and the cursor changes to a vertex marker.
5. Click the Create Minefield tab. The Create Minefield dialog box opens (Figure 14-3).
6. In the Name box, type Minefield.

7. Draw an oblong area to define a minefield (Figure 14-3). Click to define the first three vertices. Right-click to define the last vertex.

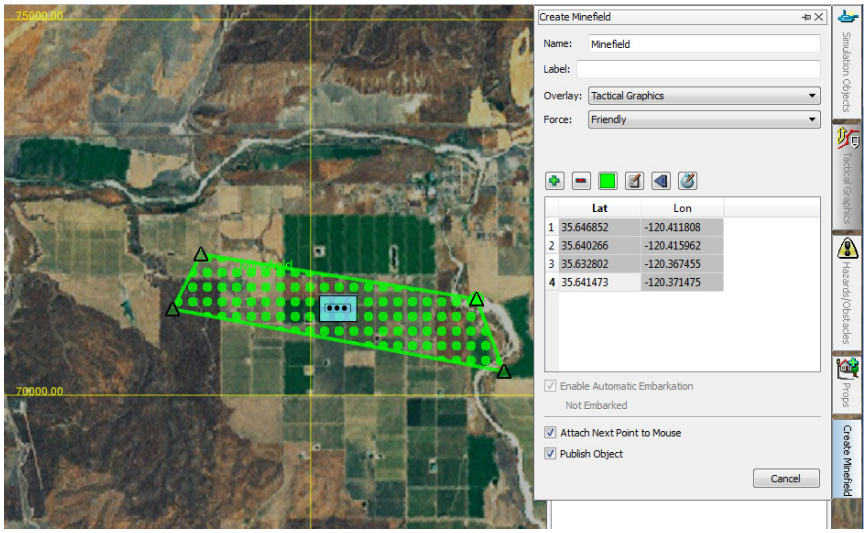


Figure 14-3. Create the minefield

14.1.3. Create the Opposing Forces

Now that we have the minefield as a reference point, we can create the hostile entities.

To create the opposing forces:



1. Click the Simulation Objects Palette (on the right margin of the window).
2. In the Category list, select Ground.
3. In the Force list, select Opposing.
4. In the list of entities, select T-80 Main Battle Tank.
5. Click on the map north of the minefield four times, as indicated in Figure 14-1, to place the T-80s.
6. Right-click to exit create mode.

Aggregate the Opposing Forces

We will want to test for when all of the opposing forces are destroyed. This will be easier to do if they are in a unit.

To aggregate the opposing forces:

1. Select the four T-80 tanks.
2. Choose **Objects** → **Aggregates As**. The Aggregate As dialog box opens (Figure 14-4).

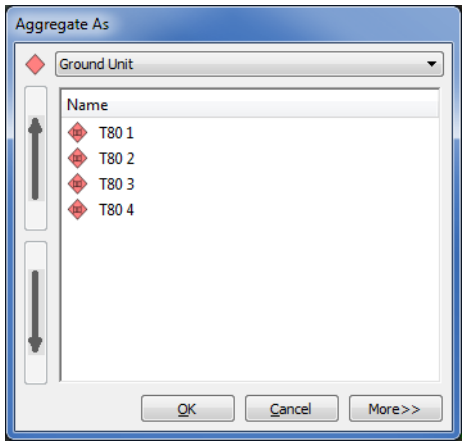


Figure 14-4. Aggregate As dialog box

3. In the list, select Ground Unit.
4. Click OK. The tanks are aggregated into a platoon.
5. In the Echelon View tab on the Objects List Panel, expand the unit (Figure 14-5).

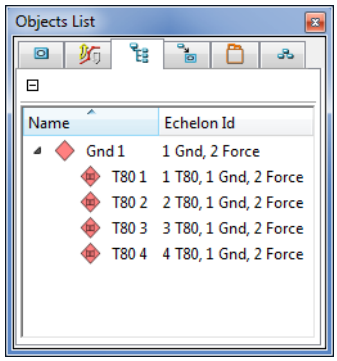


Figure 14-5. Echelon View



In the Breaching example scenario, the opposing force unit is named ARPlt 1.

14.1.4. Create the Tactical Graphics

We need to create two waypoints and a route for the Blue Force.

Create the Waypoints

To create the waypoints:



1. Click the Tactical Graphics Palette (on the right margin of the window).
2. In the Category list, select Control Objects.
3. In the list of control objects, select Waypoint. The Create Waypoint tab is added to the window.
4. Click the Create Waypoint tab. The Create Waypoint dialog box opens.
5. Type a name for the waypoint, such as Red.
6. Using Figure 14-1 as a guide, click on the map where you want to place the waypoint for Red platoon.
7. In the Create Waypoint dialog box, type a name for the second waypoint, such as Blue.
8. Click on the terrain where you want to place the waypoint.
9. Right-click to exit draw mode.

Create the Route

The route will guide the engineers as they swing past the Red Platoon and move to the minefield. The route is shown as Breach route in Figure 14-1.

To create the route:

1. Click the Tactical Graphics Palette.
2. In the list of control objects, select Route. The Create Route tab is added to the window.
3. Click the Create Route tab. The Create Route dialog box opens.
4. Type a name for the route, such as “Breach route”.
5. Click on a point just east of waypoint Red.
6. Move the mouse cursor to the edge of the minefield.
7. Right-click to place the end point in the route.
8. Save the scenario.

At this point, the scenario should look something like Figure 14-6.

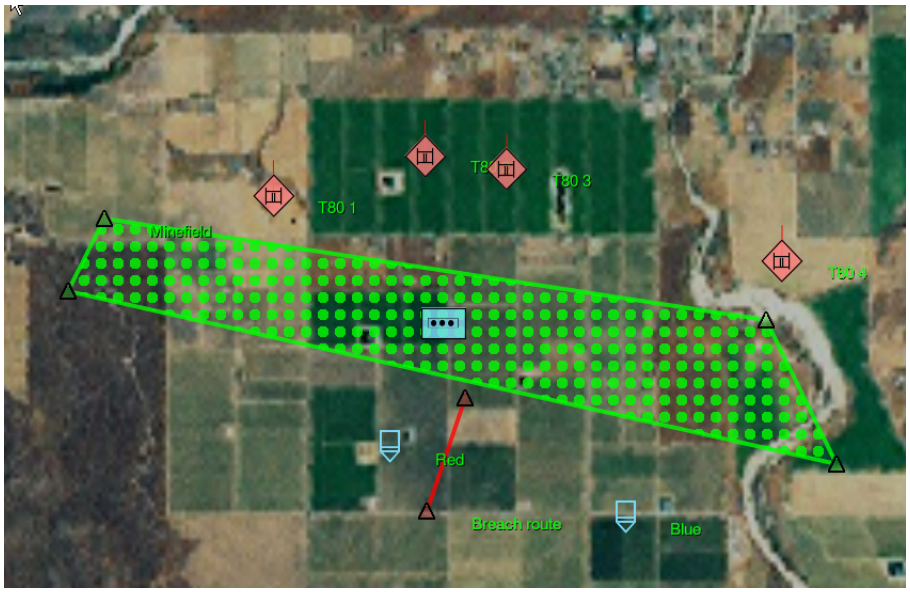


Figure 14-6. Breaching scenario progress

14.1.5. Create the Friendly Simulation Objects

Using Figure 14-1 as a guide to location, create the friendly simulation objects. We will create one platoon from individual entities, and the other two using preconfigured platoons.

To create the friendly simulation objects:



1. Click the Simulation Objects Palette.
2. In the Categories list, select Unit.
3. In the Force list, select Friendly.
4. In the list of entities displayed, select Armor Plt(US).
5. Click on the map to place the red platoon and blue platoon as shown in Figure 14-1. Right-click to exit create mode. The aggregates are created with default names (ARPlt 1 and ARPlt 2). They are disaggregated and expanded, so you can see the individual entities.
6. Select one of the platoons. To select a platoon, you must select the platoon icon, not one of its subordinates. The easiest place to do that is on the Echelon View tab of the Object Lists panel (Figure 14-7).

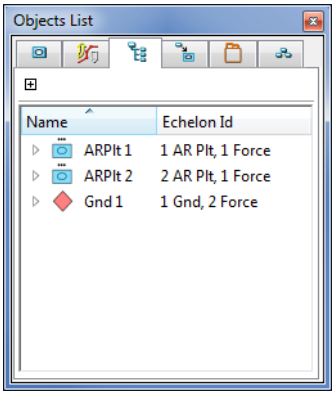


Figure 14-7. Echelon View tab

7. Choose **Objects** → **Edit**. The Edit Armor Plt (US) dialog box opens.
8. In the name text box, replace the default name with Red Plt or Blue Plt, depending on which platoon you selected.
9. Click Update.
10. Rename the other platoon that you created.
11. Click the Simulation Objects Palette.
12. In the Categories list, select Ground.
13. In the list of simulation objects, click M1A2 Tank. The cursor changes to draw mode.
14. Click on the map where White Platoon should be. A simulation object icon is displayed.
15. Click three more times to create the rest of the tanks in White Platoon. Right-click to exit create mode. (We will create the unit from these simulation objects in the next section.)
16. Open the Simulation Objects Palette.
17. In the Ground/Friendly category, select M58 MICLIC.
18. Click twice on the map to the South of Red Plt to create two MICLIC entities.
19. Right-click to exit create mode.
20. Save the scenario.

Create White Platoon

To create the white platoon:

1. Select the four members of White Platoon.
2. Choose **Objects** → **Aggregates As**. The Aggregate As dialog box opens.
3. Select Ground Unit from the list.
4. Click OK.
5. Select the new platoon.
6. Choose **Objects** → **Edit**. The Edit Scene Object dialog box opens.
7. Change the name to White Plt.
8. Save the scenario.

14.1.6. Write the Plans

The plans tell the simulation objects how to carry out their roles in the scenario. The White platoon just stays in place. We do not give it a plan. It will just respond to any opposing force entities that it detects.

Write the Plan for Blue Platoon

Blue Platoon will move to a waypoint. As it moves it responds to any opposing force entities that it detects. The plan is quite simple to write.

To write the plan for Blue Platoon:



1. Select Blue Platoon.
2. Choose **Objects** → **Plan**. The Plan window opens (Figure 14-8).

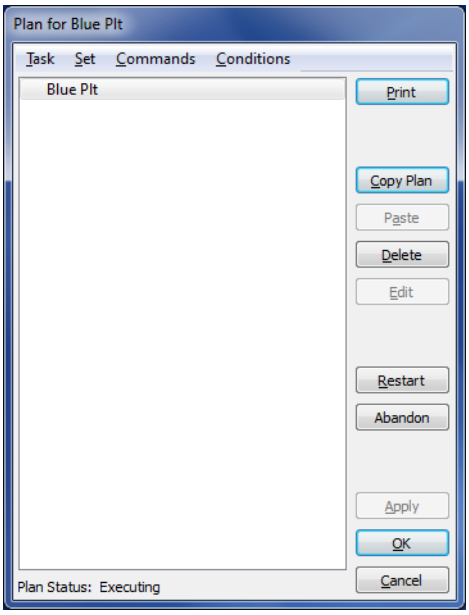


Figure 14-8. Blue Platoon plan

3. In the Plan window, choose **Task** → **Movement** → **Move to Waypoint**. The Move to Waypoint dialog box opens (Figure 14-9).

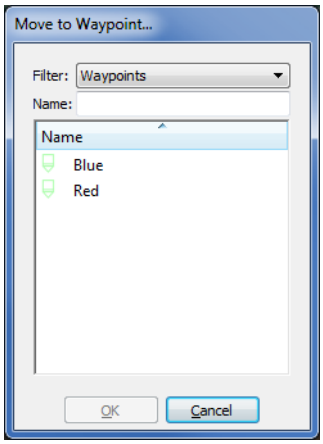


Figure 14-9. Move to Waypoint dialog box

4. In the Move to Waypoint dialog box, or on the map, select the Blue waypoint.
5. Click OK. The task is added to the plan. The plan is now complete (Figure 14-10).

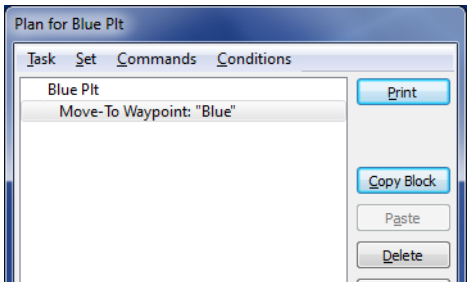


Figure 14-10. Blue plan - complete

6. In the Plan window, click OK.

Write the Plan for Red Platoon

The Red Platoon moves to the Red waypoint. It has another job to do. When all the opposing forces are destroyed, it sends a message on the radio network. The MICLICs are listening for the message. They will not try to clear the minefield until they receive it.

To write the plan for red platoon:

1. Select Red Platoon.
2. Press **p**. The Plan window opens.
3. In the Plan window, choose **Task** → **Movement** → **Move to Waypoint**. The Move to Waypoint dialog box opens (Figure 14-9).
4. In the Move to Waypoint dialog box, or on the map, select the Red waypoint.
5. Click OK. The task is added to the plan (Figure 14-11).

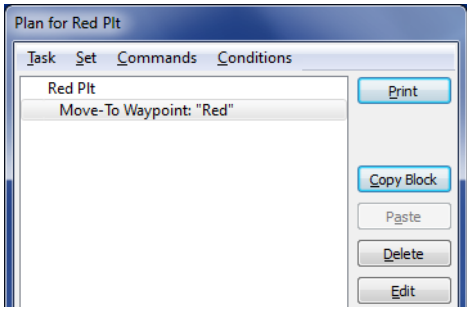


Figure 14-11. Plan for Red Platoon - move to waypoint

6. Select the top line in the plan.
7. In the Plan window, choose **Conditions** → **When**. The When Expression dialog box opens (Figure 14-12).

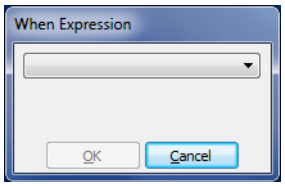


Figure 14-12. When Expression dialog box

8. In the list, select Entity Destroyed. The Entity Destroyed dialog box opens.
9. In the list of simulation objects, select Gnd 1 (Figure 14-13).

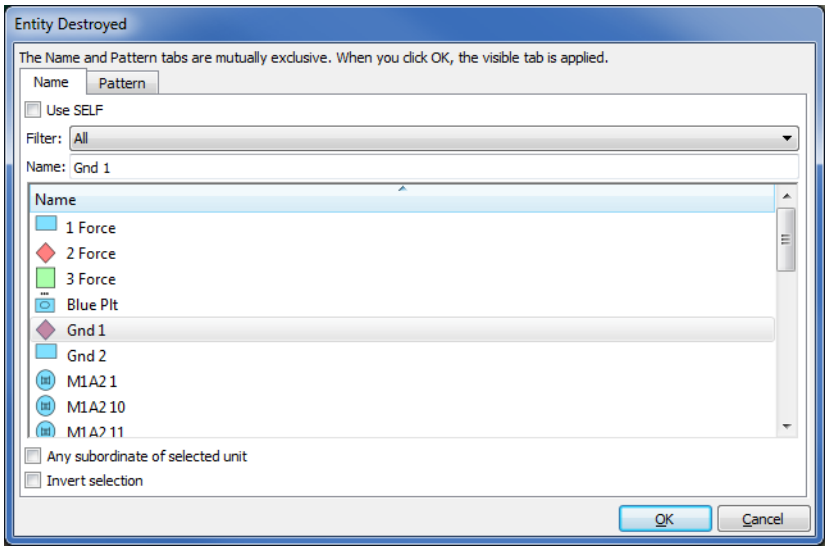


Figure 14-13. Entity Destroyed dialog box

10. Click OK. The expression is added to the When Expression dialog box (Figure 14-14).

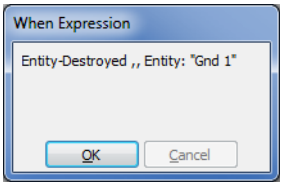


Figure 14-14. Completed When Expression dialog box

11. In the When Expression dialog box, click OK. The When expression is added to the plan (Figure 14-15). Now we need to add tasks to execute when the expression becomes true.

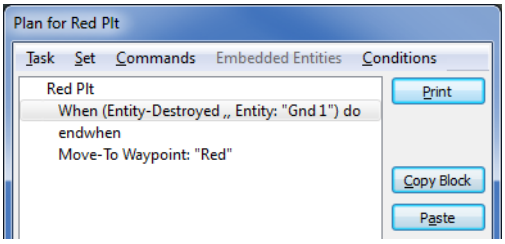


Figure 14-15. Red Plan with incomplete When expression

12. The When expression should be selected. If not, select it.
13. Choose **Task** → **Radio** → **Send Text Message**. The Send Text dialog box opens.
14. In the list, select SmWhel 1.
15. In the Message Text box, type Opposing Force Destroyed (Figure 14-16).

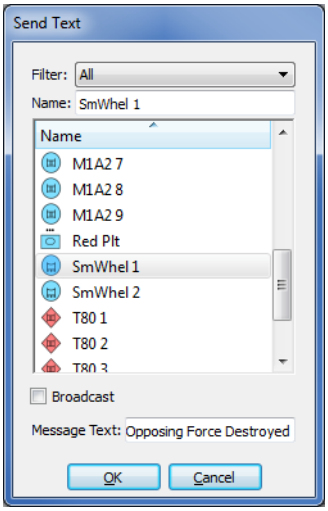


Figure 14-16. Send Text dialog box

16. Click OK. The plan for Red Platoon is finished (Figure 14-17).

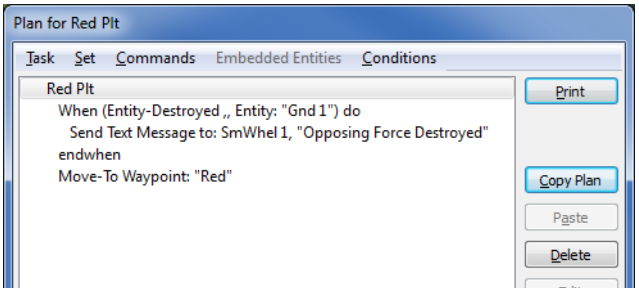


Figure 14-17. Completed plan for Red Platoon

17. Click OK.
18. Save the scenario.

Write the Plans for the Engineering Simulation Objects

When the MICLICs are notified that the opposing force has been destroyed, they move along a route to the edge of the minefield. Then they clear it.

The MICLICs provided with VR-Forces do not actually have the ability to simulate clearing a minefield. We are going to provide some visual interest by using global commands to remove the original minefield and replace it with two minefields with a space between them.

We will start with the plan for SmWhel 1:



1. Select SmWhel 1.
2. Press **p**. The Plan window opens.
3. In the Plan window, choose **Conditions** → **When**. The When Expression dialog box opens.
4. In the list, select Receive Text Message. The Receive Text Message dialog box opens.
5. Type the message that we are waiting for (Opposing Force Destroyed) (Figure 14-18).



Figure 14-18. Receive Text Message dialog box

6. Click OK.

7. In the When Expression dialog box, click OK. The empty expression is added to the plan window (Figure 14-19). Now we need to tell the entity what to do when it gets this message (move along the route, then redraw the minefield).



Figure 14-19. Plan for SmWhel 1

8. In the Plan window, choose **Task** → **Movement** → **Move Along Route**. The Move Along Route dialog box opens.
9. Select Breach route, then click OK. The task is added to the plan.

   As we said earlier, *EntityLevel.sms* does not actually model minefields or the types of simulation objects that could clear one. To provide the visual illusion of clearing a corridor through the minefield, after the MICLICs arrive at the end of the route, we will delete the minefield and create two new minefields that occupy most of the space of the original one, but leaves a corridor for entities to travel through.
10. Choose **Commands** → **Delete**. The Delete dialog box opens.
11. In the list, select Minefield.
12. Click OK.
13. Choose **Commands** → **Create** → **Engineering** → **Neutral** → **Minefield**. The Create Minefield dialog box opens.
14. Type Minefield-left in the Name box.
15. Draw an area over most of the minefield to the left of where the breach route intersects the minefield.
16. Click OK. You will not see this new area, because it does not exist in the simulation yet. It will not exist until the command gets executed.
17. Repeat steps 13 through 16, but this time draw the minefield to the right of the route. Call it Minefield-right. The plan should now be similar to Figure 14-20.

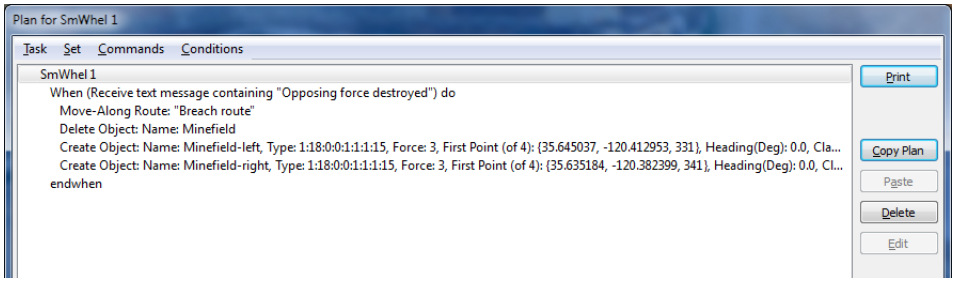


Figure 14-20. Plan for SmWhel 1 after When expression complete

18. Click OK. The plan is now complete.
19. Save the scenario.

Write the Plan for SmWhel2 2

The plan for SmWhel 2 is a bit simpler than that of SmWhel 1. It follows SmWhel 1, offset to the right, so that it arrives at the minefield next to the other entity. Try creating the plan yourself. It should look like (Figure 14-21).

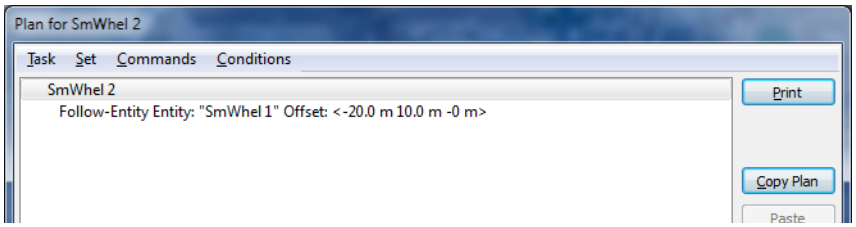


Figure 14-21. Plan for SmWhel 2

14.1.7. Save the Scenario

Be sure to save the scenario before you run the simulation. If you want to save it in its starting state, do not save it after you run the simulation.

14.1.8. Running the Scenario

Once you have created the scenario and saved it, click the Run button and watch the simulation run. Open the plan windows for the MICLICs and observe the progression of tasks through the plan.

If hostile entities do not get destroyed, it may be that the blue entities cannot see them. If they cannot see them, they cannot shoot each other. Use the intervisibility feature to test line-of-sight. Move the entities around, or find a flatter section of the terrain until you get the results you want.

You can run the breaching scenario included with VR-Forces and see how it compares to the one that you created.

Figure 14-22 shows the breaching scenario after the MICLICs have completed their plans. All of the aggregates are expanded so that you can see the individual entities.

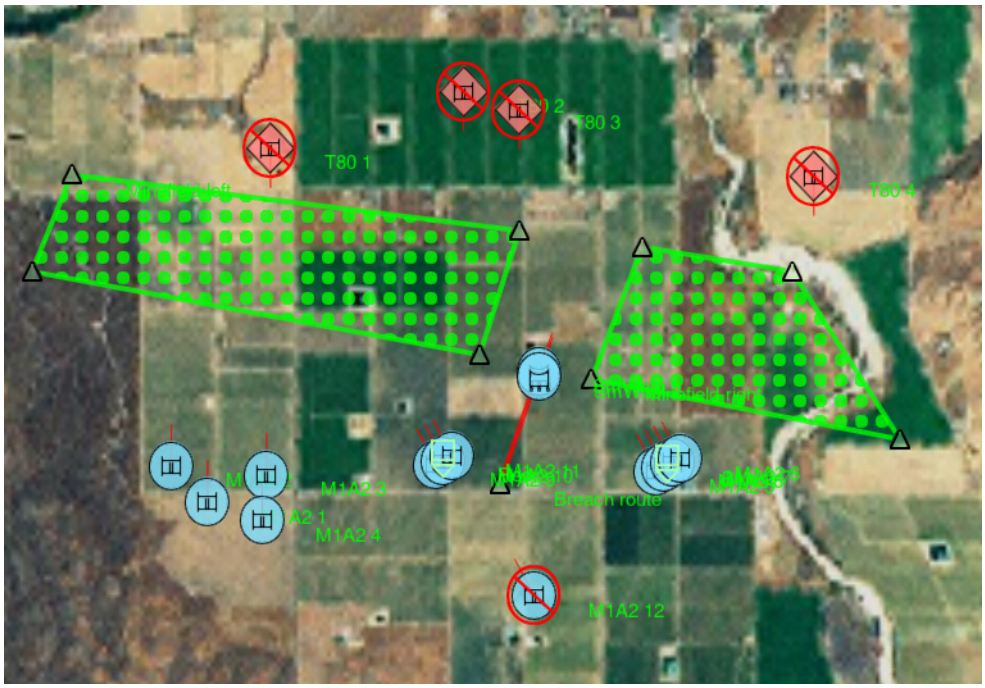


Figure 14-22. Breaching scenario after minefield cleared

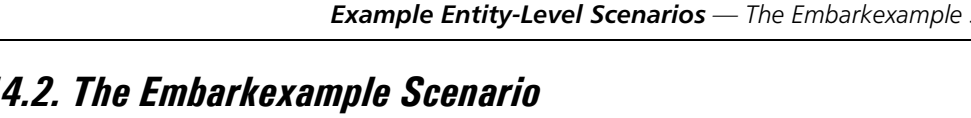
14.2. The Embarkexample Scenario

The embarkexample scenario (Figure 14-23) demonstrates the embarkation feature. It shows the following:

- Embarking and disembarking entities using embarkation tasks.
- Embarking an entity using automatic embarkation.

The scenario also demonstrates various types of conditional statements including nested conditions, radio text messages, radio task messages, and a global plan.

A completed scenario is included with VR-Forces in *./userData/scenarios/embarkexample*. This section is a step-by-step explanation of how to create the scenario. It assumes that you have completed the breaching scenario example and does not provide quite as much detail for how to create simulation objects and tactical graphics. This scenario uses *EntityLevel.sms*.



If you are using HLA, you must use RPR FOM 2 draft 17 or later for scenarios that use embarkation.



Figure 14-23. Embarkexample scenario

14.2.1. Create the Simulation Objects

The scenario has the following simulation objects (marking text in Figure 14-23):

- An F/A-18 Hornet fixed-wing aircraft (FtrFW 1).
- A helicopter (UtilRW 1, not visible) embarked on an LSD-49 Harpers Ferry ship (Harper 1).
- A DI (R 1) standing behind a Bradley vehicle (M2A2 1) in the town.
- An aircraft carrier (Carrier 1).

The scenario has the following tactical graphics:

- A waypoint for the Bradley vehicle to drive to.
- A waypoint for the helicopter to land at.
- A waypoint on the carrier deck for the DI to walk to.

Create the Entities

To create the entities:

1. Start VR-Forces.
2. Choose **File** → **New Scenario**. The Initial Scenario Information dialog box opens.
3. Select *AlaMoana.mtf*.
4. Click Open. The New Scenario dialog box opens.
5. Click OK. The Ala Moana database is displayed.
6. Navigate to Honolulu, Hawaii. (You can move here quickly if you load the *AlaMoandInset.osrx* saved views file. For information about loading saved views, please see "Saving and Recalling Views," on page 44-29)
7. Using the Simulation Objects Palette, create the following entities in the locations shown in Figure 14-23:
   - Harpers Ferry-class LSD.
   - Nimitz class-Carrier.
   - F/A-18 Hornet.
   - M2A2 Bradley IFV (Figure 14-24 shows a closeup view of the location of the Mm2A2 Bradley and the US Army M16.)
   - US Army M16.

   We will create the helicopter in a separate procedure that demonstrates automatic embarkation.

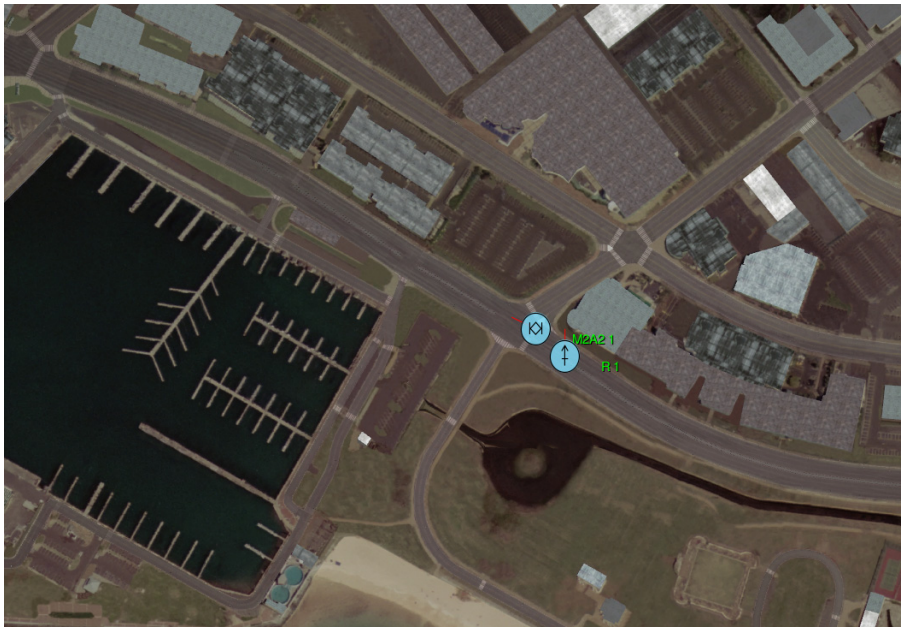


Figure 14-24. M2A2 Bradley and US Army M16 closeup

8. Set the altitude of the F/A-18 to 1000 meters.
9. Save the scenario.

Add the Helicopter

We will add the helicopter using automatic embarkation.

To add the helicopter:

1. Set the display to Stealth observer mode.
2. In the Objects List Panel, right-click Harper 1 and choose **Observer Attach** → **Attach Follow**. The observer attaches to the Harpers Ferry LSD ship.
3. Press the right arrow key a few times to get a broadside view of the ship.
4. On the Simulation Objects Palette, in the Rotary Wing category, select CH-46E Sea Knight. The cursor changes to create mode.
5. Move the cursor over the Harpers Ferry LSD ship. A green selection box is displayed (Figure 14-25).

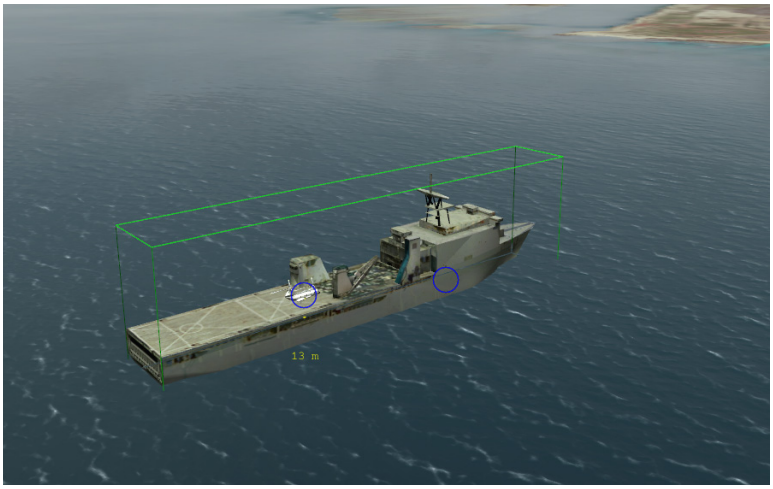


Figure 14-25. Automatic embarkation selection box

6. Click the left mouse button. A CH46E Sea Knight helicopter is created and embarked onto the Harpers Ferry LSD.
7. Right-click to exit create mode.

14.2.2. Create the Tactical Graphics

We need a few tactical graphics to specify the movement tasks for the entities. The Bradley vehicle will drive across town to a landing area. The helicopter will land there and pick up the DI.

Create Waypoints in the Helicopter Landing Area

To create the waypoints at the landing area:

1. Set the observer to Plan View observer mode.
2. Zoom in on the landing area, outlined in Figure 14-23 and shown more closely in Figure 14-26.
3. Put a waypoint along the road for the Bradley to drive to.
4. Put a waypoint in the middle of the grassy area for the helicopter to land at.

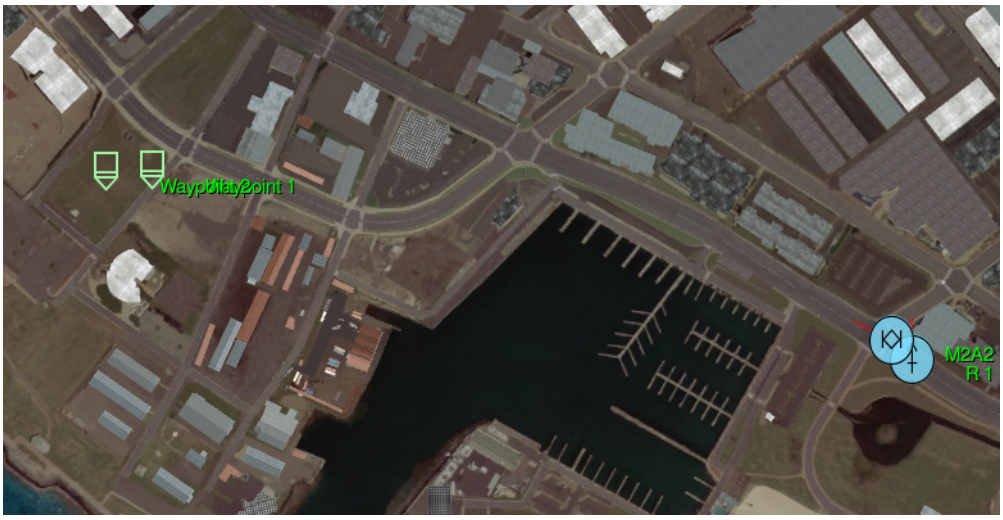


Figure 14-26. Waypoints at landing area

Attach a Waypoint to the Carrier

We want to attach a waypoint to the carrier. However, in the 2D view, there is no way to figure out where to place the waypoint because the carrier is shown as a 2D icon. So, we will switch to the 3D view.

To attach a waypoint to the carrier:

1. Change to Stealth observer mode.
2. Attach the observer to the aircraft carrier.
3. Zoom in a bit so you have a good view of the carrier deck and control tower.
4. Choose **Create** → **Waypoint**.
5. Click on the deck of the carrier next to the control tower (Figure 14-27).



Figure 14-27. Waypoint on carrier

6. Right-click.
7. Save the scenario.

14.2.3. Write the Plans for the Entities

We need to write plans for the entities to do the following:

- The jet will embark on the carrier (land and taxi to a parking place). When instructed to do so by the DI, it will disembark (take off). It then gets a task from a global plan to fly to 5000 meters at its current heading.
- The DI will embark on the Bradley and ride with it to the landing area. It will then disembark from the Bradley and embark on the helicopter. When the helicopter gets to the carrier, the DI will disembark and walk to the waypoint. It will then tell the helicopter and the jet to disembark.
- The helicopter will disembark from the Harpers Ferry LSD and fly to the landing area, where it will land. The DIs will embark on it. Then it will take off and embark on the carrier. When instructed to do so by the DI, it disembarks from the carrier and embarks on the Harpers Ferry LSD.

As you can see, the entities do a lot of embarking and disembarking. They will use radio messages to coordinate their activities.

Write the Fixed-Wing Aircraft's Plan

At the start of the scenario, the jet is in the air executing its default behavior, which is to loiter. Its plan will tell it to embark on the carrier. To accomplish this, it creates a route to line it up with the carrier deck, lands on the deck, and taxis to a parking slot. However, you do not have to figure any of this out. You just have to tell it to embark.

A `When` condition triggers when the jet receives a message from the DI. It disembarks and flies to 5000 meters.

To write the fixed-wing aircraft's plan:

1. Select the fixed-wing entity.
2. Choose **Objects** → **Plan**. The Plan window opens.
3. Choose **Task** → **Embarkation** → **Embark**. The Embark On dialog box opens.
4. In the Embark On dialog box, select Carrier 1 from the list and click OK.
5. Choose **Conditions** → **When**. The When Expression dialog box opens.
6. In the When Expression dialog box, select Receive Text Message.
7. In the Receive Text Message dialog box, type the message text `Start patrol.`.
8. Click OK to back out of the dialog boxes.
9. In the When block, add a Disembark task.
10. Choose **Task** → **Fly Altitude**. The Fly Altitude dialog box opens. (When the jet disembarks, it loiters at about 175 meters. We want to get it quite a bit higher.)
11. Specify an altitude of 5000 meters.
12. Specify a climb rate of 60 km/hr.

13. In the Plan window, click OK. The plan for the fixed-wing entity is finished (Figure 14-28).

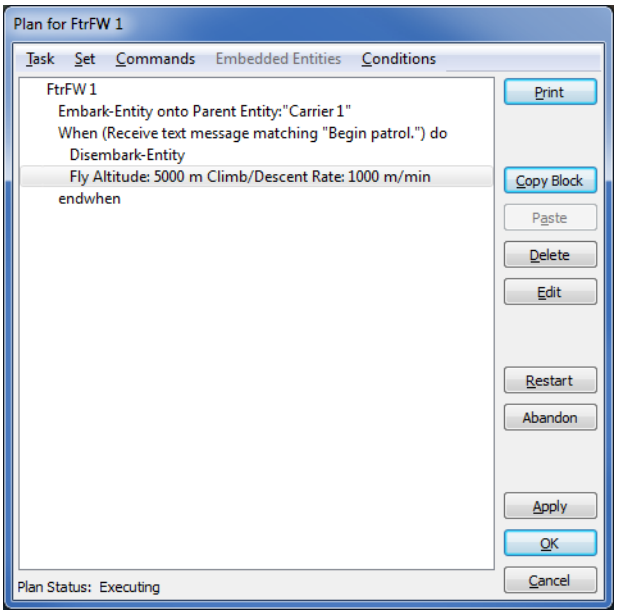


Figure 14-28. FtrFW 1 plan

Write the Helicopter's Plan

The helicopter needs to disembark from the Harpers Ferry LSD, fly to the landing area, pick up the DI, and fly to the carrier. We will have it wait on the ship until the DI is embarked on the Bradley, so that the timing works out.

To write the helicopter's plan:

1. On the Objects List Panel, Entity List view, select the helicopter entity. (Since the helicopter is embarked, it is not visible on the map in Plan View mode, so you cannot select it there. If you are in the 3D view, you can select the copter in the Objects List Panel or by clicking the model.)
2. Choose **Objects** → **Plan**. The Plan window opens.
3. Choose **Conditions** → **When**. The When Expression dialog box opens.
4. Select Entity Embarked. The Entity Embarked dialog box opens.
5. On the Name tab, in the top list, select R 1. This is the entity whose embarkation status we want the helicopter to monitor.
6. In the bottom list, select M2A2 1. We want to test the condition that R 1 is embarked on M2A2 1.

7. In the Entity Embarked dialog box, click OK. Then, click OK in the When Expression dialog box. The condition is added to the plan. Now we need to specify what the helicopter should do when the condition is true.
8. The line in the plan window that starts "When (Entity-Embarked" should already be selected. If it is not, select it now.
9. In the Plan window, choose **Task** → **Disembark**.
10. Choose **Task** → **Rotary Wing Land**. The Rotary Wing Landing dialog box opens.
11. In the list of waypoints, select Waypoint 2 and click OK.

    When the helicopter lands, it sends a message to the DIs to tell them that it is OK to embark.
12. Choose **Task** → **Send Text Message**. The Send Text dialog box opens.
13. Select R 1. This will direct the message just to this entity.
14. In the Message Text box, type `All aboard.`.
15. Click OK. When the helicopter lands, it will tell DI 1 to load up, so that it knows it is time to embark on the helicopter.

    After the helicopter lands, it needs to wait until the DI has embarked before it takes off. So we use a When statement with an Entity Embarked condition. The helicopter does not need to watch for this entity to be embarked until it lands, so we can put the When statement in the middle of the plan rather than at the beginning as we might with other conditions.
16. Choose **Conditions** → **When**. The When Expression dialog box opens.
17. Select Entity Embarked. The Entity Embarked dialog box opens.
18. On the Name tab, in the top list, select R 1.
19. In the bottom list, select UtilRW1.
20. In the Entity Embarked dialog box, click OK. Then, click OK in the When Expression dialog box. The condition is added to the plan.
21. The line in the plan window that starts "When (Entity-Embarked" should already be selected. If it is not, select it now.
22. In the Plan window, choose **Task** → **Embark**.
23. Specify that the helicopter embark on Carrier 1.
24. Click OK. The plan is complete (Figure 14-29).

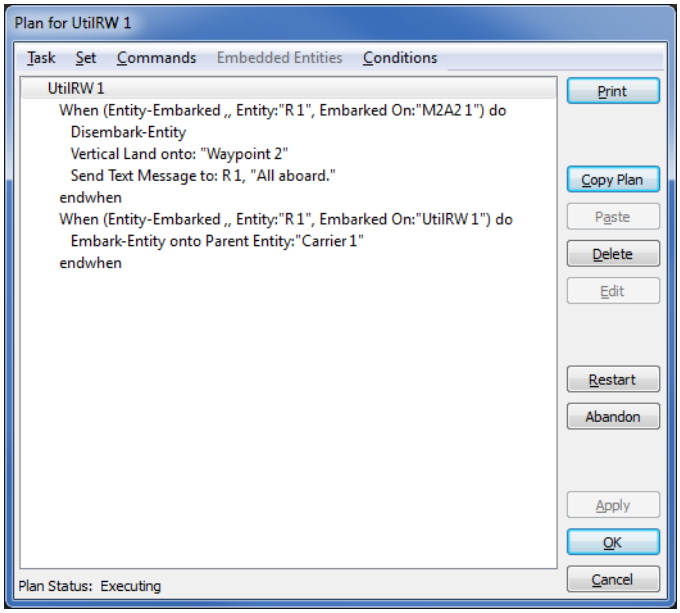


Figure 14-29. Rotary-wing plan

Write the Bradley Vehicle's Plan

The Bradley waits until R 1 embarks on it, then drives to the landing area. It tells R 1 to get out.

To write the plan for the Bradley vehicle:

1. Select the Bradley.
2. Choose **Objects** → **Plan**. The Plan window opens.
3. Choose **Conditions** → **When**. The When Expression dialog box opens.
4. Select Entity Embarked. The Entity Embarked dialog box opens.
5. On the Name tab, in the top list, select R 1. This is the entity whose embarkation status we want the helicopter to monitor.
6. In the bottom list, select M2A2 1. We want to test the condition that R 1 is embarked on M2A2 1.
7. In the Entity Embarked dialog box, click OK. Then, click OK in the When Expression dialog box. The condition is added to the plan. Now we need to specify what the entity should do when the condition is true.
8. The line in the plan window that starts "When (Entity-Embarked" should already be selected. If it is not, select it now.
9. Within the When block, add a Move to Waypoint (Plan Along Roads) task for the Bradley to move to Waypoint 1.

10. Add a Send Text Message task to send the text `Disembark now.` to R 1.
11. Save the plan (Figure 14-30).

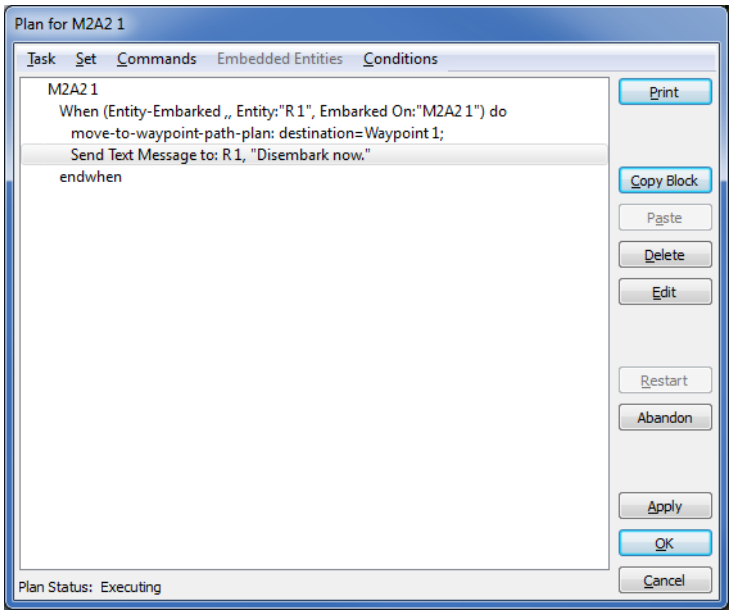


Figure 14-30. M2A2 plan

Write the Plan for the DI

The DI embarks on the Bradley and rides to the landing area. It gets on the helicopter and flies to the carrier, where it disembarks and walks to the waypoint on the carrier deck. Then it tells the helicopter and jet to disembark.

The logic of the plan uses three `When` conditions. The first sets up the response of the DI when the Bradley reaches the landing area and tells it to disembark. The second has the tasks for the entity when it is told to embark on the helicopter. The third is inside the second.

The organization of conditional statements in a plan is important. (For discussion, please see "Considerations for Using Triggers," on page 31-15.) We want to be sure that the conditions in the plan get registered, so we put them at the top of the plan. However, we do not want some tasks to be carried out until the DI is embarked on the helicopter and the helicopter is embarked on the carrier, so we nest a `When` statement inside a `When` statement.

To write the plan for the DI:

1. Select the DI.
2. Choose **Objects** → **Plan**. The Plan window opens.
3. Choose **Conditions** → **When**. The When Expression dialog box opens.
4. In the When Expression dialog box, select Receive Text Message.
5. In the Receive Text Message dialog box, type the message text `Disembark now.`. This matches the text you put in the Send Text Message task for the Bradley vehicle.
6. Click OK to back out of the dialog boxes.
7. In the When block, add a Disembark task. The plan should look like Figure 14-31.

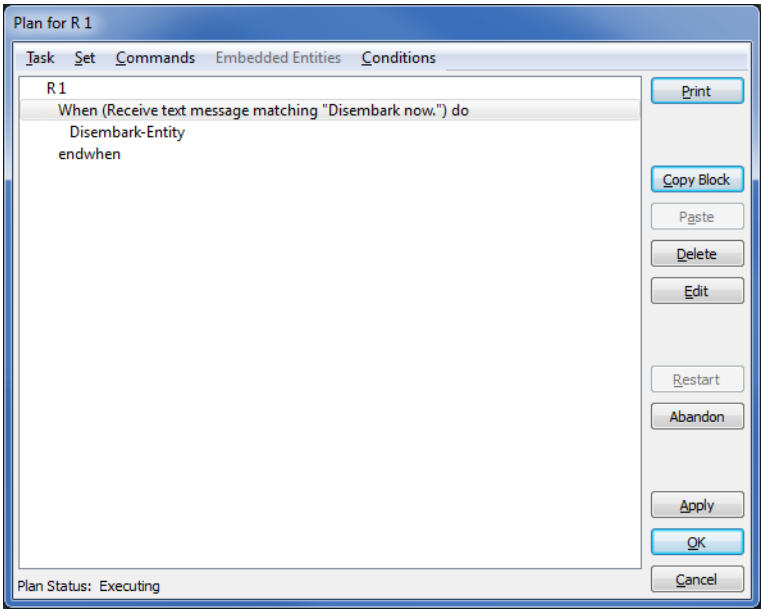


Figure 14-31. RW 1 plan with first When condition

8. Select endwhen.
9. Choose **Conditions** → **When**. The When Expression dialog box opens.
10. In the When Expression dialog box, select Receive Text Message.
11. In the Receive Text Message dialog box, type the message text `All aboard.`. This matches the text you put in the Send Text Message task for the helicopter. In the When block, add the following tasks and sets.
12. Set the speed to 10.0 km/h. This is fast enough so that the DI will run to the helicopter instead of walking.
13. Add a task to embark on UtilRW1. The plan should now look like Figure 14-32.

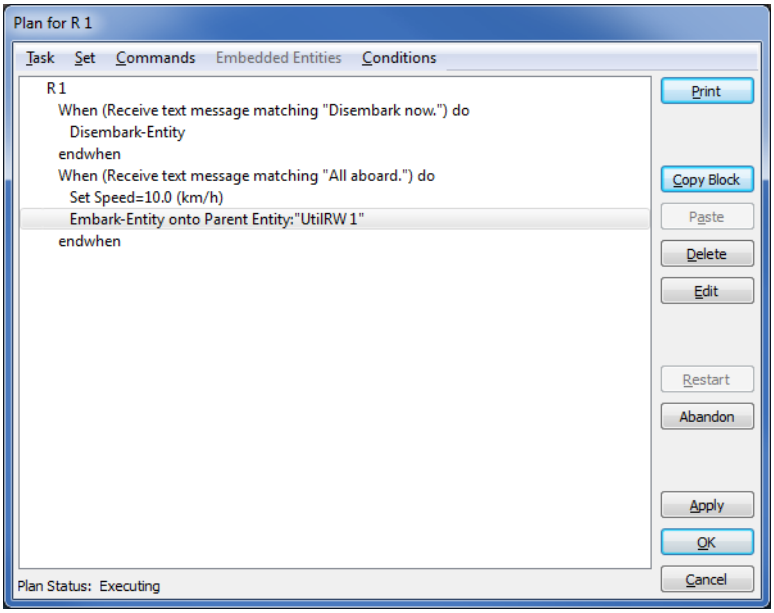


Figure 14-32. RW 1 plan with second When condition

14. Add another `When` condition inside this block under the Embark Entity task. Select the Entity Embarked condition and specify the helicopter embarked on the carrier. This condition will trigger when the helicopter flies to the carrier and lands on the deck.
15. In the `When` block, add a Wait Duration, 30 seconds task. This task means that when the helicopter embarks on the carrier, the DI entity waits 30 seconds before it disembarks. (This is a timing issue. The simulation engine usually considers the helicopter to be embarked before it has finished the visualization of the landing process. If you do not put in a Wait here, the DI will disembark as soon as the simulation engine thinks the helicopter is embarked. This could put it in the ocean or under the carrier deck.)
16. Add a Disembark task. This disembarks the DI from the helicopter. Since the helicopter is embarked on the carrier, the DI is automatically embarked on the carrier.
17. Add a Move to Waypoint task to Waypoint 3. After the DI disembarks from the helicopter, it walks across the deck to the waypoint.
18. Send a task message to UtilRW 1 to embark on Harper 1. It will disembark from the carrier, fly to the Harpers Ferry LSD, and embark on it.
19. Send a text message to FtrFW 1 to `Start patrol.`. The plan should look like Figure 14-33.

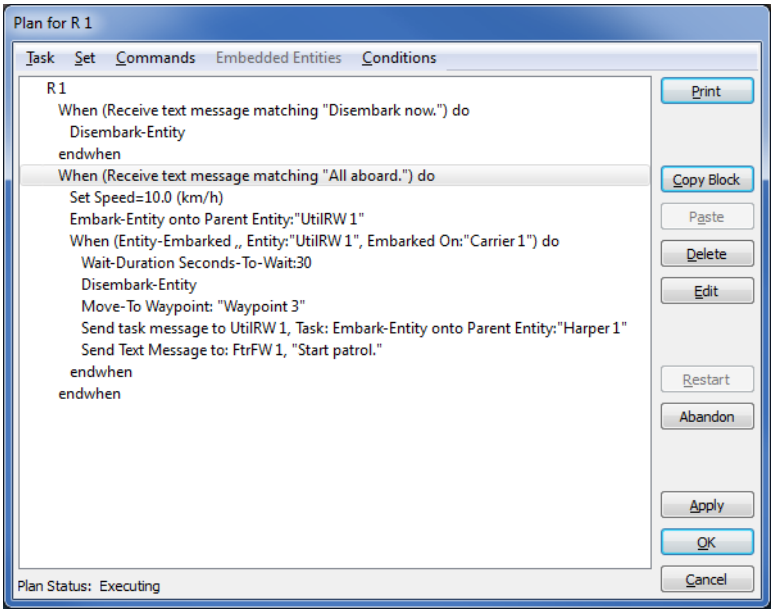


Figure 14-33. RW 1 plan with nested When condition

20. Select the last statement in the plan (endwhen).
21. Task this entity to embark on the Bradley vehicle. This task will actually be the first thing the DI does when the scenario starts. The plan should look like (Figure 14-34).

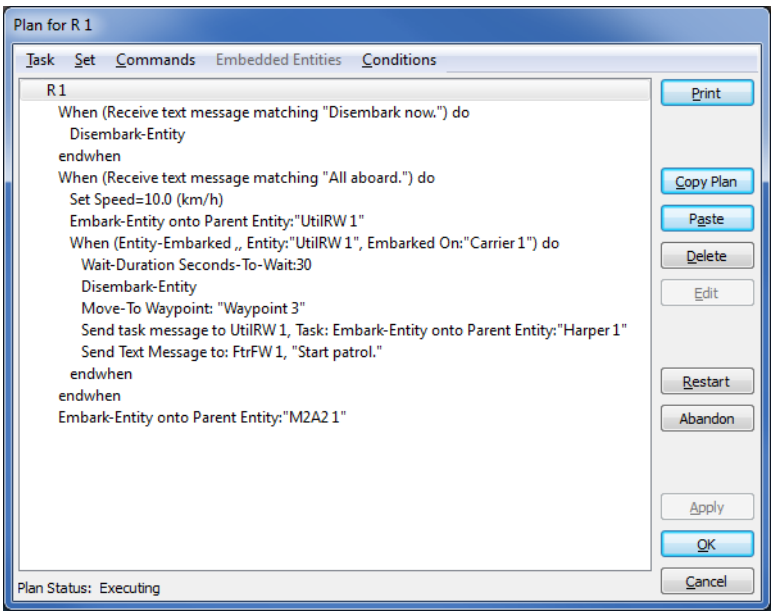


Figure 14-34. RW 1 plan

22. Click OK.
23. Save the scenario.
24. Run the scenario.



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.



15. Introduction to Simulation Objects

Simulation objects are the entities and units that VR-Forces simulates as part of a scenario.

Simulation Objects 15-2
How Simulation Objects are Identified 15-2
How Simulation Objects are Organized 15-5
How Simulation Objects Communicate 15-6
VR-Forces Uses a Component Architecture 15-7
Simulation Object Behaviors and Tasks 15-9
Entity-Level Modeling and Aggregate-Level Modeling 15-9

15.1. Simulation Objects

The players in a simulation are called simulation objects. There are two broad classes of simulation objects:

- Entities, which represent a discrete individual vehicle or person, such as a tank or a submarine.
- Units, which 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 15-9.

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*.

15.1.1. How Simulation Objects are Identified

VR-Forces identifies simulation objects by UUID (universal unique ID), name, echelon ID, object ID, and label. Table 15-1 summarizes the characteristics of each type of identifier.

Table 15-1: 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 15-1: 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.

UUIDs

UUIDs are unique identifiers generated by VR-Forces to ensure that every simulation object can be uniquely identified and that this 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.)

Simulation Object Names

A name is a text string that, by default, consists of the simulation object category, 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`, `USAtRW 1`, and `USAtRW 2`. When VR-Forces generates a name, it picks an integer that makes the name unique. However, if you import a scenario into another one 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.

Echelon IDs

An echelon ID is a text string that identifies the simulation object's place in the force hierarchy. For example, the hypothetical entities in the previous section would be part of Blue force, or Force 1. The echelon ID for `M1A2 1` 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 15-5. 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.

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.

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 18-18.



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 18-10.

15.1.2. 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
```

How Echelon IDs are Assigned

The category 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 echelon ID category 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.

15.1.3. 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/simulationModel-Sets/<sms>/CommModelParams.mtl*. For details about configuring radios, please see “Configuring Radios,” on page 70-13.

External Communication Model

VR-Forces includes an external communication model that can communicate with an external communication effects server to determine how to deliver radio messages. For details, please see Chapter 13, *Using a Communications Effects Server*.

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.

15.1.4. 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.

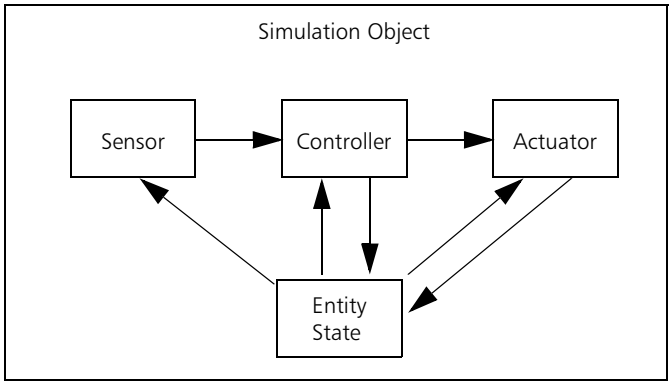


Figure 15-1. Simulation object component architecture

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 B, *Object Parameters*.

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.

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.

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.

15.1.5. 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*.

To learn about changing behavior parameters, please see Chapter 70, *SMS Configuration Files*.



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.

15.2. 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 focuses on 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 MÄK 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.



16. 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 16-3
Selecting the Object to Create 16-4
Selecting the Object to Create on an Object Palette 16-5
Selecting the Object to Create on the Create Menu 16-8
Draw Mode 16-9
Placing Objects 16-11
Placing an Object Using Default Values (Click to Create) 16-11
Specifying an Object's Properties Before You Create It (Click to Locate) 16-13
Specifying an Object's Altitude 16-14
Setting Altitude Dynamically 16-15
Setting Altitude in the Create Object or Edit Object Dialog Box 16-15
Specifying the Altitude for All of the Vertices in a Route 16-16
Specifying an Object's Heading Dynamically 16-17
Locking the Mouse to the Object Being Created 16-18
Moving Objects 16-19
Dragging an Object to a New Location 16-19
Copying and Pasting Objects 16-21
Copying Objects 16-21
Pasting Objects 16-22
Pasting Specific Entity Characteristics 16-23
Simulation Object Groups 16-25
Creating Cultural Features 16-25

Creating Props 16-26
Adding an Object to the Favorites List 16-26

16.1. 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 and Hazards/Obstacles palettes. 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 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 16-1 lists the properties that you can set for each object type. You can also edit these same properties after you create an object.

Table 16-1: Editable object properties

	Entity	Cultural Feature	Tactical Graphic	Prop
Force	X	X	X	
Heading	X	X		
Label (and extended labels)	X	X		
Location	X	X	X	X
Name	X	X	X	
Overlay	X (Edit only)	X (Edit only)	X	
Style			X	

16.2. 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, Hazards/Obstacles 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 16-11.

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 16-26.

16.2.1. 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.
- Hazards/Obstacles. Lets you create weather objects, volumetric lines and areas, contamination areas, and various types of hazards and obstacles. (Some of these graphics are aggregate-level only.)
- Props Palette. Create props that can be saved as part of the terrain.

Figure 16-1 shows the location of the palettes.



Figure 16-1. Object creation palette tabs

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 16-2).

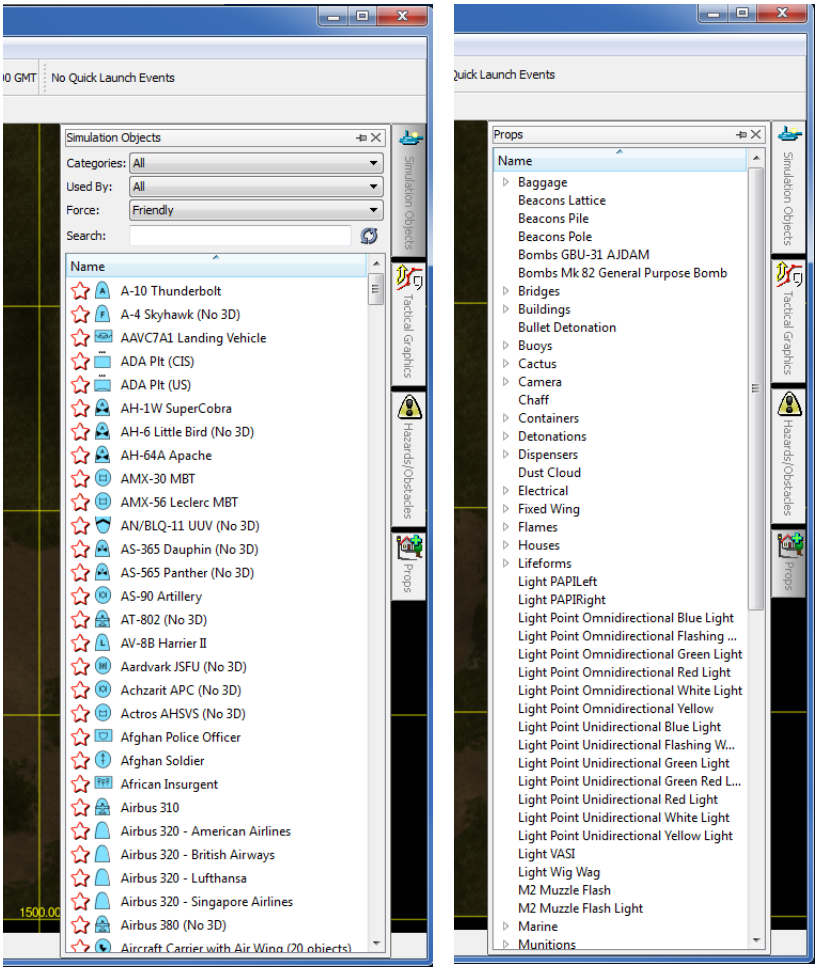


Figure 16-2. Object palettes

By default, the palettes list all objects that can be created. You can filter the Simulation Objects Palette list by category, force, and country of use. You can filter the Tactical Graphics Palette and Hazards/Obstacles Palette by category and force. The Props Palette organizes props by category in a tree list.

3. Optionally, in the Categories list, select the category for the object you want to create, such as ground or fixed-wing for entities, or control objects for tactical graphics.

4. Optionally, filter the Simulation Objects Palette by selecting a country in the Used By list.



The Used By filter lists simulation objects of the selected Category 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 Used By Countries List," on page 64-8.

5. In the Force list, select the force for which you want to create this simulation object. All object types for a given platform are listed. There is no preconceived force for any of them.
6. 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 16-9.
7. Create the object, as described in "Placing Objects," on page 16-11.

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 74-2.)

➤ 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 16-3).

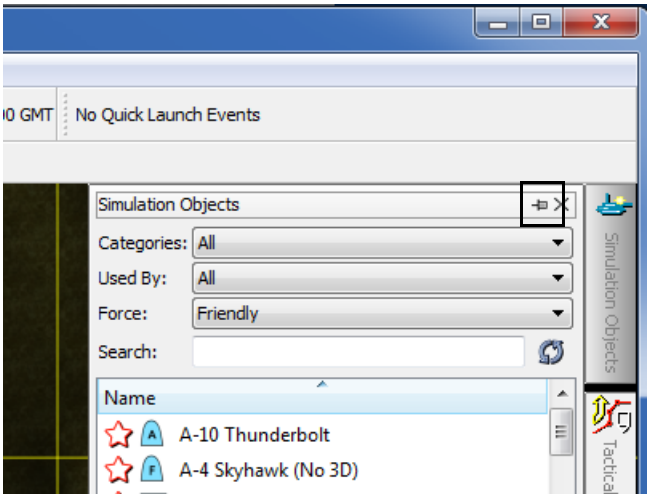


Figure 16-3. Palette pin

16.2.2. 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** → *category* → *object_type*, where:
   - *category* 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 16-9.
3. Create the object, as described in "Placing Objects," on page 16-11.

16.2.3. 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 16-4).



Figure 16-4. 2D entity creation and Create *object* tab

In the 3D view, the cursor changes to the 3D model for the chosen entity (Figure 16-5). 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 as in Figure 2-1 Ground-level view

Figure 16-5. 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 16-6 illustrates tactical graphics draw mode cursors.

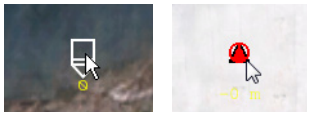


Figure 16-6. Tactical graphics cursors in draw mode

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.

16.3. 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 Props Palette.

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.

16.3.1. 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 19-8 and "Attaching a Tactical Graphic Automatically," on page 35-11.

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 16-4. A Create object tab is added to the side of the window below the Props Palette.
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 16-18.

   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 35, *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 33-5.

---

3. Optionally, you can set the altitude of an object as you create it. For details, please see "Setting Altitude Dynamically," on page 16-15.
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 16-17.



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.

16.3.2. 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 16-1 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 16-4. A Create Object tab is added to the window below the Props Palette (Figure 16-4).
2. Click the Create Object tab. A Create Object dialog box that is appropriate for the object opens (Figure 16-9).

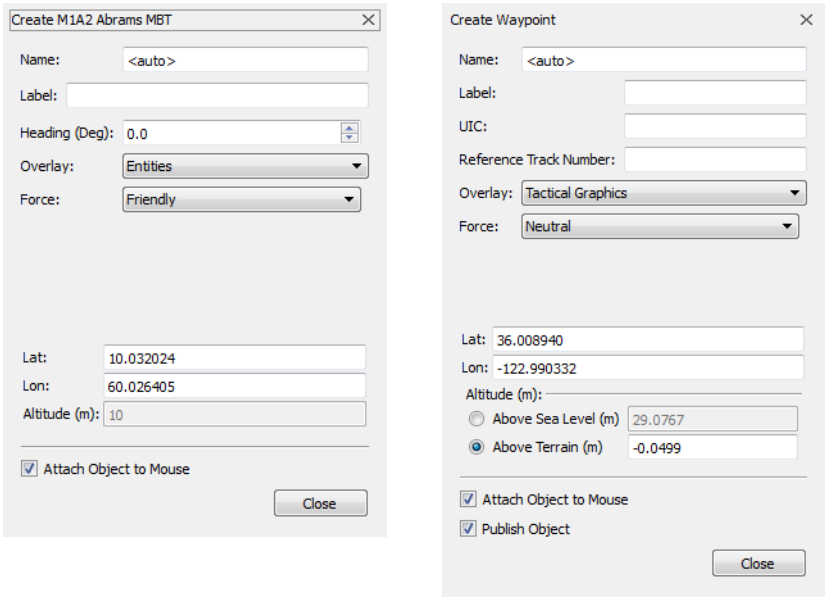


Figure 16-7. Create Object dialog boxes

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 16-18.
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 15-2.
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 18-18.
7. To specify the object's heading, type or select a value, in degrees, in the Heading box.
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 16-14.

    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 35, *Creating and Editing Tactical Graphics*.
11. For tactical graphics, specify whether or not to publish the object.
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.

16.4. 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.)

16.4.1. 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.

16.4.2. 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 16-16.

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.

16.4.3. 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 16-8).

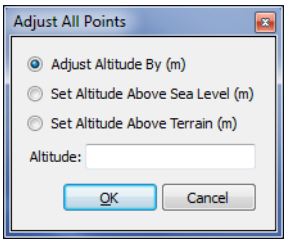


Figure 16-8. Adjust All Points dialog box

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.

16.5. 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 30-23.)

➤ 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.



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.

16.6. 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.

➤ 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 16-9). When you clear the check box, a Create button gets added to the dialog box. The setting gets carried over to future object creations.

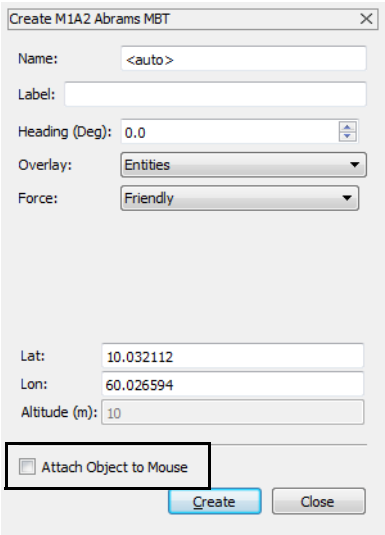


Figure 16-9. Create Object dialog box

16.7. 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 request.

For information about setting the location of a simulation object, please see "Location," on page 30-26.

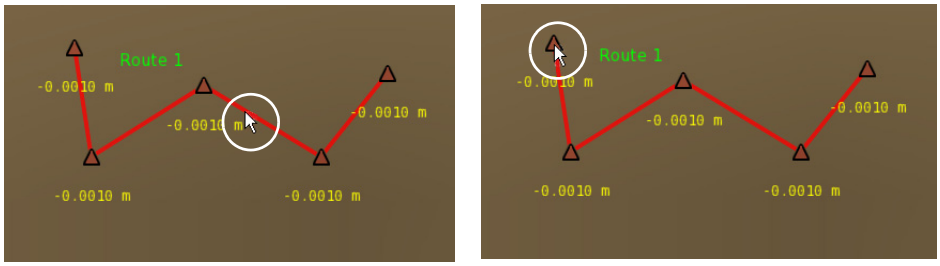


- You can only move objects simulated by VR-Forces.
- You cannot move crowds.

16.7.1. 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 16-10).
  - Areas attach to the cursor at the center of mass.



Double-click drag Move command drag

Figure 16-10. 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 16-11).

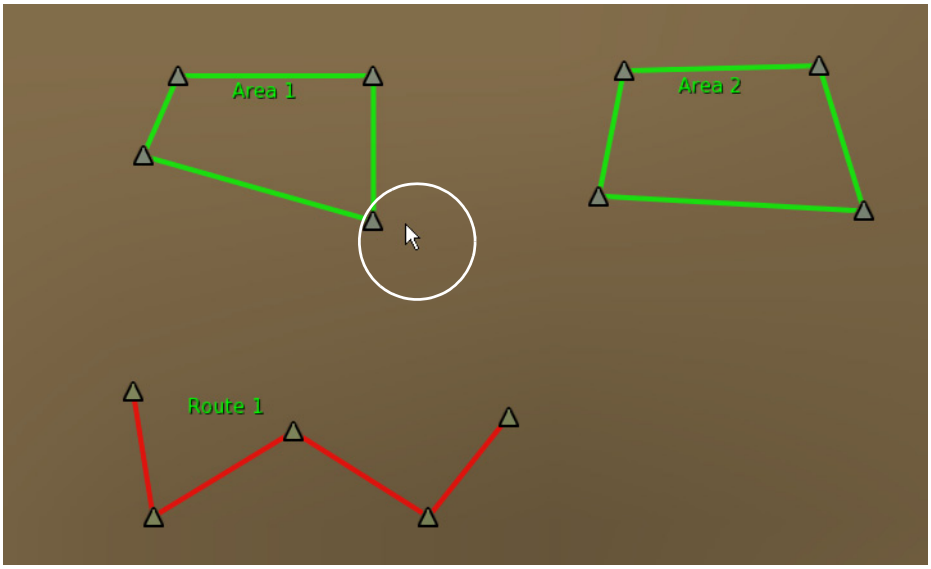


Figure 16-11. Moving multiple objects

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.

16.8. 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.

16.8.1. Copying Objects

To copy objects:

1. Select the objects you want to copy.
2. Choose **Objects** → **Copy**.

16.8.2. 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 16-23.

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.

16.8.3. 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.

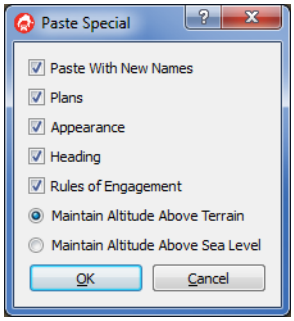


Figure 16-12. Paste Special dialog box

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 16-13 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.

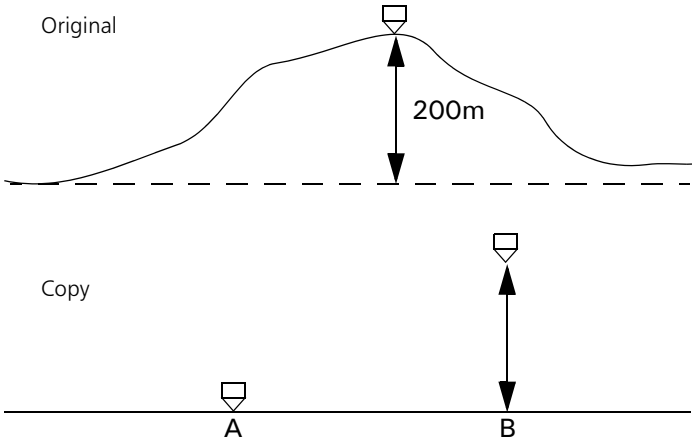


Figure 16-13. Pasting altitude

16.9. 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 tab.

The Simulation Objects Palette has a category for SOGs (Object Group), but like other simulation objects, they can be in multiple categories. 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 64-41.

16.10. 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 are in the Building and Cultural Feature categories on the Create menu and the Simulation Objects Palette. You add them to a scenario the same way you would add a simulation object. Cultural features are listed in 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 order of battle file.

16.11. Creating Props

Props are terrain objects that you can select independently of the terrain. "Extracting Props from a Terrain Patch," on page 50-24 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.
- 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 59-13.

16.12. Adding an Object to the Favorites List

As mentioned in "Selecting the Object to Create," on page 16-4, 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.

The favorites list is specific to a simulation model set. It is saved to *./data/simulation-ModelSets/<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. Select the category for the object you want to add to the Favorites list.
3. Click the star to the left of the object's name.



17. Selecting Objects

This chapter describes ways to select simulation objects and tactical graphics.

The Objects List Panel 17-2
Selecting Simulation Objects, Tactical Graphics, and Props 17-4
Selecting Objects in the Window 17-5
Selecting Objects in the Objects List Panel 17-6
Selecting a Simulation Object in the Object Console Summary Panel .. 17-6
Unselecting Objects 17-7
Using Selection Groups 17-8
Creating a Selection Group 17-8
Selecting a Selection Group 17-10
Adding Objects to a Selection Group 17-10
Removing Objects from Selection Groups 17-11
Renaming a Selection Group 17-11
Deleting a Selection Group 17-11

17.1. 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:

- Simulation object list: displays simulation objects in the simulation (Figure 17-1).
- Environmentals list: lists tactical graphics (Figure 17-1).
- Echelon view: displays the simulation object hierarchy (Figure 17-2).
- Embarkation view: displays the embarkation relationships of simulation objects (Figure 17-2).
- Overlays view: lists overlays and the objects that are on them (Figure 17-3).
- Selection groups view: lists selection groups and their members (Figure 17-3).

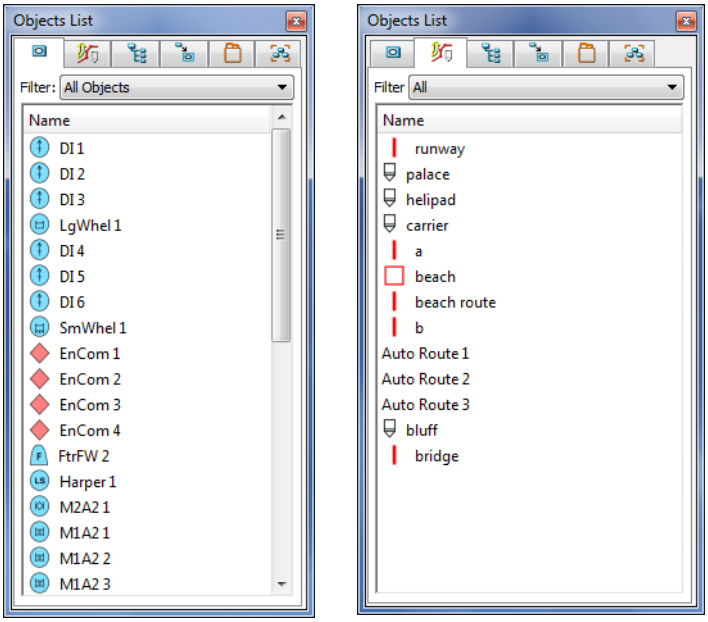


Figure 17-1. Objects List Panel – Simulation Objects list and Environmentals list

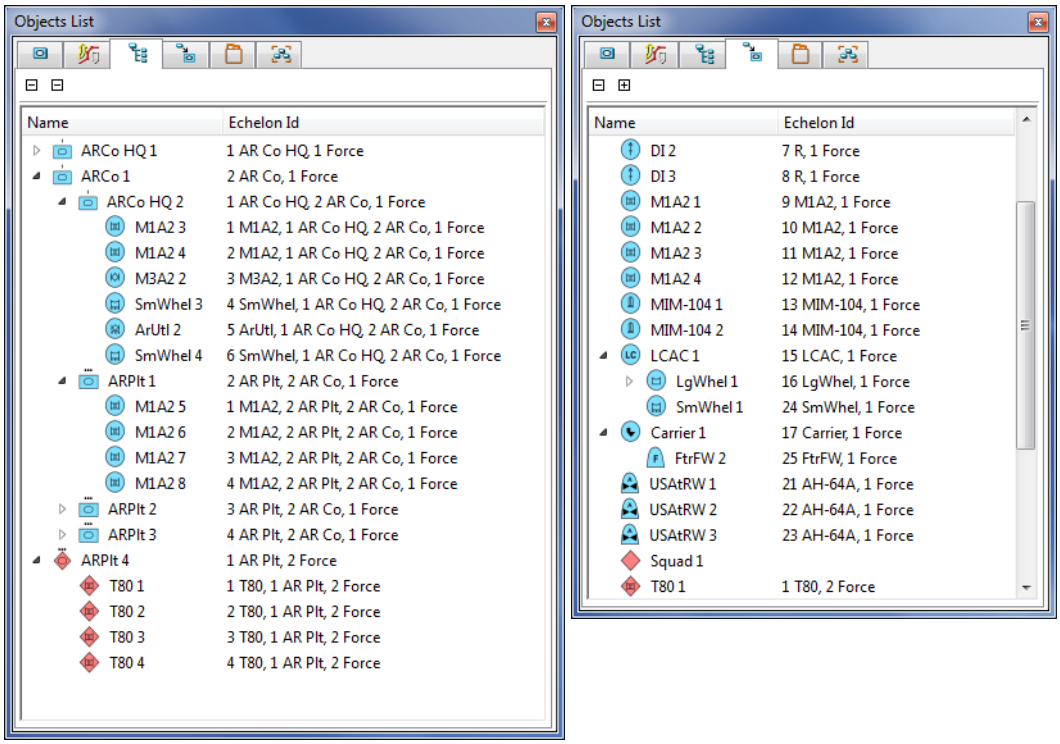


Figure 17-2. Objects List Panel – Echelon View and Embarkation View

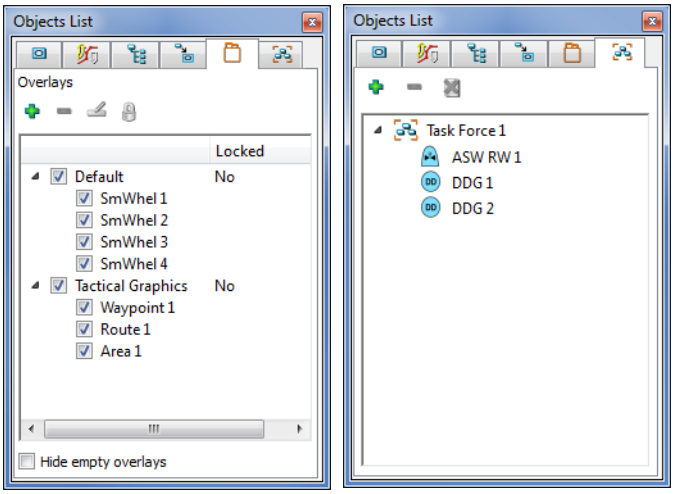


Figure 17-3. Objects List Panel – Overlays view and Selection Groups view

17.2. 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 in the Building and Cultural Feature categories on the Simulation Objects Palette.
- Tactical graphics. Control objects and the points, lines, areas, text, and symbols on the Tactical Graphics Palette, Hazards/Obstacles 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 17-1). 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 17-4).

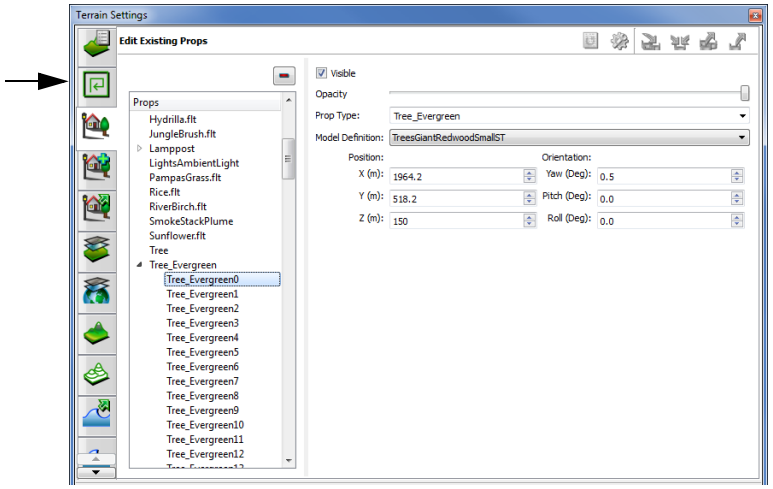
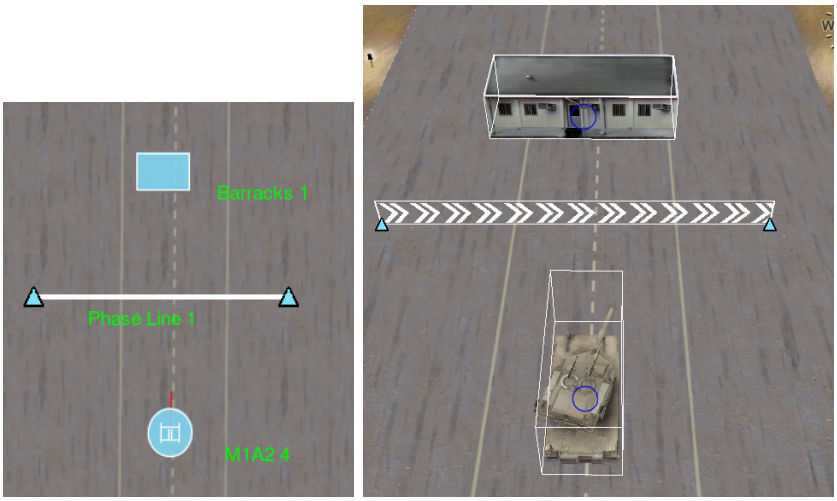


Figure 17-4. Terrain Settings dialog box, Props page

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 17-5). Tactical graphics change to white lines.



2D 3D

Figure 17-5. Selected objects

17.2.1. 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.

17.2.2. 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 17-1).
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 17-6.) 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 Entities List and the Environmental List.

➤ To filter the display of objects, select a category to view from the list at the top of the tab (Figure 17-1).

17.2.3. Selecting a Simulation Object in the Object Console Summary Panel

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 popup menu, choose Select Entity.

17.2.4. Unselecting Objects

To unselect objects in the window, choose one of the options in Table 17-1.

Table 17-1: 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 17-2.

Table 17-2: 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.

17.3. 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 Entity List 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 tabs, 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.

17.3.1. 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 16-25.

To create a selection group in the Selection Groups Panel:

1. On the Objects List Panel, select the Selections Groups view (Figure 17-6).

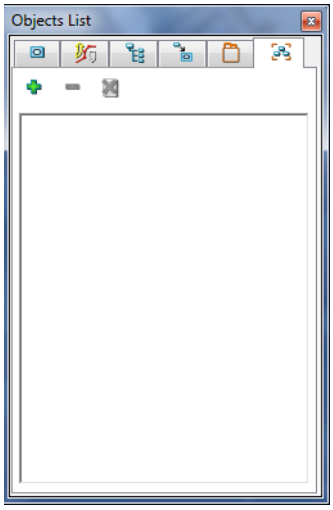


Figure 17-6. Selection Groups tab

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 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.)

17.3.2. 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.)

17.3.3. 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. Do one of the following:
   - Choose **Objects** → **Selection Groups** → **Add To Selection Group** → *group_name*, where *group_name* is one of the groups listed on the pull right menu.
   - On the Selection Groups view, click the Add Current to Selection Group button ().

17.3.4. 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 ().

17.3.5. 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`**.

17.3.6. 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 ().



18. Viewing Information about Objects

This chapter describes how to get information about simulation objects.

Displaying Information About an Object 18-3
Viewing Information for Multiple Objects 18-10
Displaying Entity Labels 18-10
Displaying Text Labels for 3D Models 18-13
Turning Entity Labels On and Off 18-14
Pinning 3D Text Labels to the Window 18-14
Displaying Labels for 2D Icons 18-15
Configuring Shadowing and Background Color for 2D Labels 18-17
Using Extended Labels 18-18
Adding an Extended Label 18-19
Changing an Extended Label's Index 18-20
Deleting an Extended Label 18-20
Simulation Object Icons 18-21
Changing the Size of 2D Icons 18-22
Rotating 2D Icons to a Simulation Object's Heading 18-23
Displaying HAT Lines for Entities (3D Only) 18-24
Displaying Threat and Sensor Range Rings 18-25
Enabling and Disabling the Display of Range Rings 18-27
Pinning Range Rings for a Simulation Object 18-27
Configuring the Display of Range Rings 18-28
Displaying Entity Resources 18-30
Viewing Numerical Resource Data 18-30
Displaying Bounding Volumes 18-31
Showing Bounding Volumes Only for Selected Simulation Objects 18-32

Configuring Object Console Messages 18-32
Setting the Notification Level for Console Messages 18-33
Viewing All Console Messages 18-34
Answering Questions from Scripted Tasks 18-35
Saving Console Messages 18-35
Displaying an Object Console Warning Icon 18-36
Clearing the Object Console 18-36
Viewing Object Counts 18-37

18.1. 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 18-10.

An information dialog box has the following sections:

- Summary.

  Displays the following information for simulation objects in entity-level scenarios (Figure 18-3):

  – Simulation object icon.
  – Simulation object name.
  – Current task.
  – Sensor information.
  – Speed.
  – Altitude.
  – Health and weapon status.

  Displays the following information for simulation objects in aggregate-level scenarios (Figure 18-4):

  – Simulation object icon.
  – Simulation object name.
  – Current task.
  – Sensor information.
  – Posture.
  – Health status.
  – Health and weapon status gumballs.
- Detailed information.
- Object console.

The summary section for units in aggregate-level scenarios (Figure 18-5) includes gumball icons that display information about the health of the simulation object. Simulation object icons also display gumballs.

Figure 18-1 describes the status represented by different gumball states.

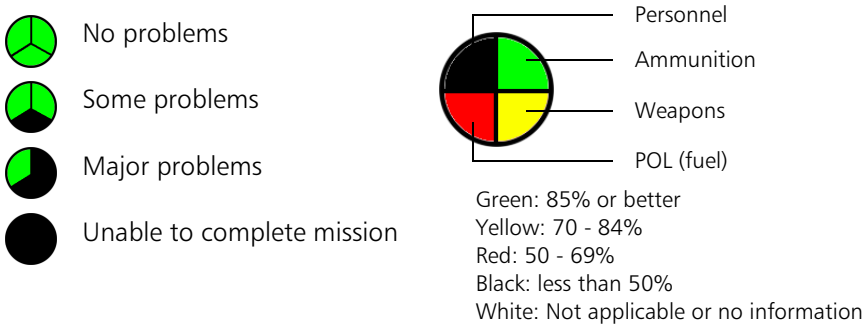


Figure 18-1. Gumball charts

Figure 18-2 illustrates gumball charts for a simulation object icon.



Figure 18-2. 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.

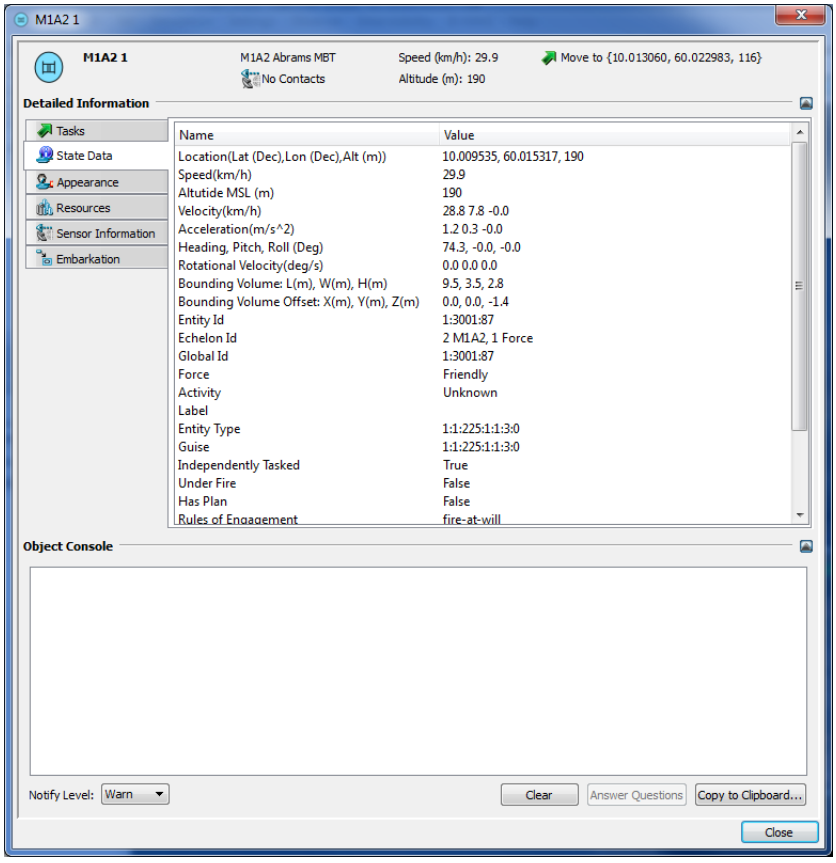


Figure 18-3. Information dialog box for simulation object in entity-level scenario

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, firepower, 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 18-4) has the following pages:

- Tasks.
- State Data.
- IFF.
- Subordinates. Lists the subordinates.
- Acoustics.
- Embarkation.

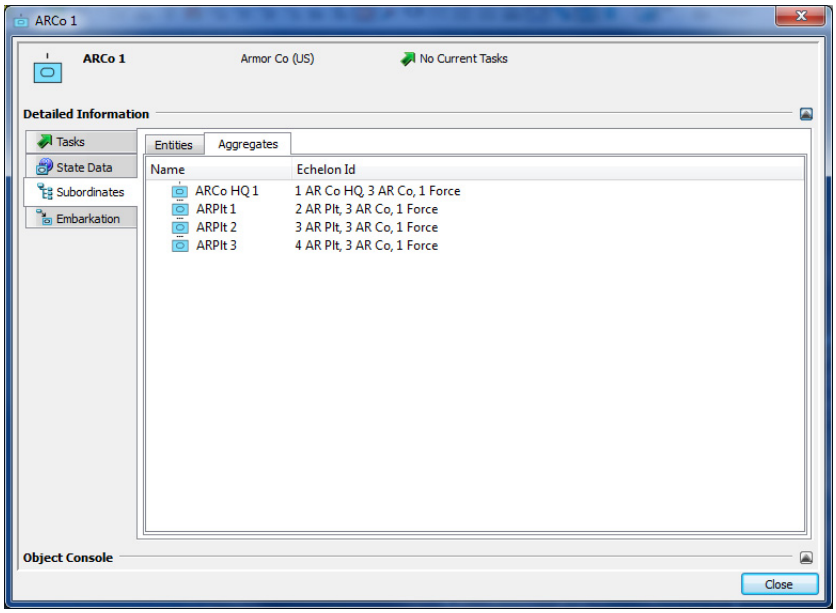


Figure 18-4. Information dialog box for units in entity-level scenarios

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 Suppliers 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.

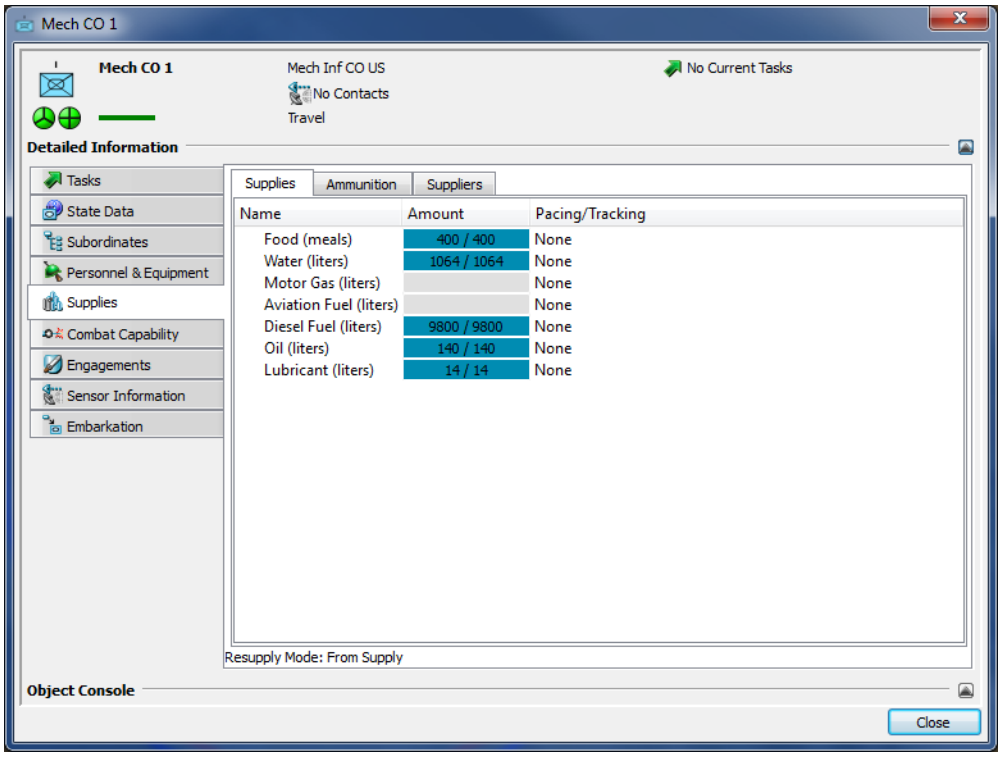


Figure 18-5. Information dialog box for simulation object in aggregate-level scenario

The detailed information section for an environmental object (Figure 18-6), 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.

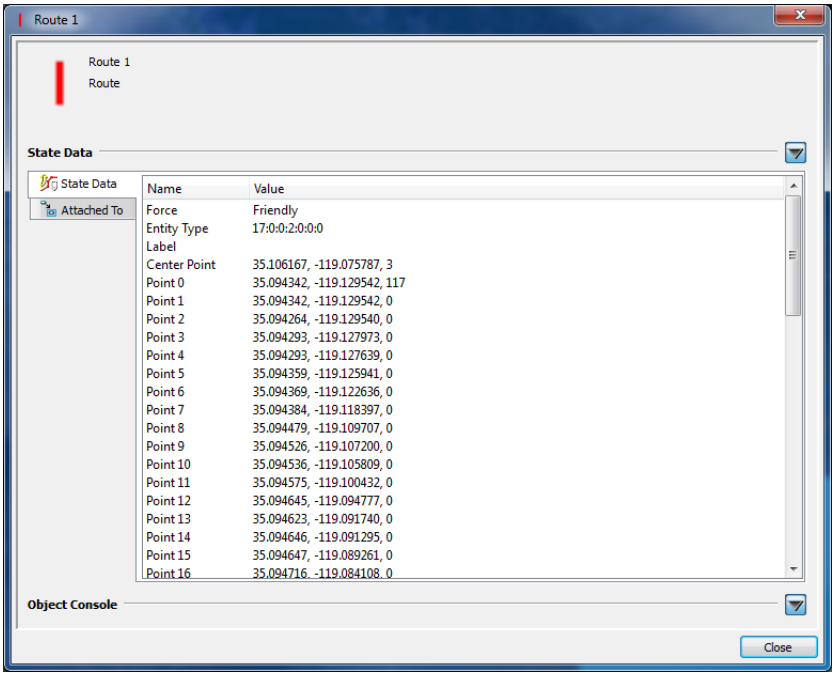


Figure 18-6. Environmental object information dialog box

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 popup menu.
- Click any field in the summary panel to open the relevant dialog box page.
- Click the Information button (🛈) 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.

18.1.1. 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 18-7).

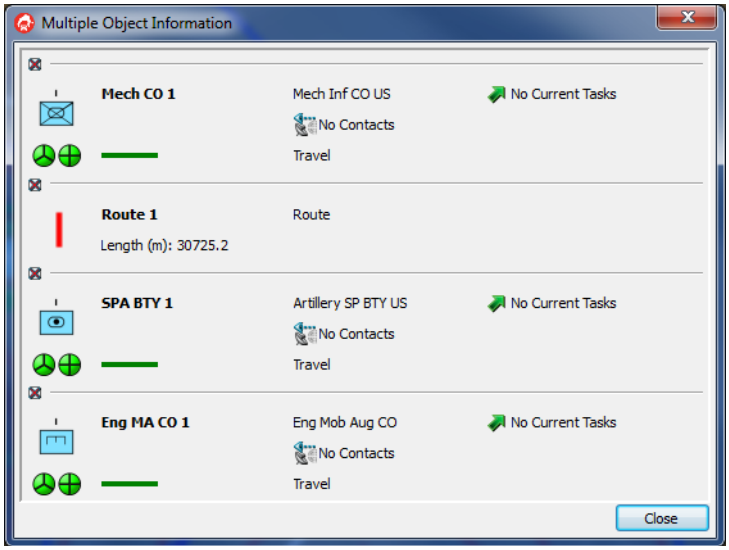


Figure 18-7. Multiple Object Information window

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.

18.2. Displaying Entity Labels

VR-Forces provides extensive information about simulation objects in the Information dialog box ("Displaying Information About an Object," on page 18-3). 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 18-8). 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 57-8.

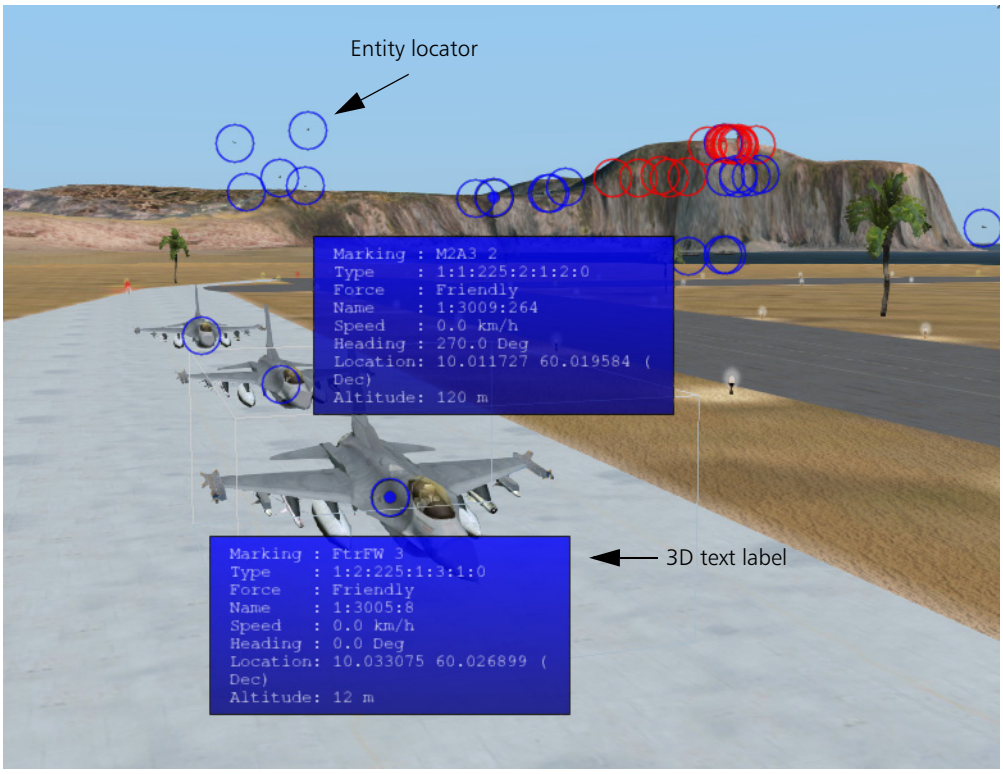


Figure 18-8. Entity labels for 3D models

For 2D icons, labels are a set of individually enabled text labels, called symbol decorations (Figure 18-9). 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.

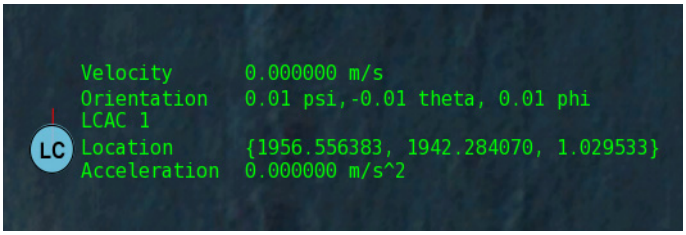


Figure 18-9. Label for a 2D icon

If spot reports are enabled, you can display labels for spot reports. For details, please see "Displaying Labels for Spot Reports," on page 9-9.

18.2.1. 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 18-10). Then select the entity whose label you want to see.

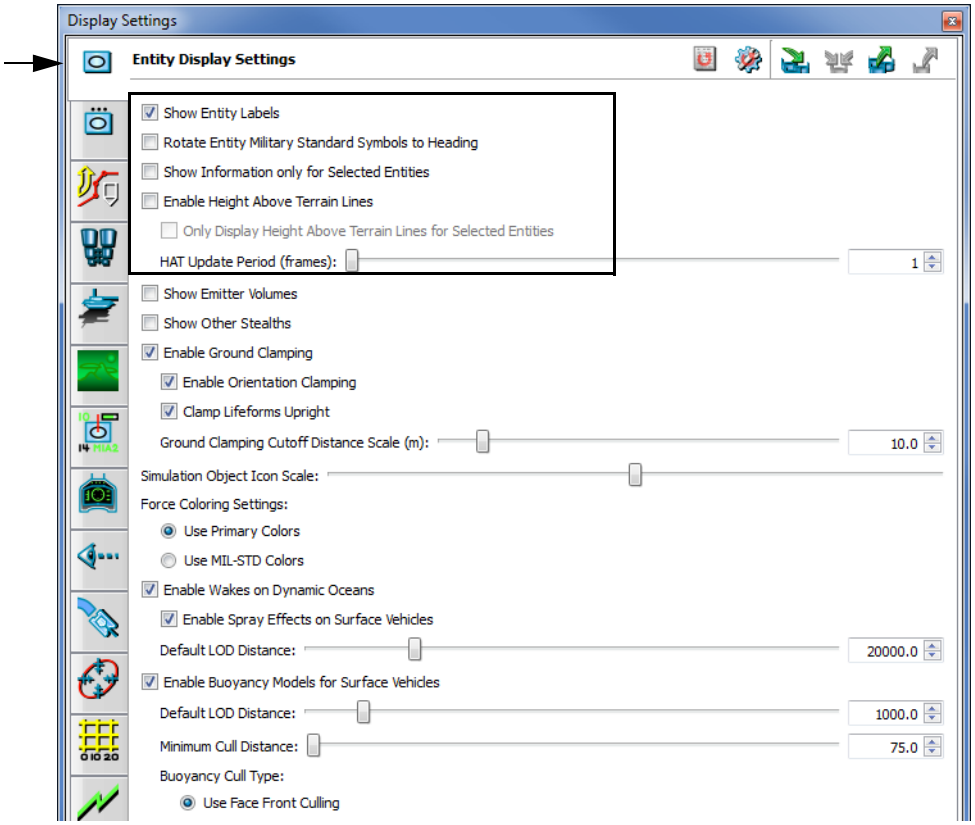


Figure 18-10. Display Settings, Entity Display Settings page

18.2.2. 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 18-10).
3. Select or clear the Show Entity Labels check box.

> You can also enable and disable entity labels by choosing **Settings** → **Entity Labels** or by clicking the Entity Labels button ( ) on the Display Settings Toolbar.

18.2.3. 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 18-10). Then, in the Objects List Panel, multi-select the entities whose labels you want to see.

18.2.4. 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 18-11).

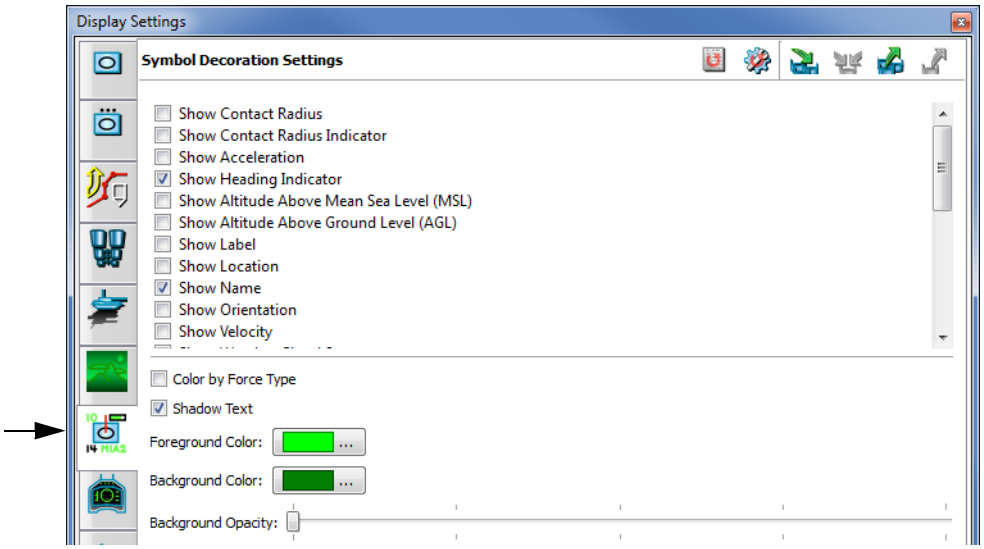


Figure 18-11. Symbol Decoration Settings page

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.

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.

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.

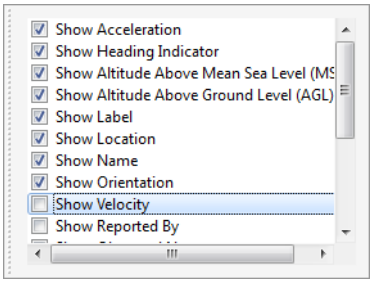


Figure 18-12. Symbol Decoration Panel

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.

18.2.5. 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.

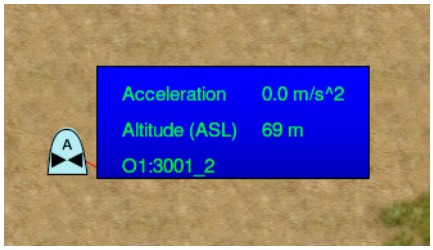


Figure 18-13. 2D label with background

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 18-11).
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 18-11).
3. Select or clear the Shadow Text check box.

18.3. Using Extended Labels

One of the ways to identify a simulation object is by its label. (For more information, please see "Labels," on page 15-4.) 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 18-14 shows an extended label (called Extended Label) added in the Application Settings dialog box, Session Options page. Figure 18-15 shows the text for the extended label for a specific entity entered in the Edit dialog box and then displayed on the screen.

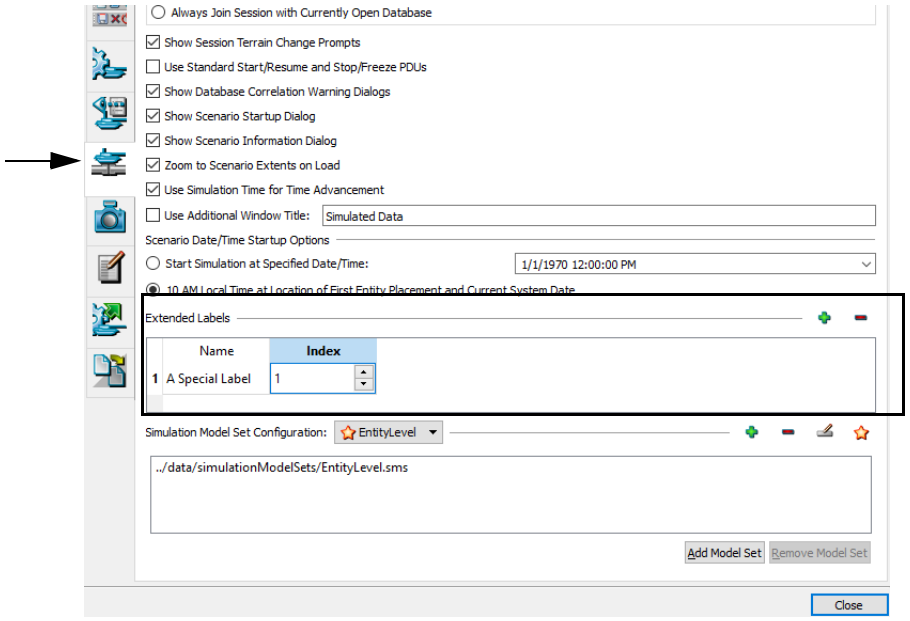


Figure 18-14. Extended label added to session

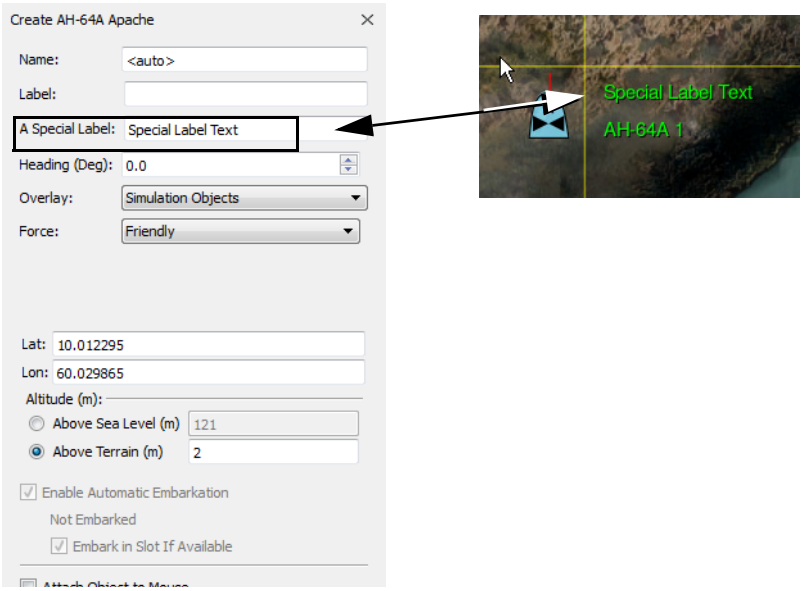


Figure 18-15. Extended label text for a simulation object

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 18-15.

18.3.1. Adding an Extended Label

To add an extended label:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the Session Options page (Figure 18-14).
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.

18.3.2. 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 Session Options page (Figure 18-14).
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.

18.3.3. 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 Session Options page (Figure 18-14).
3. In the Extended Labels window, select the label you want to delete.
4. Click the Delete button ( ).

18.4. Simulation Object Icons

The 2D Icons model set uses MIL-STD 2525B icons. Figure 18-16 illustrates basic icon shapes for friendly forces.

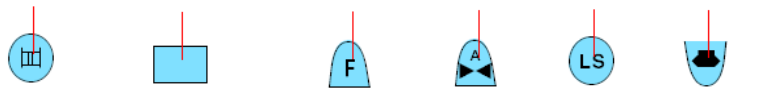


Figure 18-16. MILSTD-2525B icons

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 18-17.

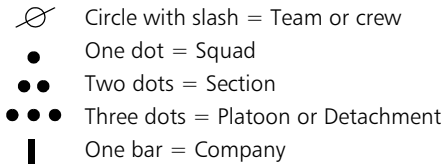


Figure 18-17. Unit echelon indicators

18.4.1. 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 18-18).

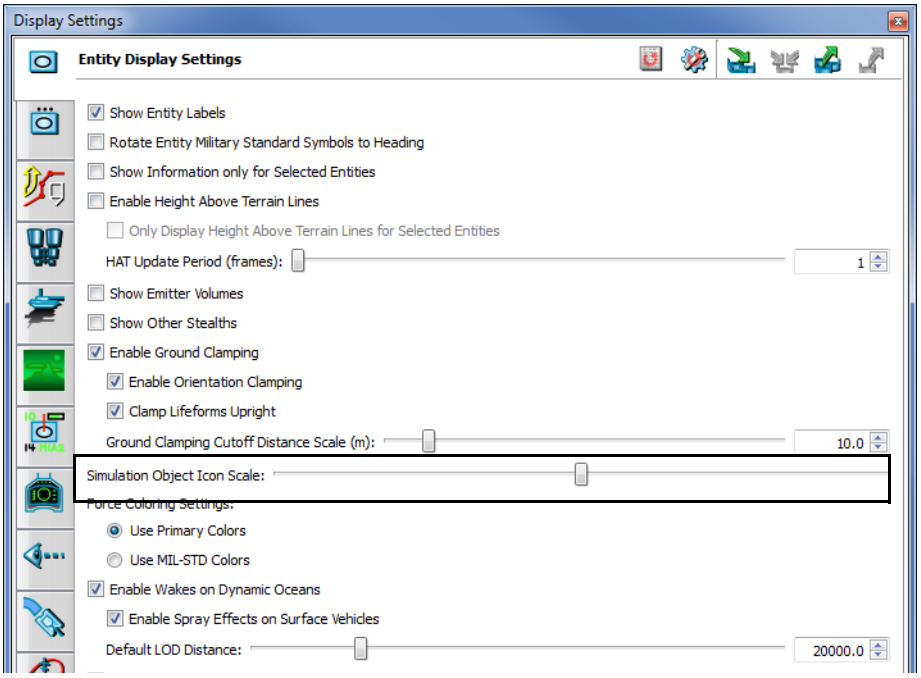


Figure 18-18. Entity Display Settings page

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 Entity Icon Scale Factor slider opens.
2. Adjust the slider to change the size of the icons.

The Entity Icon Scale Factor slider is a dockable widget that you can undock and leave visible as long as icon scaling is enabled.

18.4.2. 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.



Figure 18-19. 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 18-18).
3. Select the Rotate Entity Military Standard Symbols to Heading check box.

18.5. 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 18-20). 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.

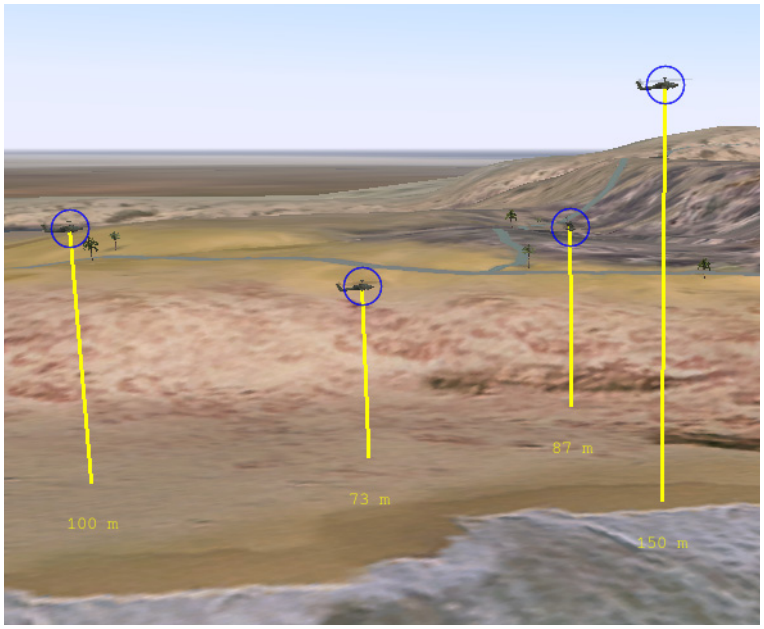


Figure 18-20. Entity height-above-terrain lines

! 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 18-10).
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.

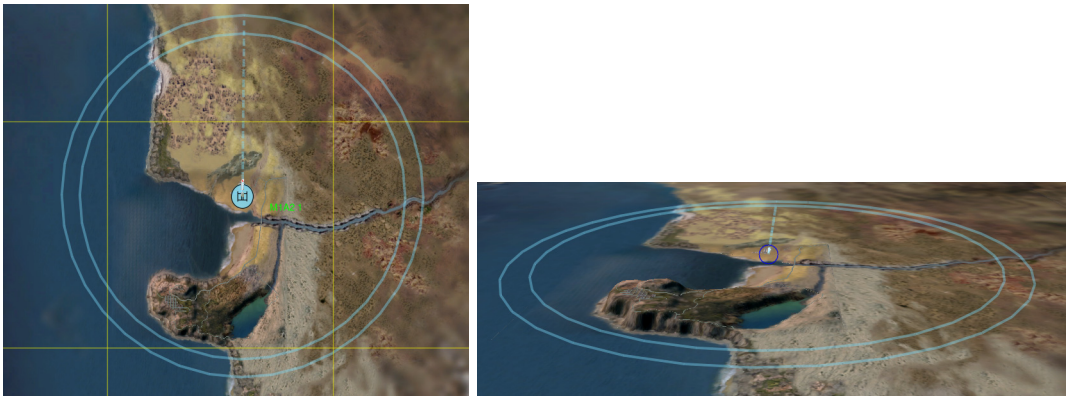


You can also enable and disable entity HAT lines by choosing **Settings** → **Entity Height Above Terrain Lines** or by clicking the Entity Height Above Terrain Lines button () on the Display Settings Toolbar.

18.6. 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 18-21). A ring is displayed for each weapon system that the entity has. For example, in Figure 18-21, 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 (Figure 18-22).



2D 3D

Figure 18-21. 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 18-22). Unit range rings have breaks to show the front, rear, left flank, and right flank sectors.

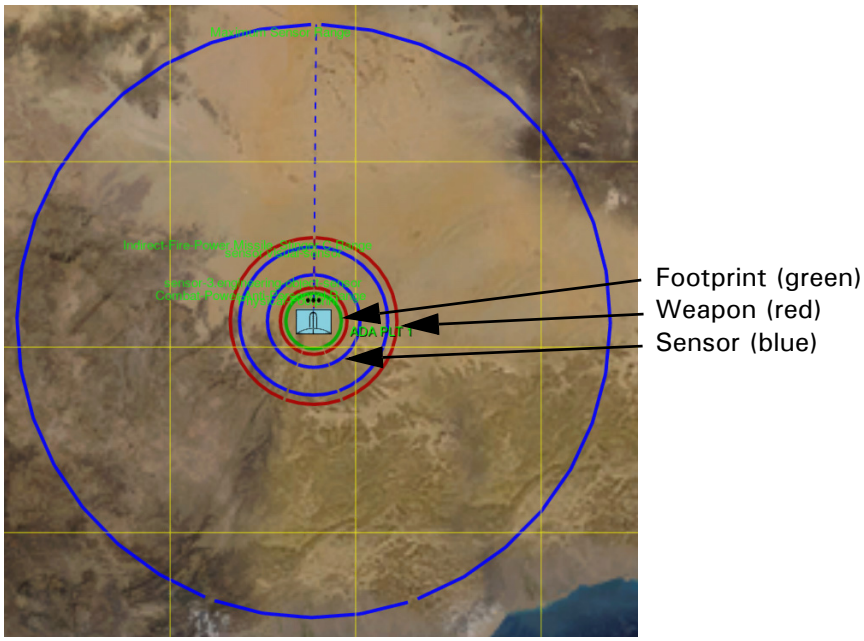


Figure 18-22. Range rings (aggregate-level scenario)

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 18-23).

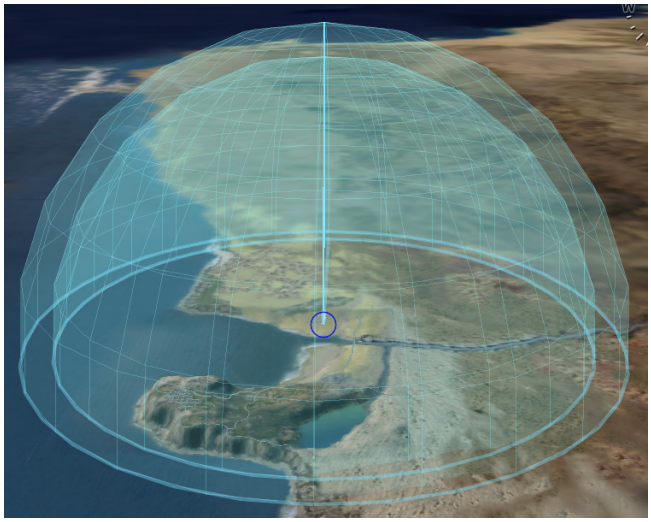


Figure 18-23. Range ring sphere



On small databases, a simulation object's range ring may extend beyond the edge of the terrain display.

18.6.1. 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 19-9).
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.

18.6.2. 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 18-27.
2. Select a simulation object.
3. Choose **Objects** → **Pin Range Rings**.

18.6.3. 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 18-24). A label can display the entity name, weapon type, and range of the weapon.

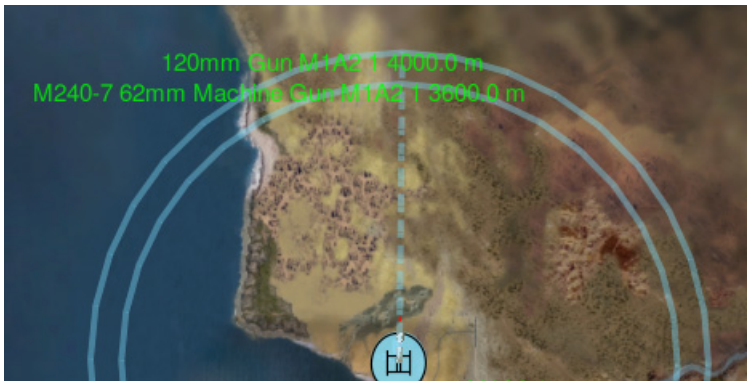


Figure 18-24. Range ring labels

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 18-25).
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.

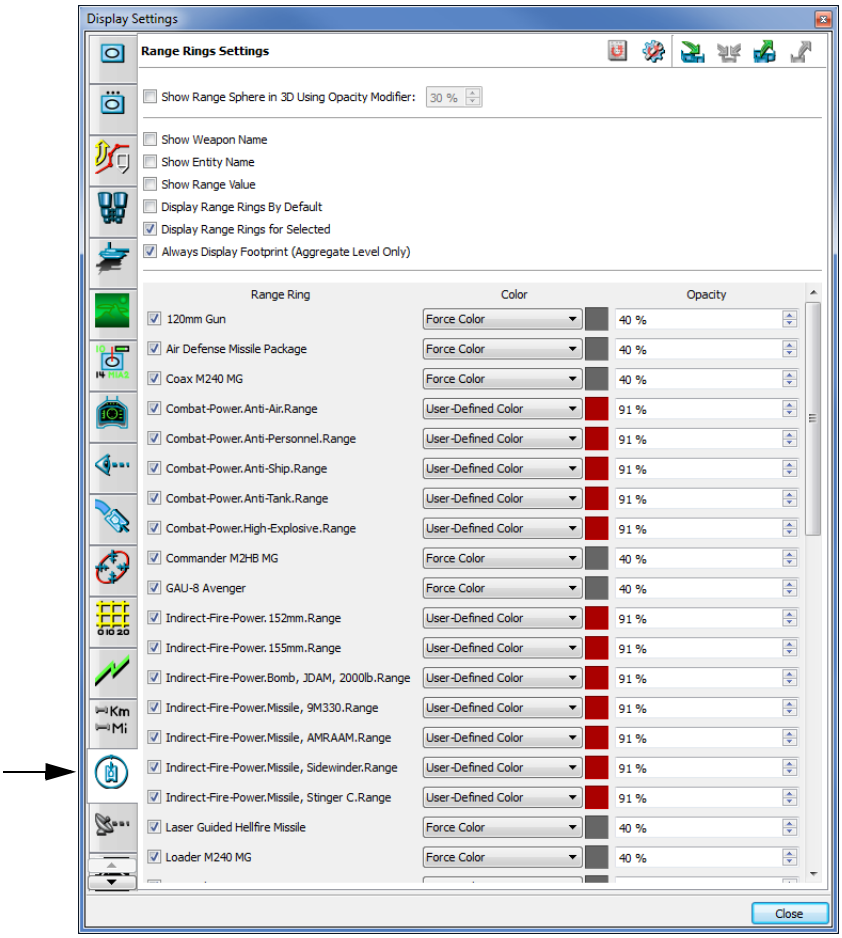


Figure 18-25. Range Rings Settings page (*AggregateLevel.sms*)

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.

18.7. 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.

18.7.1. 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 18-26).

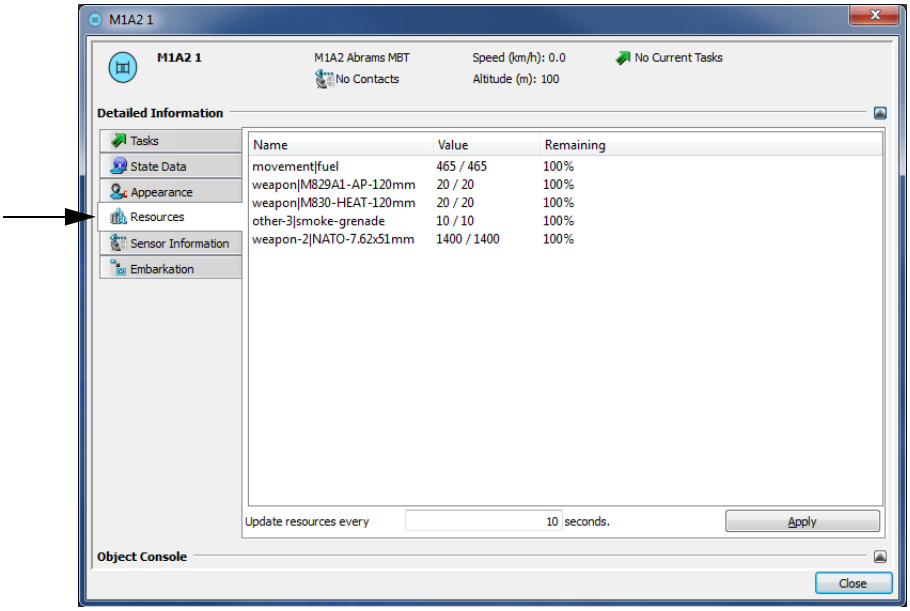
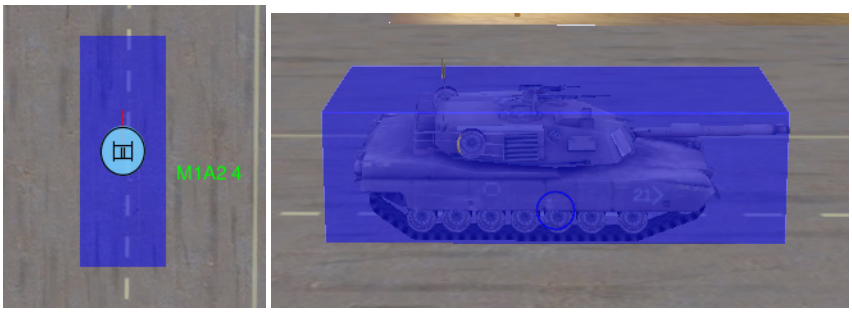


Figure 18-26. Resource Information page

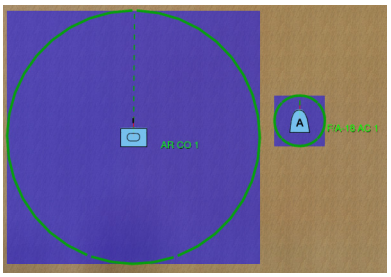
➤ To view a simulation object's resource status, on the Information dialog box, select the Resources page.

18.8. 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 18-27). For simulation objects in aggregate-level scenarios, the bounding volume is equivalent to the footprint.



2D 3D



Aggregate-level bounding volumes

Figure 18-27. Bounding volumes

To enable or disable bounding volumes:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 19-9).
3. Select the observer mode for which you want to enable or disable bounding volumes.
4. Select or clear the Bounding Volume check box.

18.8.1. Showing Bounding Volumes Only for Selected Simulation Objects

You can configure VR-Forces so that it shows bounding volumes only for selected simulation objects.

To display bounding volumes only for selected simulation objects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 18-18).
3. Select the observer mode for which you want to enable bounding volumes.
4. Select the Bounding Volume check box.
5. Select the Entity Display Settings page (Figure 18-10).
6. Select the Show Bounding Volumes for Selected Entities check box.

18.9. 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 scripted tasks. 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.

18.9.1. 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 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 30-28.)

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 18-28).

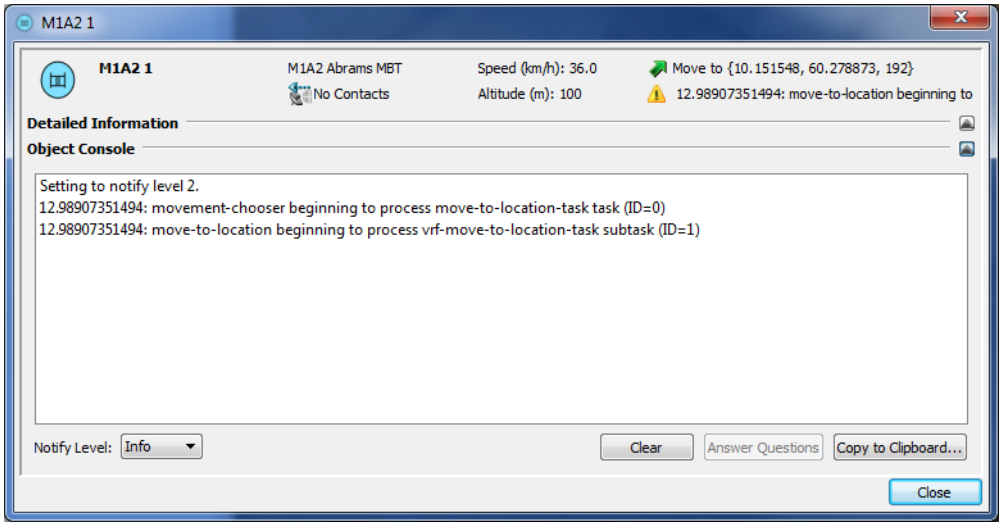


Figure 18-28. Information dialog box, Object Console

4. Select a notification level from the Notify Level list.

18.9.2. 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 using 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.

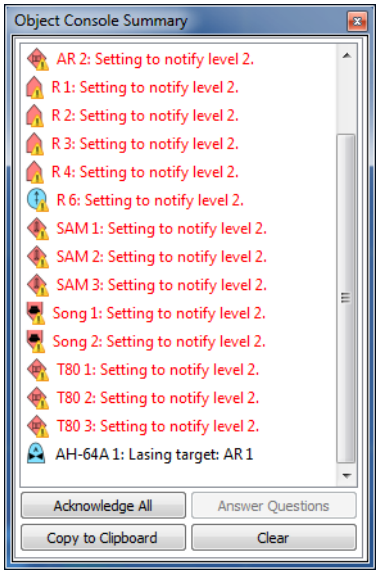


Figure 18-29. Object Console Summary Panel

➤ 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.

18.9.3. 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 18-30). 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.

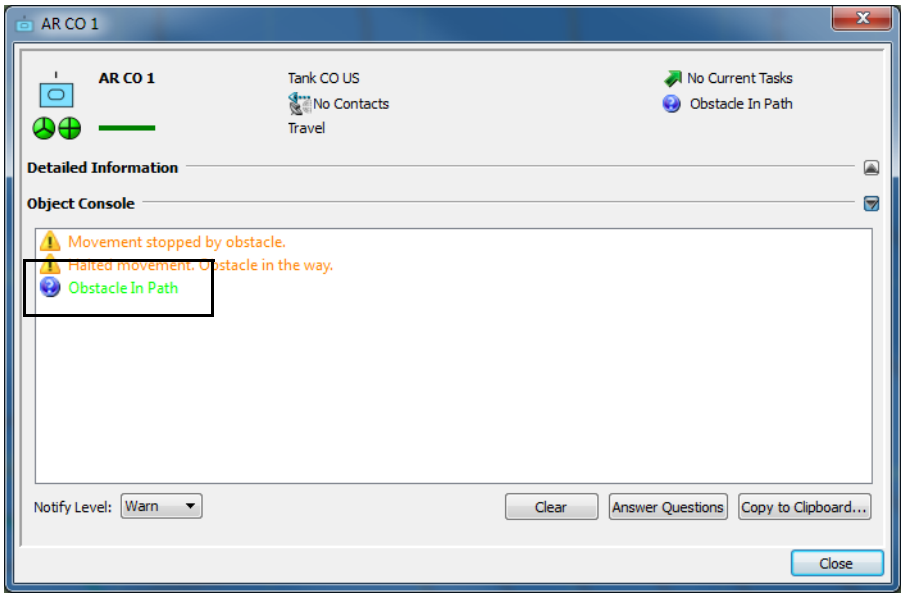


Figure 18-30. Question in console

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.

18.9.4. 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 18-28).
3. To save all messages in the console, click Copy to Clipboard. To save specific messages, select them and click Copy to Clipboard.
4. Paste the text into a the text editor or word processor of your choice.

18.9.5. Displaying an Object Console Warning Icon

VR-Forces can display a notification icon when an object receives a message on its object console (Figure 18-31). 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 18-33.



Figure 18-31. Object console warning icon

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.

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 18-10).
3. Select or clear the Show Object Console Warning Icon check box.

18.9.6. Clearing the Object Console

➤ To clear the messages from the object console, click Clear.

18.10. 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 18-32).

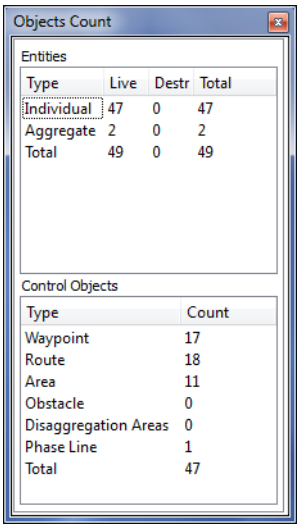


Figure 18-32. Objects Count Panel



19. Creating Simulation Objects

This chapter explains how to create and manipulate simulation objects in general, and entities in particular. For information about working with units, please see Chapter 23, *Creating and Controlling Units*.

- Creating Simulation Objects 19-3
  - Default Placement of New Entities 19-3
  - Placing Entities Inside Buildings 19-4
  - Placing Simulation Objects on Other Simulation Objects 19-4
  - Simulation Object Resources 19-4
  - Deleting Simulation Objects 19-5
- Editing Simulation Objects 19-5
- Embarking and Disembarking Simulation Objects 19-7
  - Embarking and Disembarking Simulation Objects Instantly 19-8
  - Disembarking Simulation Objects Using the Disembark Command .. 19-11
- Embedded Entities 19-12
  - Configuring an Entity to Support Embedded Entities 19-12
  - Deploying Embedded Entities 19-15
  - Deploying an Entity with a Task 19-15
  - Recovering Deployed Entities 19-16
  - Assigning Tasks and Plans to Embedded Entities 19-16
  - Restoring an Entity that has Embedded Entities 19-16
- Collision, Obstacle, and Feature Avoidance 19-17
- Entity Movement and Soil Type 19-18
- Using a Joystick to Control Entities 19-19
  - Switching Between Systems 19-20
  - Disabling Joystick Control 19-21

Using the Keyboard to Control an Entity 19-21
Using Ground Clamping 19-21
Configuring Ground Clamping 19-23
Exaggerating the Scale of 3D Entity Models 19-24
Hiding 3D Models 19-27
Changing Force Colors 19-27

19.1. Creating Simulation Objects

The basic procedure for creating a simulation object is fairly simple:

1. On the Create menu or Simulation Objects Palette, select the simulation object that you want to create.
2. Click on the terrain to place the simulation object.
3. Optionally, specify the properties of the simulation object, such as name, altitude, and heading.

The generic procedures for choosing the simulation object to create, placing it on the terrain, and setting properties are described in Chapter 16, *Creating and Placing Objects*. This section provides additional details about simulation object creation and placement.

You can also create simulation objects by copying existing simulation objects. For details, please see "Copying and Pasting Objects," on page 16-21.

You can create preconfigured units the same way you create individual entities. You can also create units by aggregating simulation objects. To learn how to create a unit, please see "Creating Units," on page 23-2.

When you create simulation objects, if spot reports are enabled and a particular force is hidden, if you create a simulation object of the hidden force, you will not be able to see it. It is recommended that you disable spot reports or set the viewpoint to show all forces when you are creating scenarios. For details about spot reports, please see "Displaying Simulation Objects Based on Spot Reports," on page 9-2.

19.1.1. 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.

19.1.2. 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 73-10.
- 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 50-29.



If you enable view constraints, it may restrict your ability to move the observer into buildings. You can disable view constraints by pressing **x**.

19.1.3. 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 19-7.

19.1.4. 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 Resource set data request to set its resources at some amount less than 100%.

19.1.5. 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 41-13.

19.2. 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 Entity dialog box opens (Figure 19-1).

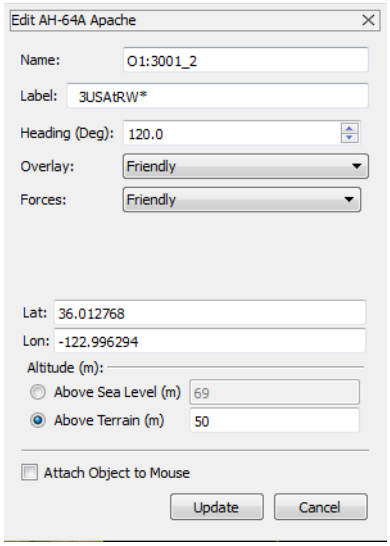


Figure 19-1. Edit Entity dialog box

3. Change the properties as desired.
4. Click OK.



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 16-15.
- "Specifying an Object's Heading Dynamically," on page 16-17.
- "Moving Objects," on page 16-19.
- Chapter 26, *Tasks for Entity-Level Scenarios*.
- Chapter 27, *Tasks for Aggregate-Level Scenarios*.
- Chapter 30, *Setting Simulation Object State*.

19.3. 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 always visible in the 3D view.

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 RPR FOM 1.0 and 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.

---

i 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 19-2).

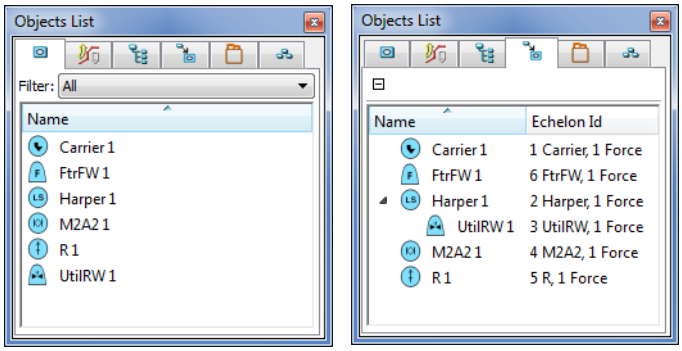


Figure 19-2. Echelon View and Embarkation View

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 26, *Tasks for Entity-Level Scenarios*.

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.

19.3.1. 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 30-16 and "Disembarked," on page 30-16.

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** → **Embark On**. The Embark On dialog box opens.

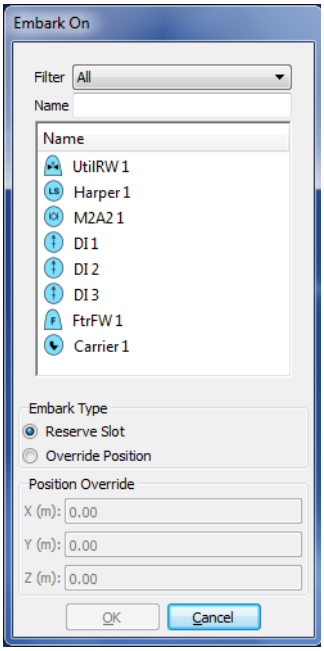


Figure 19-3. Embark On dialog box

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 35-11.

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 entity to embark on. If it supports embarkation and automatic embarkation is enabled, a green box is displayed. (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*.
5. Click to place the entity.

19.3.2. 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** → **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** → **Disembark All**.

19.4. 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.

19.4.1. 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 19-4 illustrates the properties dialog box created for the system shown in the previous paragraphs.

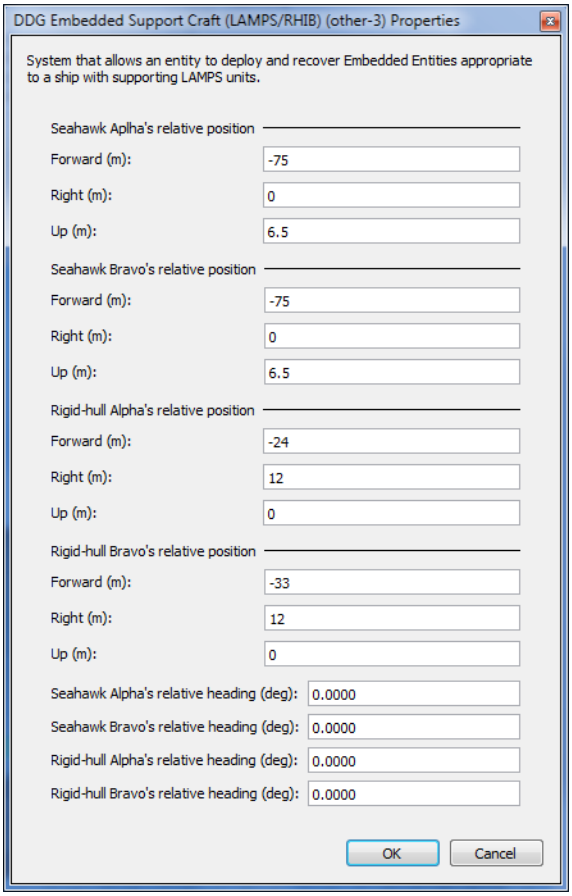


Figure 19-4. Embedded Entity Properties dialog box

19.4.2. 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 deploy an embedded entity explicitly:

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.

19.4.3. Deploying an Entity with 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 assign a task to a parent entity that requires it to deploy an 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.

19.4.4. 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 recover a deployed embedded entity:

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.

19.4.5. 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.

19.4.6. 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.

19.5. 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 70-15.

19.6. 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 19-1 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 19-1: 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

19.7. 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 6 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 72-2.

To enable joystick control:

1. Select the entity for which you want to enable joysticks.
2. Choose **Objects** → **Take Control**. The Take Control dialog box opens.

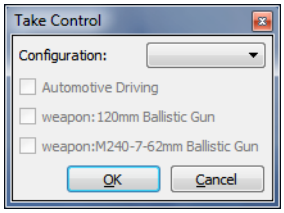


Figure 19-5. Take Control dialog box

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.

19.7.1. 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 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 72-9.

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 63, *The Simulation Object Editor and SMSs* and Chapter 64, *Editing Simulation Object Models*.)

To rename a system function group:

1. Choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.4.1** → **Tools** → **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.

19.7.2. Disabling Joystick Control

To disable joystick control:

1. Select the entity for which you want to disable joysticks.
2. Choose **Objects** → **Release Control**.

19.8. 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 19-19 and "Configuring Joysticks and Keyboard Control," on page 72-2.

19.9. 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 19-6).

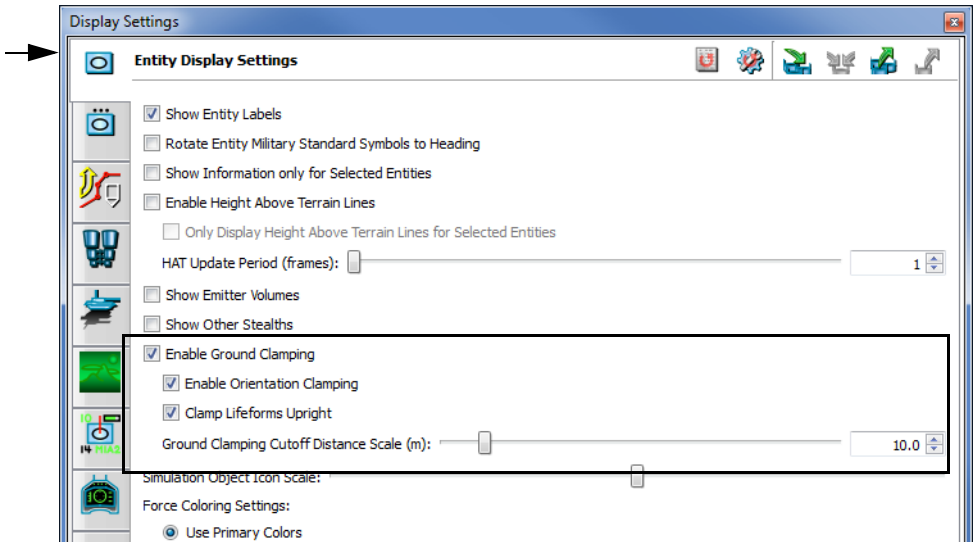


Figure 19-6. Ground clamping settings

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.

19.9.1. 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 19-6).
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.

19.10. 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 19-7 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.

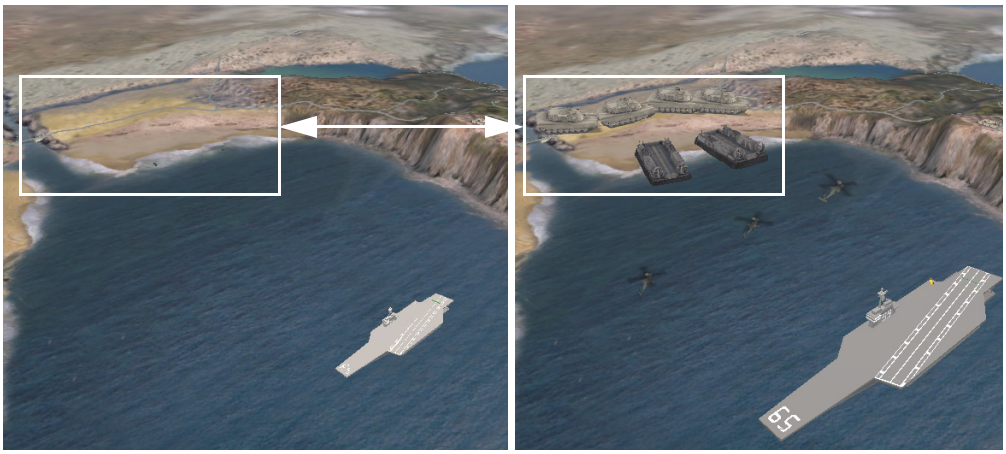


Figure 19-7. Model scaling off and on (Stealth observer mode)

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 19-8. XR mode

To scale entity models:

1. Choose **Observer** → **Advanced Settings**. The Display Settings dialog box opens to the Observer Settings page (Figure 19-9).

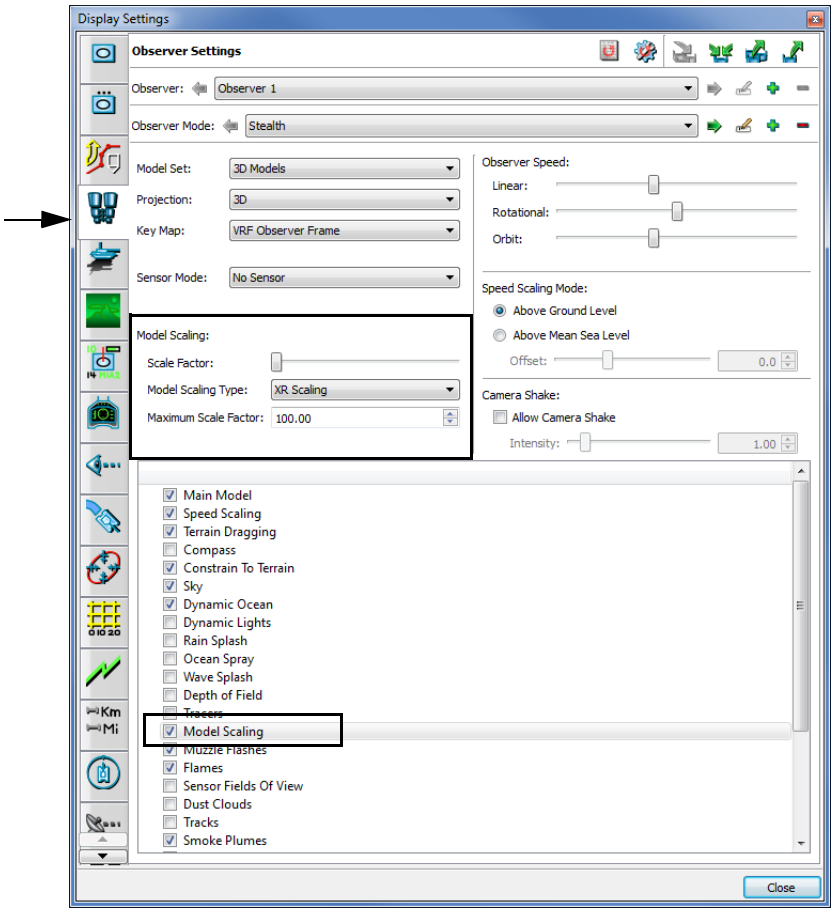


Figure 19-9. Observer Settings page

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.

19.11. 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 19-9).
3. In the list of observer settings, clear the Main Model check box.

19.12. 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).

To change force coloring:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 18-18).
3. Select the Force Coloring Settings option that you want.



20. Generating Traffic and Crowds

This chapter explains how to automatically generate pattern of life simulation object traffic and how to create and task pedestrian crowds.

Generating Traffic 20-2
Creating Spawn Points 20-2
Creating Sink Points 20-4
Spawn Patterns 20-5
How Spawn Events are Scheduled 20-6
Spawning Entities in Bursts 20-8
Using a Custom Plan 20-9
Default Spawn Patterns and Scenario-Specific Patterns 20-9
Creating a Spawn Pattern 20-10
Copying a Spawn Pattern 20-14
Editing a Spawn Pattern 20-15
Deleting a Spawn Pattern 20-16
Creating Multiple Simulation Objects and Crowds 20-16
Creating Multiple Simulation Objects 20-17
Adding an Area for Multiple Simulation Object Creation 20-20
Specifying the Simulation Objects to Create 20-20
Creating Crowds 20-21
Crowd Tasks 20-22
Pedestrian-Oriented Tactical Graphics 20-22
Using Civilian Visit Points 20-22

20.1. 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. You would typically remove the simulation object from the scenario when the plan is complete.

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.



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 (by clicking the star next to the point's name in the list of points) a duplicate entry is added to the Create menu.

20.2. Creating Spawn Points

You can create spawn points from the Create menu, the Tactical Graphics Palette, or from the Spawn Pattern Manager. Figure 20-1 illustrates a spawn point.



Figure 20-1. 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 20-10. For details about configuring console messages and warning icons, please see "Configuring Object Console Messages," on page 18-32.

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 20-2).

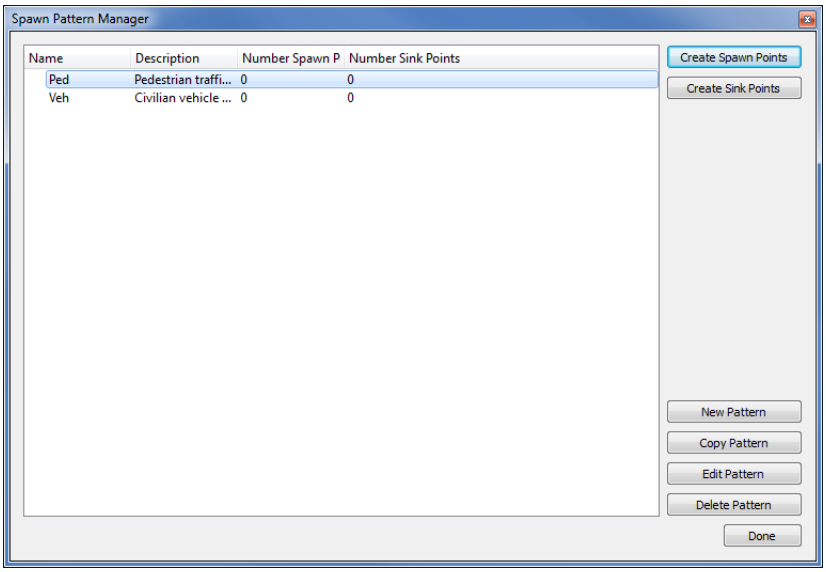


Figure 20-2. Spawn Pattern Manager

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.

20.3. 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 20-10.) You can create sink points from the Create menu, the Tactical Graphics Palette, or from the Spawn Pattern Manager. Figure 20-3 illustrates a sink point.

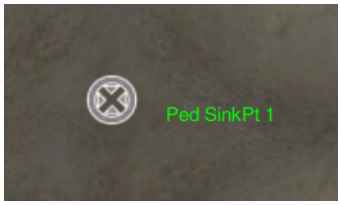


Figure 20-3. Sink point

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 20-2).
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.

20.4. 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 19-17 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.

20.4.1. 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 20-4 illustrates the difference between continuous and timed spawning.

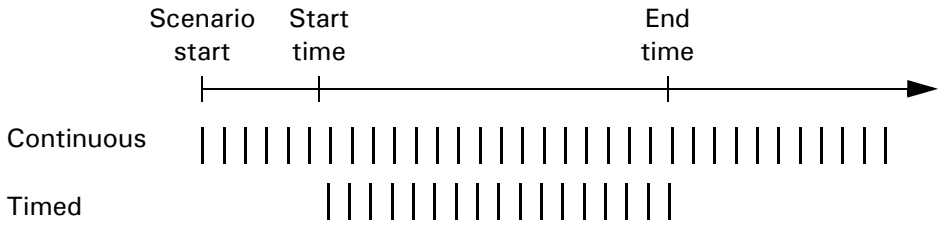


Figure 20-4. Continuous spawning compared to time-based

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, an 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 20-5 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.

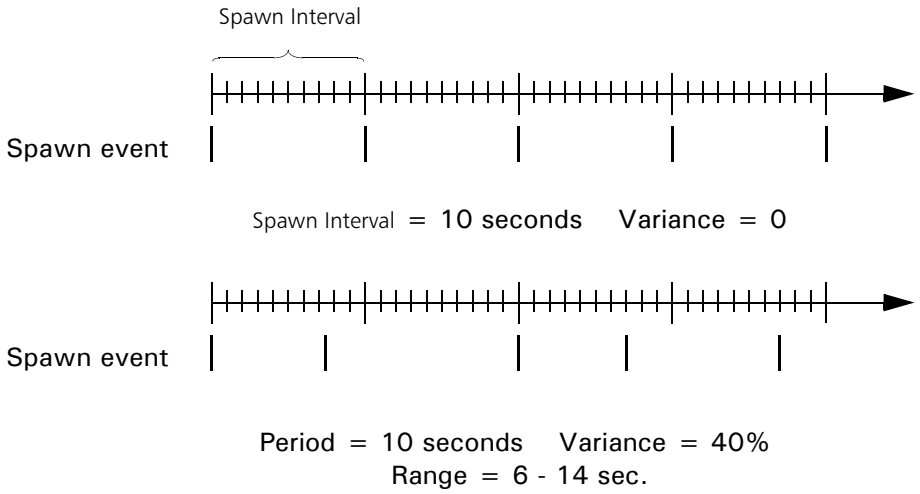


Figure 20-5. Spawn period and variance

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.

20.4.2. 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 20-6 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.

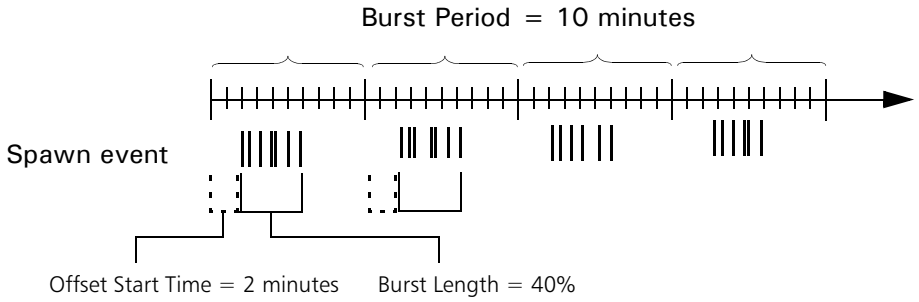


Figure 20-6. Burst periods

20.4.3. Using a Custom Plan

As noted in "Generating Traffic," on page 20-2, 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 until their plan removes them. You can delete simulation objects without needing to specify the specific simulation object with the Delete Self global command.

20.4.4. Default Spawn Patterns and Scenario-Specific Patterns

The VR-Forces includes default spawn patterns for civilian vehicles and civilian life-forms. 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*.

20.5. Creating a Spawn Pattern

Previous sections have explained the various components of a spawn pattern. This section explains how to create one.

To create a spawn pattern:

1. Choose **Simulation** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 20-2).
2. Click New Pattern. The Create Spawn Pattern dialog box opens (Figure 20-7).

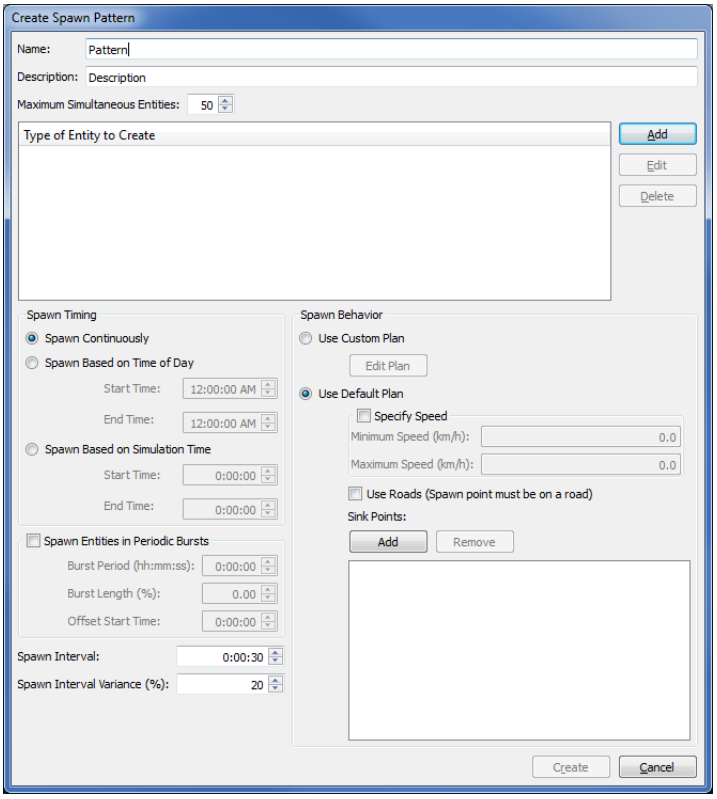


Figure 20-7. Create Spawn Pattern dialog box

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 an 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. This dialog box lists simulation objects by category.
   b. Select the category for the simulation object type you want to add.
   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.

---

i 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. Choose an option in the Spawn Timing group box, as follows:
   – Spawn Continuously. Entities are spawned through 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.

8. Optionally, select Spawn Entities in Periodic Bursts. (For an explanation of periodic bursts, please see "Spawning Entities in Bursts," on page 20-8.) 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.
9. Specify the spawn interval. This specifies the amount of time between each spawn event, modified by the spawn interval variance.
10. Specify the spawn interval variance as a percentage of the spawn interval.
11. 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.

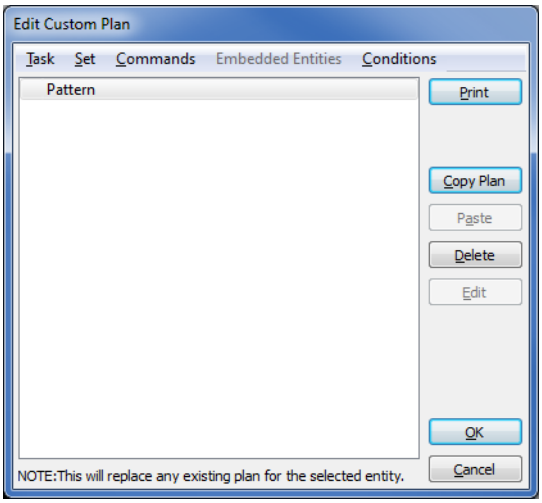


Figure 20-8. Edit Custom Plan dialog box

      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 32, *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 an 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.

12. Click Create.

20.5.1. 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 20-2).
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 20-10.
5. Click Create.

20.6. 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 20-2).
2. Select the pattern that you want to edit.
3. Click Edit Pattern. The Edit Spawn Pattern dialog box opens (Figure 20-9).

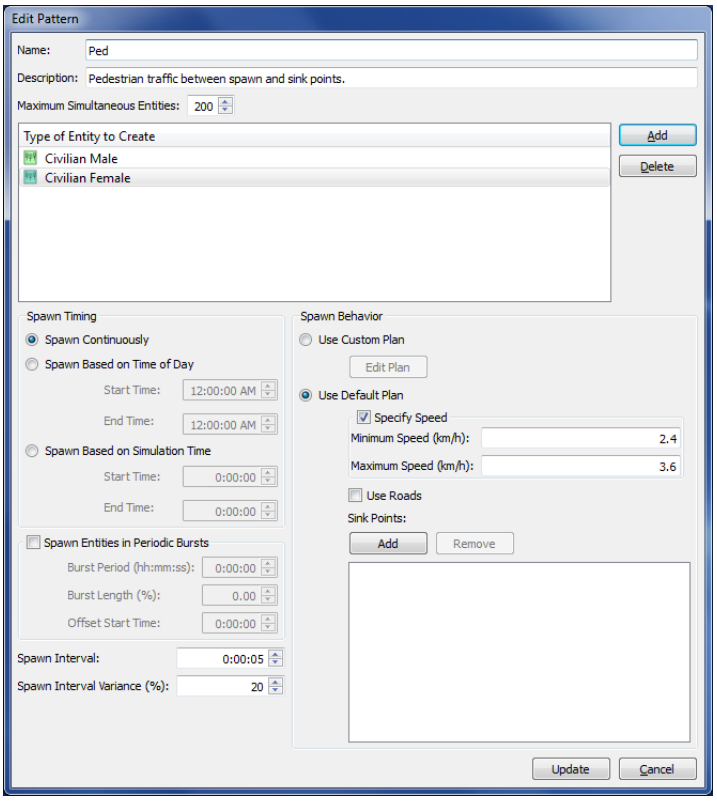


Figure 20-9. Edit Spawn Pattern dialog box

4. Change any value as desired. The meaning of each option is described in "Creating a Spawn Pattern," on page 20-10.
5. Click Update.

20.7. 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 20-2).
2. Select the spawn pattern that you want to delete.
3. Click Delete Pattern. You are prompted to confirm the deletion.
4. Click Yes.

20.8. 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:

- 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.

20.9. 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 17-8.) 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 20-10.
- 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 20-10, 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.

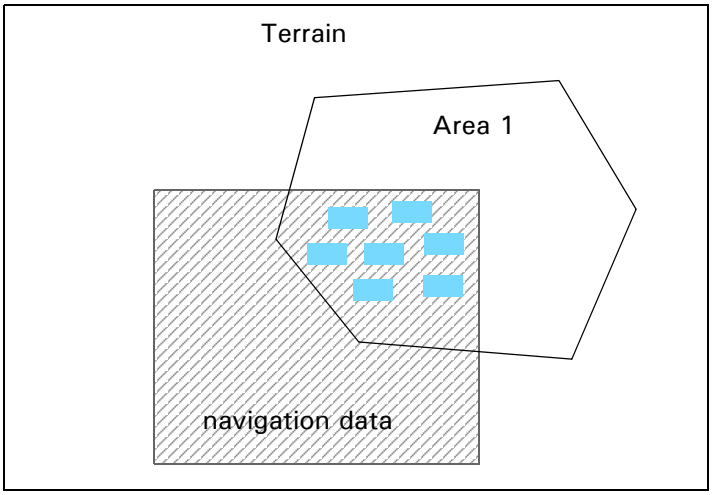


Figure 20-10. Placement of multiple simulation objects

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 Entities dialog box opens (Figure 20-11).

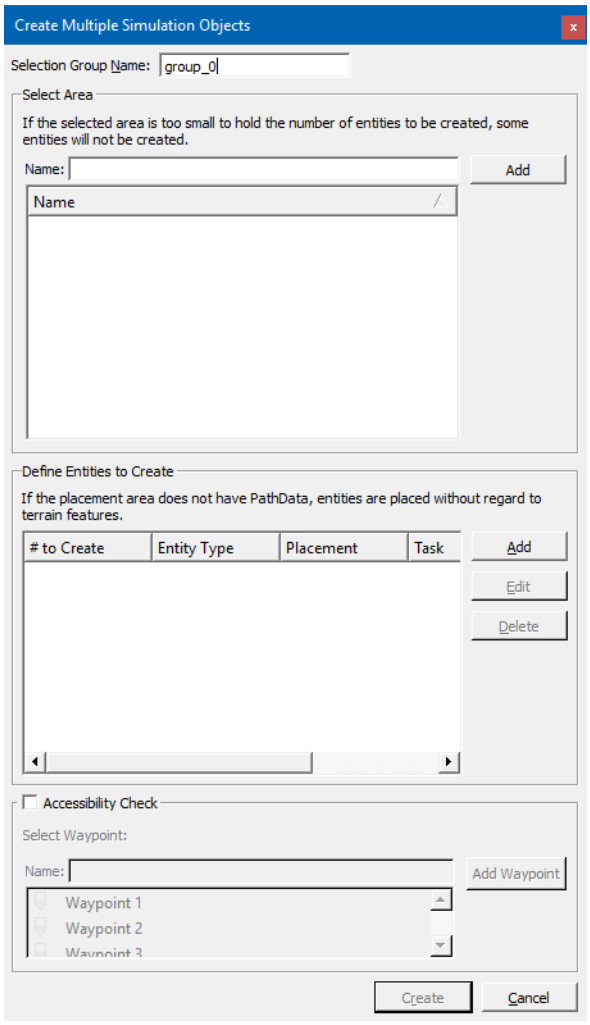


Figure 20-11. Create Multiple Entities dialog box

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 20-20.
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 20-20.

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.

20.9.1. 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 16-11.

20.9.2. 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 20-12).

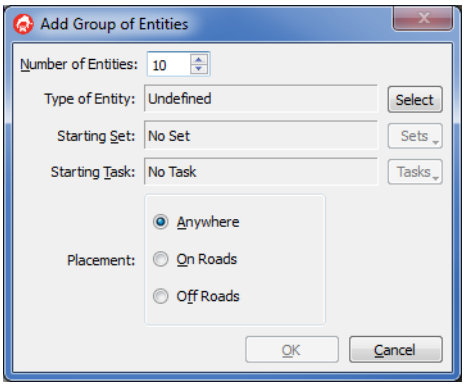


Figure 20-12. Add Group of Entities dialog box

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. Click OK. The object set is added to the list in the Define Entities To Create group box (Figure 20-11).
9. Optionally, add additional object sets.

20.10. Creating Crowds

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 and might not be available to other character types.

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.

Crowds 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. Select the Unit category.
3. Select the Civilian Crowd (Small), Civilian Crowd (Medium), or Civilian Crowd (Large) unit.
4. Click on the terrain to place the unit.

20.10.1. 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 (entity or point)
- Crowd along a line.
- Gather in front of an object.
- Flee.
- Wander.
- Protest.
- Protest along a line.
- Disperse.

For details about each task, please see its section in Chapter 26, *Tasks for Entity-Level Scenarios*.

20.11. Pedestrian-Oriented Tactical Graphics

VR-Forces has the following tactical graphics that are particular to pedestrians:

- 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. Pedestrian paths are created from the Hazards/Obstacles Palette.
- Civilian visit points. Waypoints that attract the attention of civilians. Civilian visit points are created from the Tactical Graphics Palette.

20.11.1. 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.



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

When you use entity-level modeling, VR-Forces models the behavior of individual (leaf-level) members of a unit. The members of a unit in an entity-level scenario 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.

Entity-level units are simulated in either of two states:

- Disaggregated: Subordinates are simulated and published as individual entities (Figure 21-1). 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.

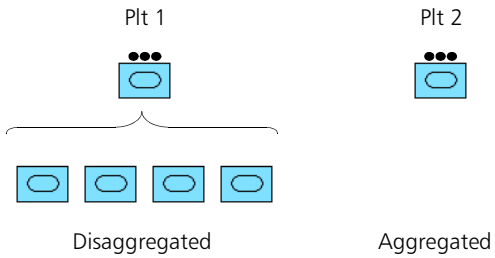


Figure 21-1. Disaggregated and aggregated units

- 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 22-2
- Unit Combat 22-3
- Unit Footprints 22-4
- Unit Posture 22-4
- Unit Movement 22-6
- Limited Munition Attack 22-7
- Contamination Areas 22-7
- Unit Sensors 22-8

Combat Engineering Objects 22-9
- Concealed Obstacles 22-11
- Creating Combat Engineering Objects 22-12
- How Engineering Units Create Engineering Objects 22-12

Logistics 22-14
- Receiving Supplies 22-16
- Providing Supplies 22-16

Electronic Warfare 22-16
- Jamming Communications 22-17
- Jamming Radar 22-18
- Sensing Electronic Emissions 22-19

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 MÄK 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.
- Electronic warfare. The ability to jam communications.
- 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 five 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 units, 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 55-9.

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 22-9.

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.

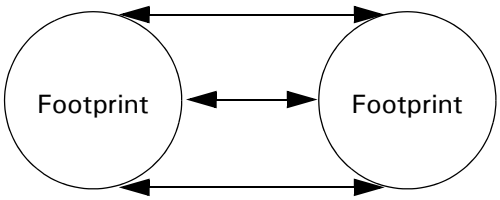


Figure 22-1. Aggregate-level sensing

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 Hazards/Obstacles 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 18-35.)



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 22-11.

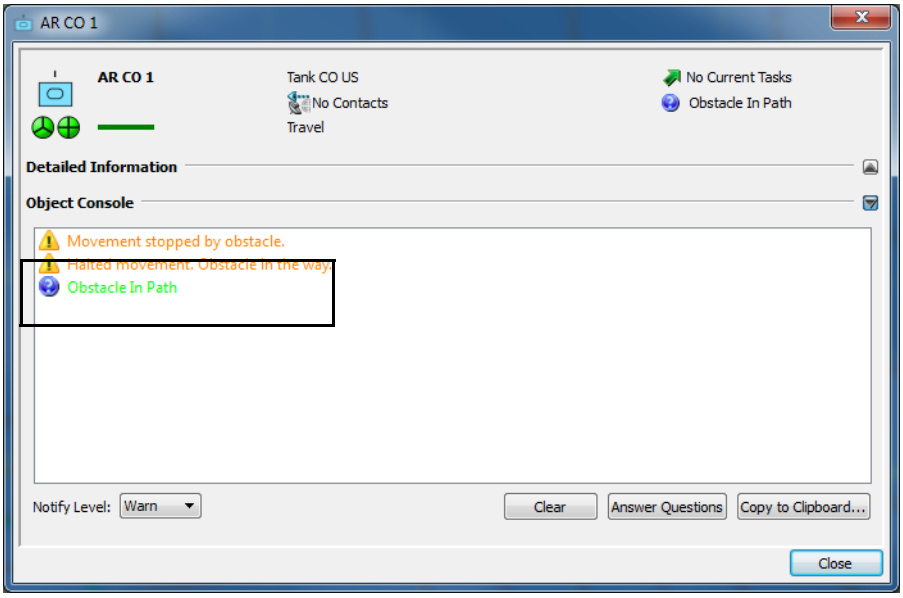


Figure 22-2. Question in entity console

- 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.
- 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 Hazards/Obstacles 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 Hazards/Obstacles 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 Hazards/Obstacles 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).

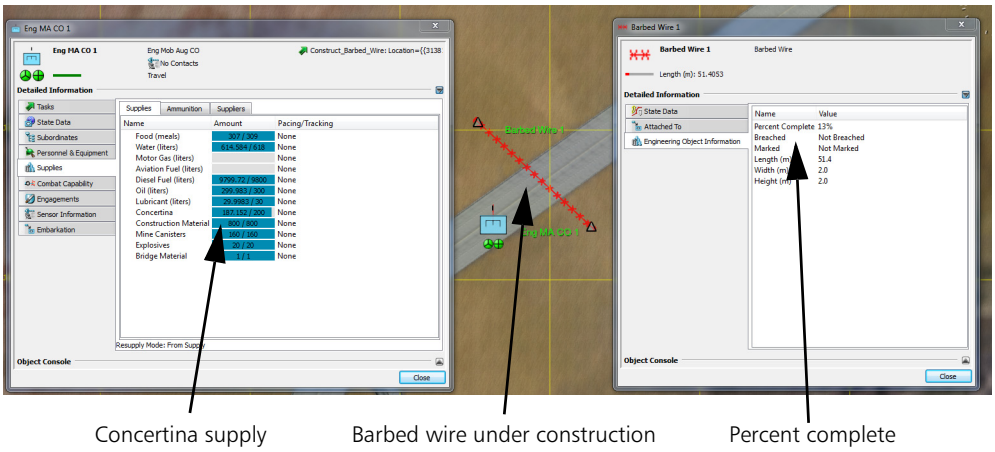


Figure 22-3. Road under construction

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.

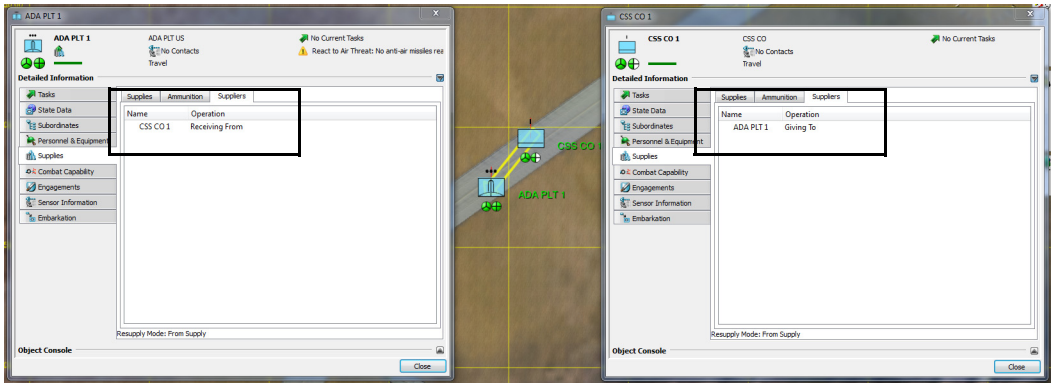


Figure 22-4. Supply information and supply lines

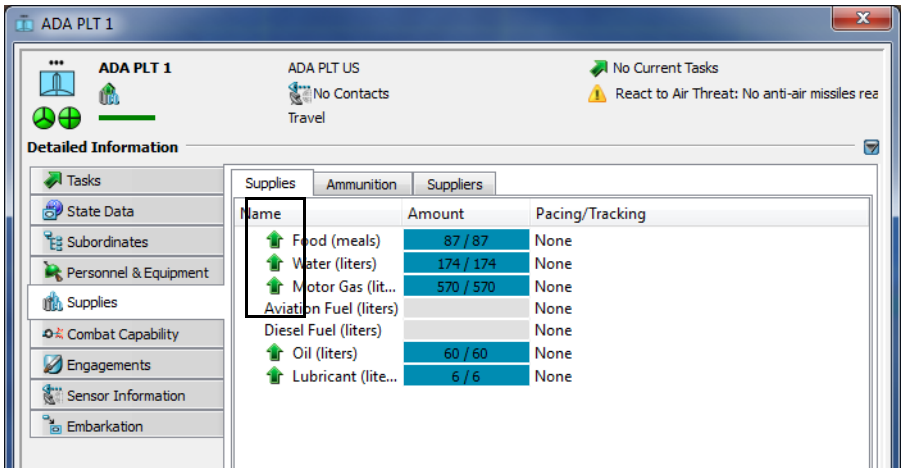


Figure 22-5. Supplies being received

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.

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.

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 aggregateReleaseBombControllerDescriptor 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.



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
Writing Plans for Units 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.

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** → **Aggregate As**. The Aggregate As dialog box opens (Figure 23-1).

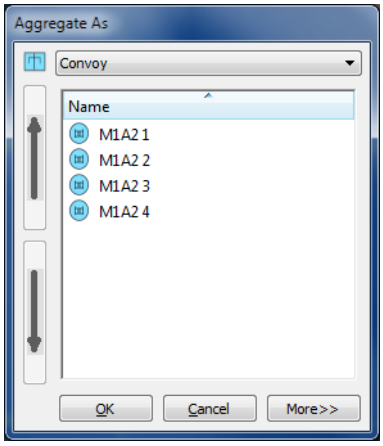


Figure 23-1. Aggregate As dialog box

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 65-3.

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).

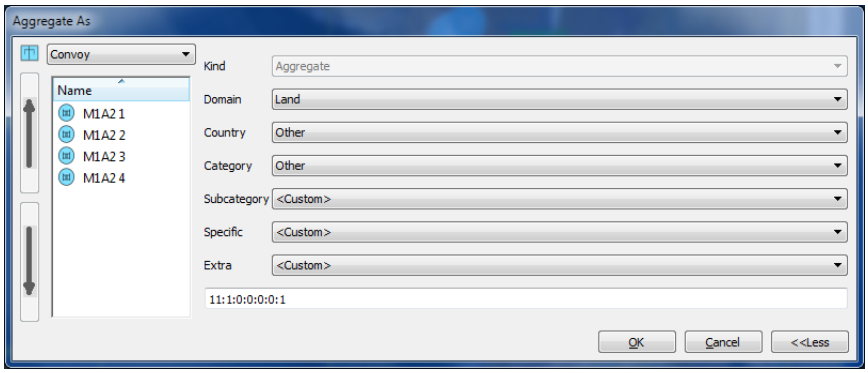


Figure 23-2. Aggregate As dialog box

   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 65-4.)

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 Options page (Figure 23-3).

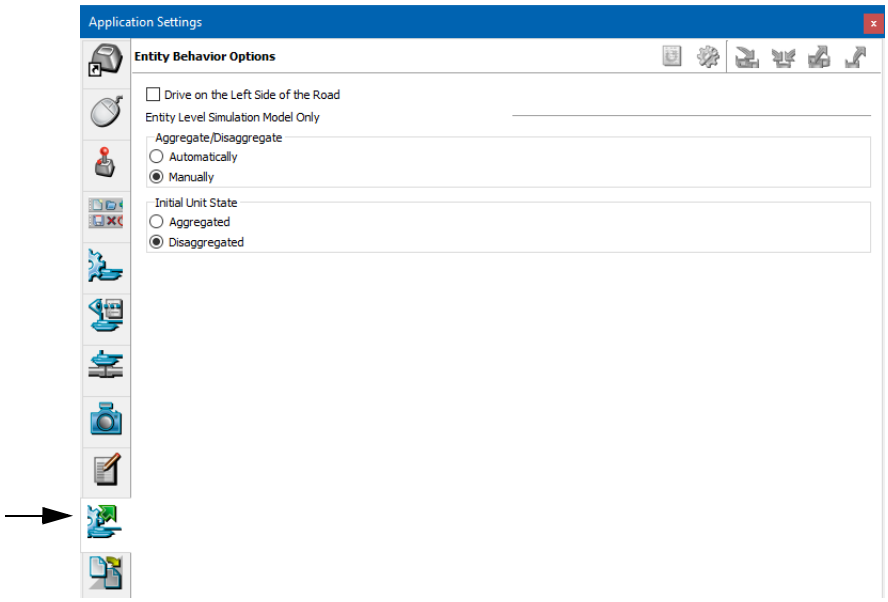


Figure 23-3. Entity Behavior Options page

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 Echelon View, select the simulation object you want to add and drag it onto the unit as illustrated in Figure 23-4. Notice that the name of the simulation object stays the same, but its echelon ID changes.

- 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 30-41.

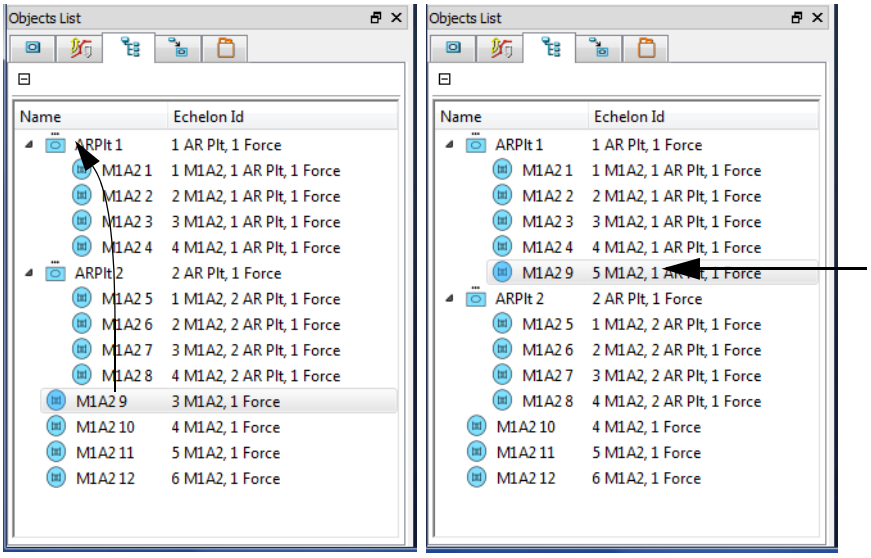


Figure 23-4. Adding a simulation object to a unit

23.4. Removing a Simulation Object from a Unit

➤ To remove a simulation object from a unit, in the Echelon 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. Notice that the name of the simulation object stays the same, but its echelon ID changes.

- 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 30-41.

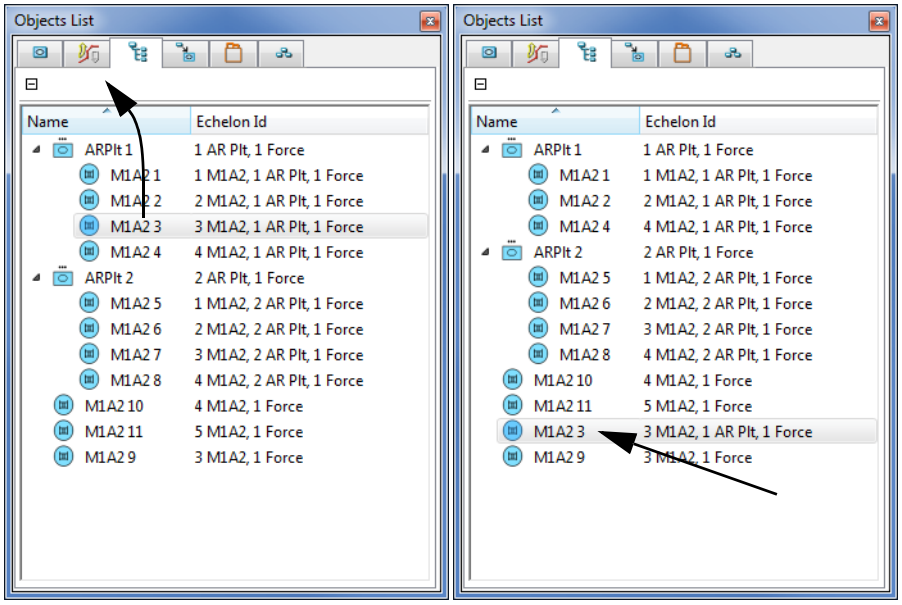


Figure 23-5. Removing a simulation object from a unit

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.

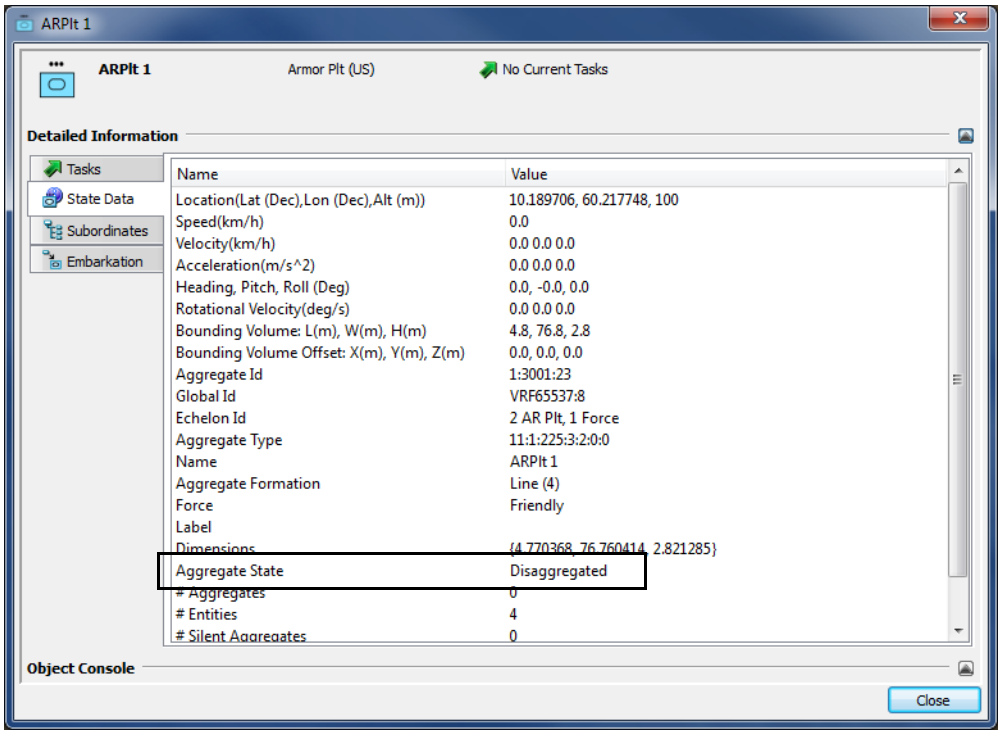


Figure 23-6. State Data page

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 Echelon 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 Echelon 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 Options 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 the 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. (You must show advanced parameters to set this parameter.)
  - 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** → **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** → **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 Options 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 16, *Creating and Placing Objects*.

23.7. Writing Plans for Units

➤ To write a plan for a unit, follow the procedures for writing a plan for a simulation object in Chapter 32, *Writing Plans*.

23.8. 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
Expanding and Collapsing Units in the Echelon View 24-4
Expanding and Collapsing the Echelon View by Level of Aggregation .. 24-4
Displaying Ghosted Icons 24-6
Displaying Unit Icons and Bounding Volumes 24-9
Specifying the Color Scheme for Unit Bounding Volumes 24-11

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 17-4.) 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 Echelon 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)

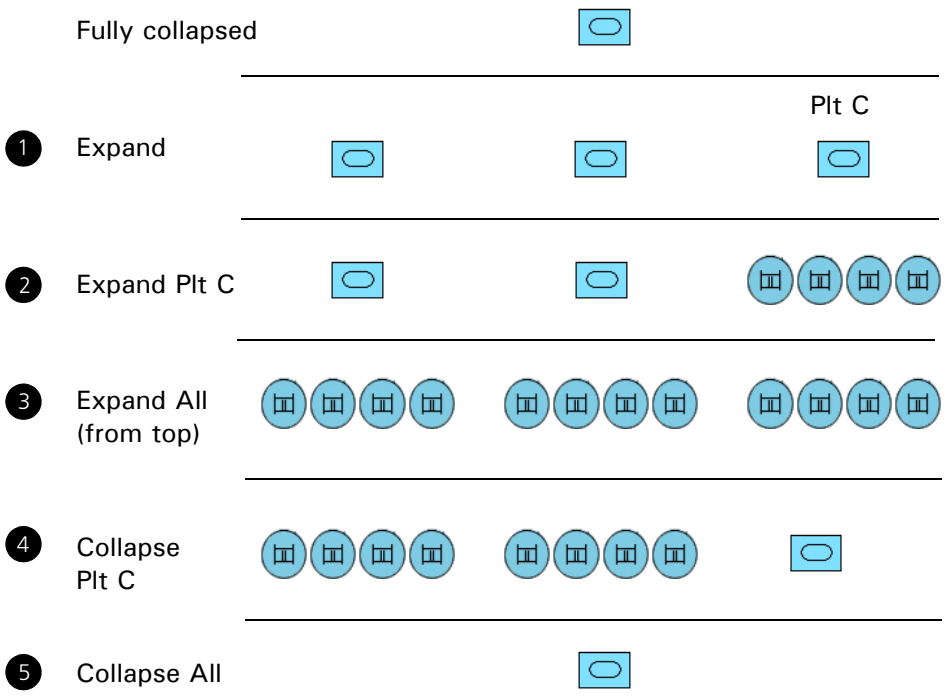


Figure 24-1. Unit expand and collapse

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 Echelon 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 Echelon 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, right-click the unit icon or its entry in the Objects List Panel and choose **Expand** on the popup menu.
- To expand a unit and all its subordinate units, right-click the unit icon or its entry in the Objects List Panel and choose **Expand All** on the popup menu.
- To collapse a unit one level, right-click any member of the unit and choose **Collapse** on the popup menu.
- To collapse a unit to its root level, right-click any member of the unit and choose **Collapse All** on the popup menu.

24.2.1. Expanding and Collapsing Units in the Echelon View

You can expand and collapse units in the Echelon View on the Objects List Panel.

➤ To expand a unit in the Echelon View, click the plus sign next to its icon.

➤ To collapse a unit in the Echelon View, click the minus sign next to its icon.

24.2.2. Expanding and Collapsing the Echelon View by Level of Aggregation

In addition to expanding and collapsing the order of battle on a node-by-node basis, as in typical hierarchical view windows, the Echelon View enables you to view simulation objects by the level of aggregation.

Individual entities are directly subordinate to the force they belong to. If a simulation object becomes part of a unit, the unit is subordinate to the force and the individual entity is one level further away. Seen another way, you can say that the force is the parent and all other simulation objects in that force are children, grandchildren, and so on.

The Echelon View displays the simulation objects in columns that are equivalent to their relationship to the force – the first column is children, the second column is grandchildren, and so on.

At the top of each column in the Echelon View is a plus (+) or minus (-) button that works like the buttons within the hierarchical display (Figure 24-2). A plus indicates that at least one simulation object in that column is a unit and that none of the units in the column are expanded. A minus indicates that at least one unit at that level is expanded. Clicking a minus sign collapses all units in the column. Clicking a plus sign expands all units in the column.

➤ To change the hierarchical level view in the Echelon View, click the level controls in the horizontal band across the top of the window.

Figure 24-2 illustrates an Echelon View with several levels of units, some of which are collapsed and some of which are expanded.

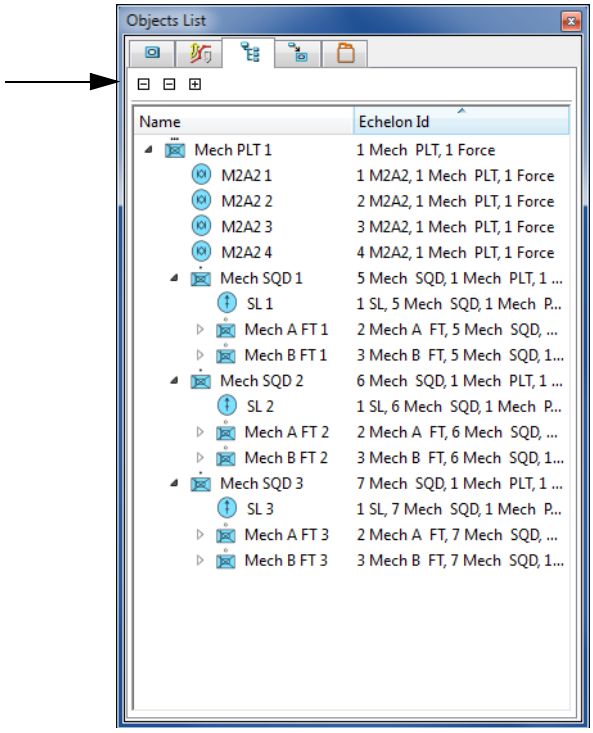


Figure 24-2. Echelon View

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 24-3).

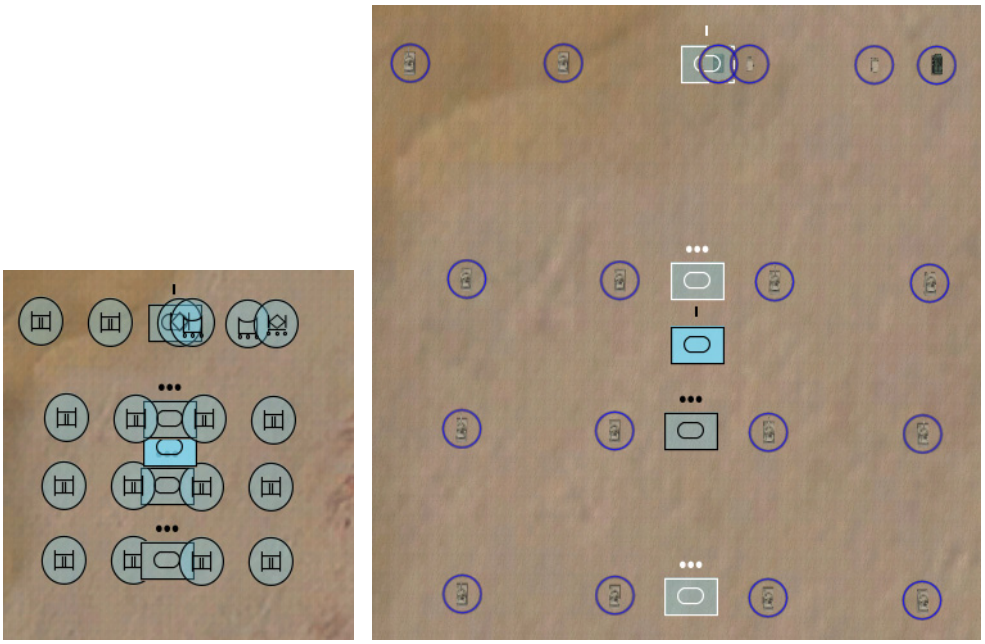


Figure 24-3. Ghosted icons

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 24-4).

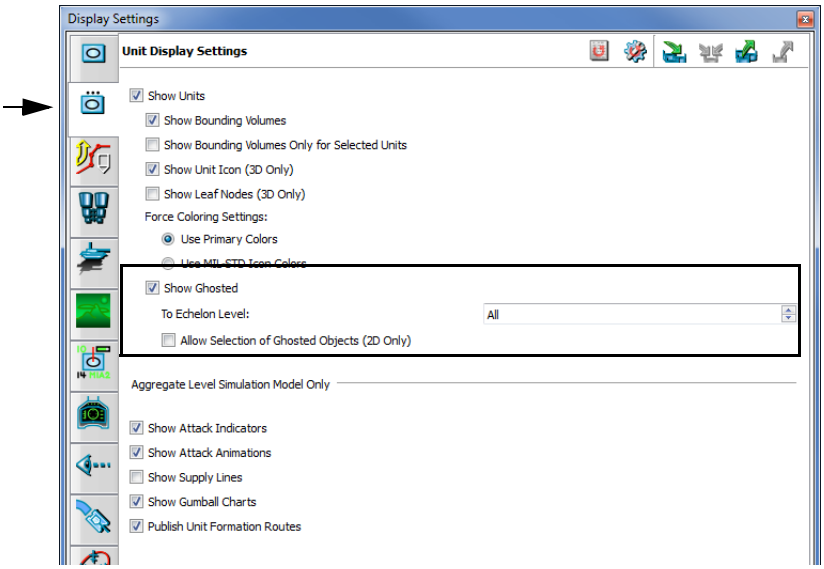


Figure 24-4. Unit Display Settings page

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 24-5 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.)

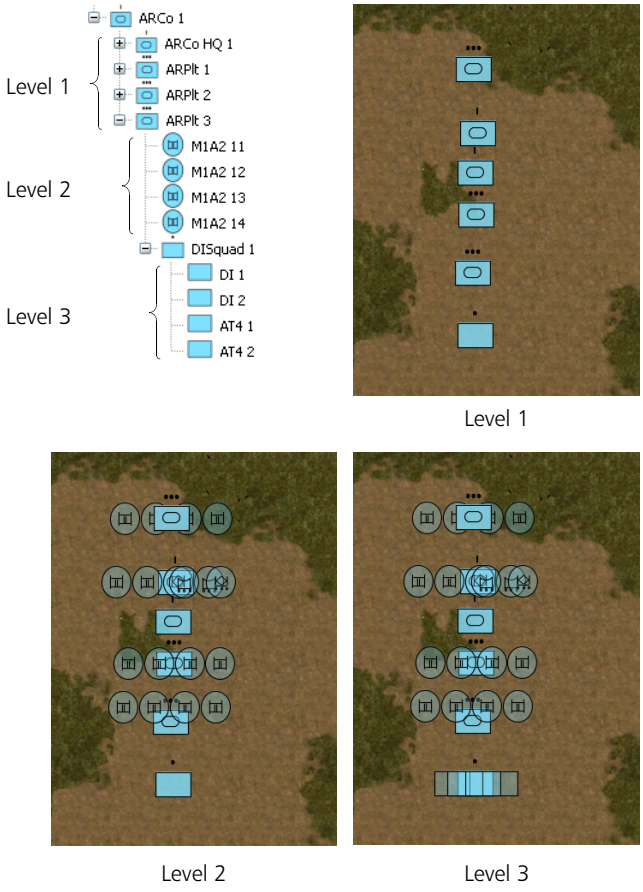


Figure 24-5. Displaying ghosted icons by echelon level

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 24-6 shows a unit bounding volume in Stealth observer mode, with leaf nodes displayed.

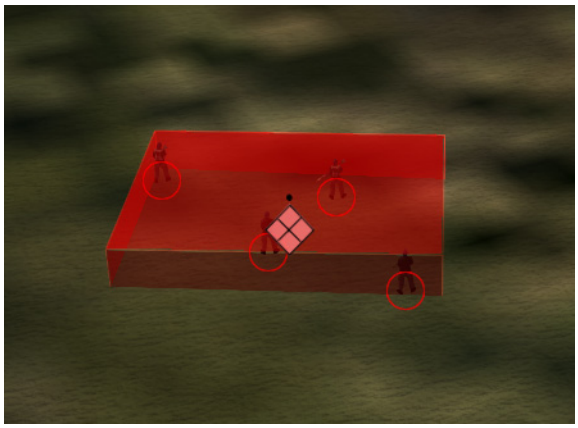


Figure 24-6. 3D bounding volume

Figure 24-7 shows a unit bounding volume in Plan View observer mode, with ghosting enabled.

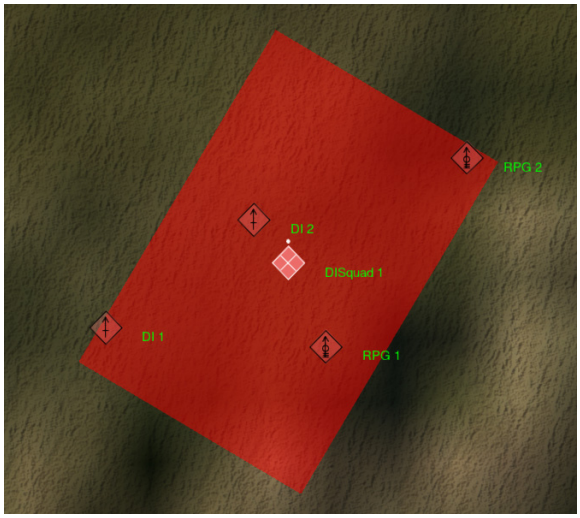


Figure 24-7. 2D bounding volume

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 24-4).
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 24-4).
3. Select the Force Coloring Settings option that you want (Use Primary Colors or Use MIL-STD Icon Colors).



V. Tasks, Sets, and Scripted Tasks

This section describes tasks, set data requests, and plans.

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 task 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 70-14.

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. Tasks are described in detail in Chapter 25, *Assigning Tasks*.

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," on page 28-1.

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 29-29.

Figure V-1 illustrates the many ways that a simulation object can have its task, behavior, and state changed.

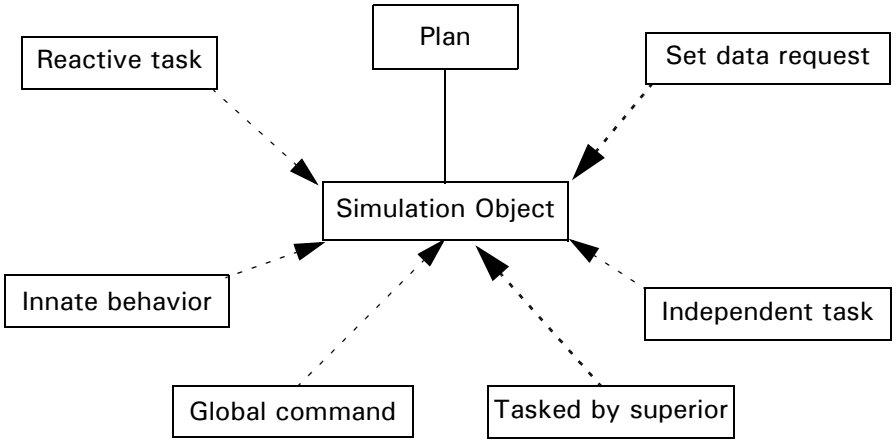


Figure V-1. Inputs to simulation object actions

V.2. Introduction to Set Data Requests

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 affect 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. The various set data requests are described in detail in Chapter 30, *Setting Simulation Object State*.



25. Assigning Tasks

This chapter provides general details about how to assign tasks to simulation objects. For descriptions of specific tasks and how to assign them, please see Chapter 26, *Tasks for Entity-Level Scenarios* and Chapter 27, *Tasks for Aggregate-Level Scenarios*.

- Assigning Tasks to Simulation Objects 25-3
  - C++ Tasks and Scripted Tasks 25-3
  - Concurrent Task Execution 25-4
  - How do I Know which Simulation Object 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 Scripted Tasks 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
Fixed-Wing Entity Tasks and Behaviors 25-22
Placement of Newly Created Fixed-Wing Entities 25-22
Fly Task Behavior is Different from Move Task Behavior 25-23
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 25-27
How Fixed-Wing Entities Land 25-28
Rotary-Wing Entity Tasks and Behaviors 25-29
Controlling Rotary-Wing Orientation 25-31

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 24-1. 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 32-3. For the procedures for assigning tasks, please see Chapter 26, *Tasks for Entity-Level Scenarios* and Chapter 27, *Tasks for Aggregate-Level Scenarios*.

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 31-17.

25.1.1. 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 28, *Creating Scripted Tasks*.

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. They only apply to aggregate-level scenarios. (For details, please see "Background Process List," on page 27-29.)

25.1.2. 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 70-14. For information about scripted tasks, please see Chapter 28, *Creating Scripted Tasks*.

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).

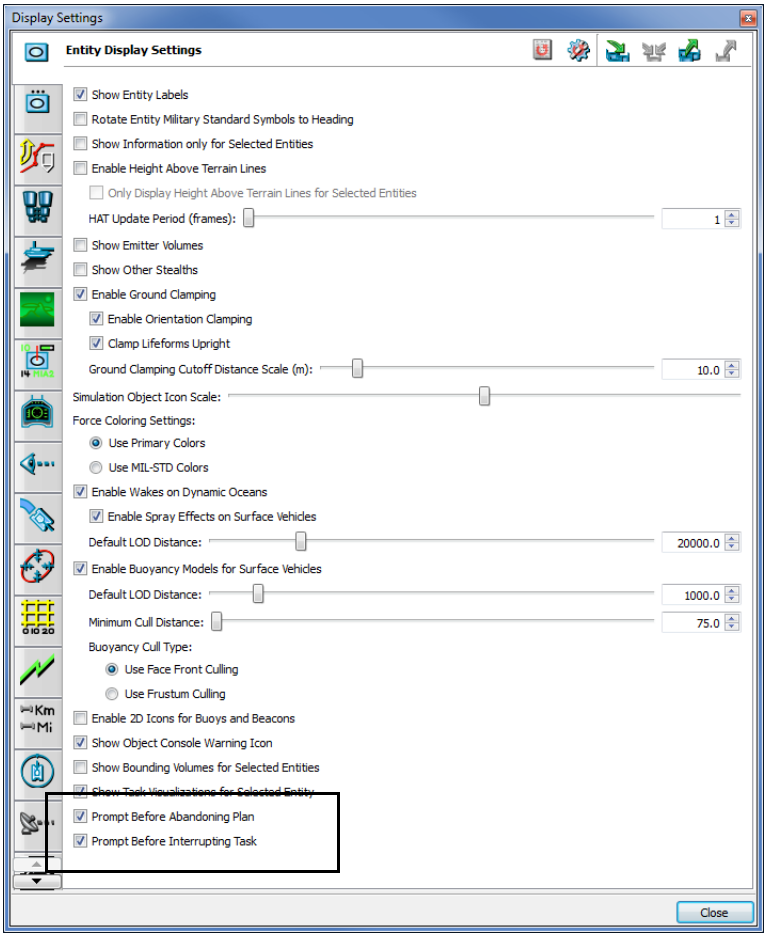


Figure 25-1. Entity Display Settings page

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.

25.1.3. How do I Know which Simulation Object 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 C, *Systems and System Usage* lists each system provided with VR-Forces, the types of simulation objects it supports, and its description. It also lists each system and the simulation objects that are configured to use it.

25.1.4. 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.

25.1.5. 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).

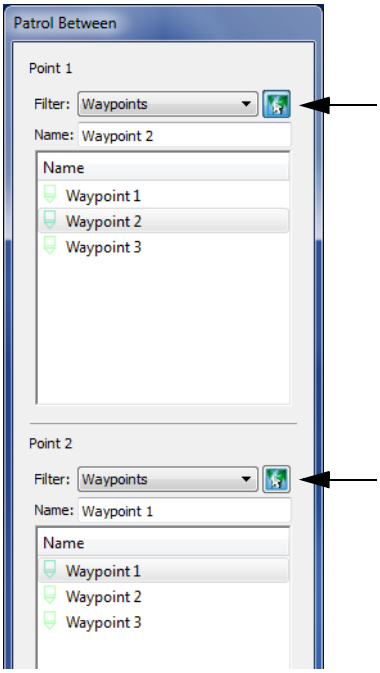


Figure 25-2. Multiple list dialog box

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 will stay 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.

25.1.6. Viewing Task Status

The Information dialog box for a simulation object lists its current task on the Tasks page. 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 also see which task it is executing by opening its Plan window (Figure 25-3).

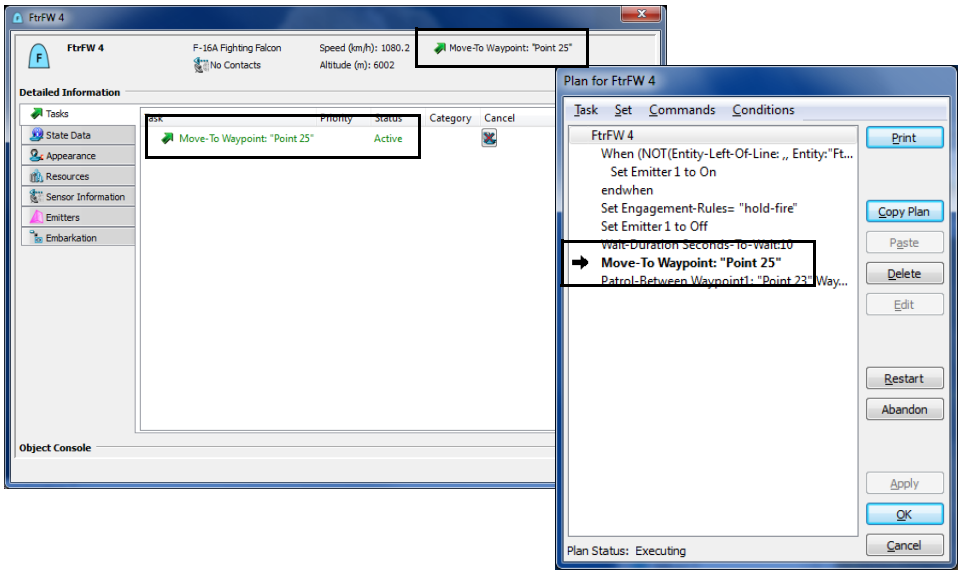


Figure 25-3. Display of simulation object task status

25.1.7. 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.

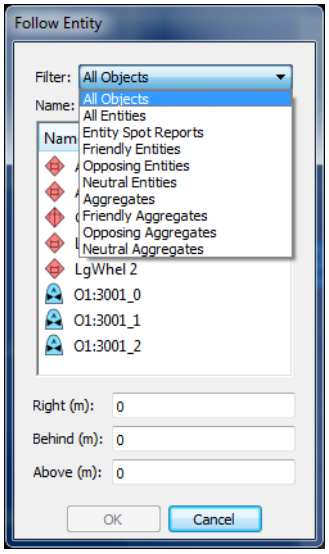


Figure 25-4. Simulation object filter list

➤ To filter a simulation object list, in a task or set data request dialog box, choose the filter type from the list.

25.1.8. 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** → **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 in the OPD Editor. 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 `When` 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 valid.
- 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 nonconcurrent independent task assignment, from the Task menu or a global plan, the reactive task and all suspended tasks are cancelled 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 29, *Writing Scripts for Scripted Tasks*.

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.

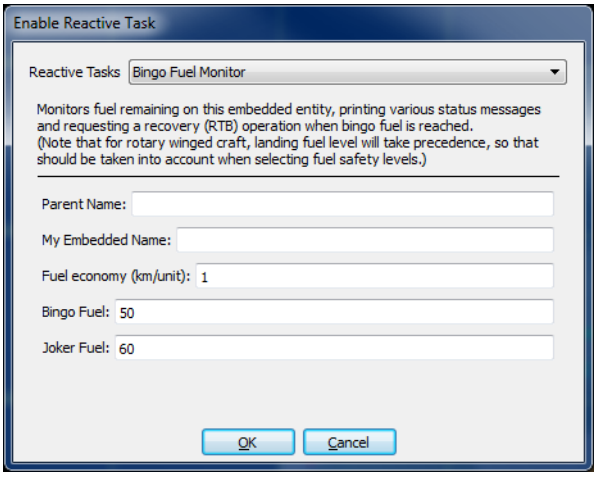


Figure 25-5. Enable Reactive Task dialog box

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.

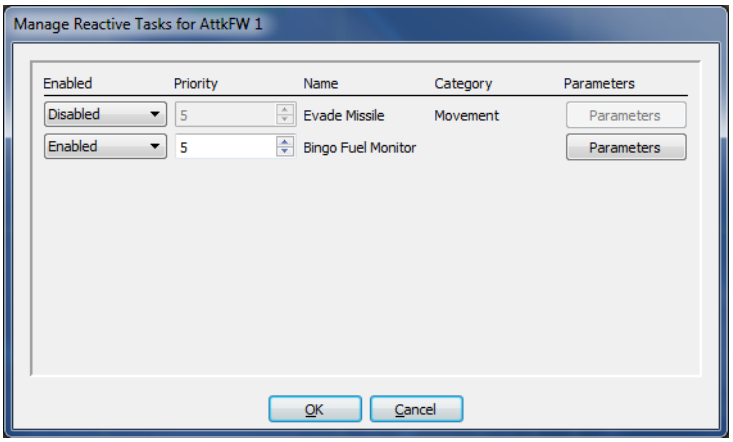


Figure 25-6. Manage Reactive Task dialog box

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).

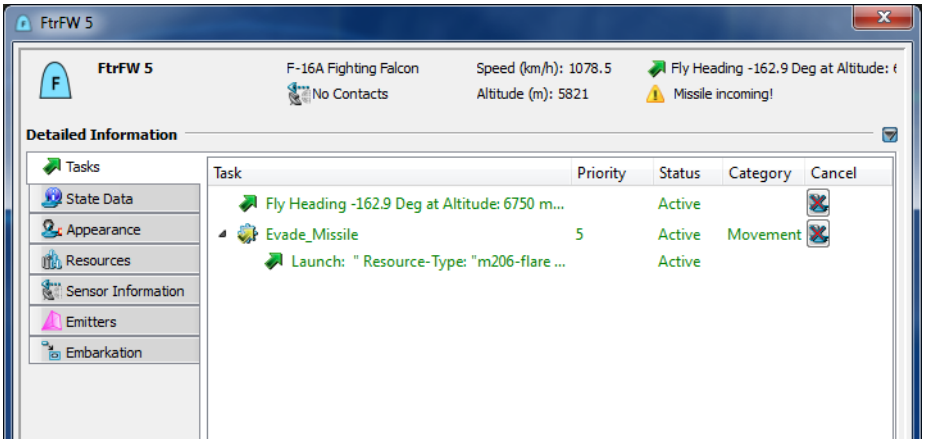


Figure 25-7. Active reactive task

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 Scripted Tasks

Behavior Sets let you organize scripted tasks so that a set of related tasks can be enabled or disabled at the force level.

If a scripted task 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 task.
- 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 scripted task 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 Scripted Tasks 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 scripted tasks to each Behavior Set.
3. Before you ran 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).

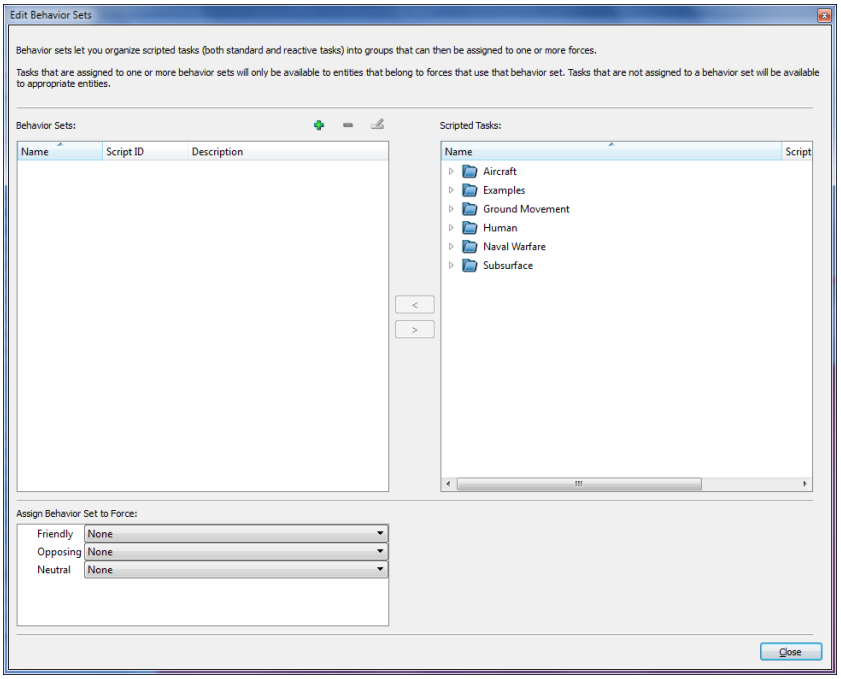


Figure 25-8. Edit Behavior Sets dialog box

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 Scripted Tasks list, select the scripted tasks that you want to assign to this Behavior Set. Click the left-facing arrow. The scripted tasks 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 Scripted Task dialog box. For details, please see "Creating a New Scripted Task," on page 28-6.)
6. 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 scripted tasks 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.

25.6. 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 a fixed-wing entity moves to a point or between the vertices in a route, it immediately ascends or descends to the altitude of the target point. This is illustrated in "Fixed-Wing Movement Between Vertices," on page 25-26.
- When an entity 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. In this case the entity immediately moves to the new altitude and then proceeds to the target point.



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.

Fixed-wing entities do not support terrain-following. Therefore, when you task a fixed-wing entity to move to a point or follow a route, you must be certain that there are no points in the terrain that are higher than the altitude of the entity's flight path, or the entity will crash. For example, if you assign a fixed-wing entity to follow a route whose vertices are set at 3000 meters above sea level, if the terrain rises above 3000 meters at some point in between two vertices of the route, the entity will crash into the terrain at that point (Figure 25-9). 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 49-3.)

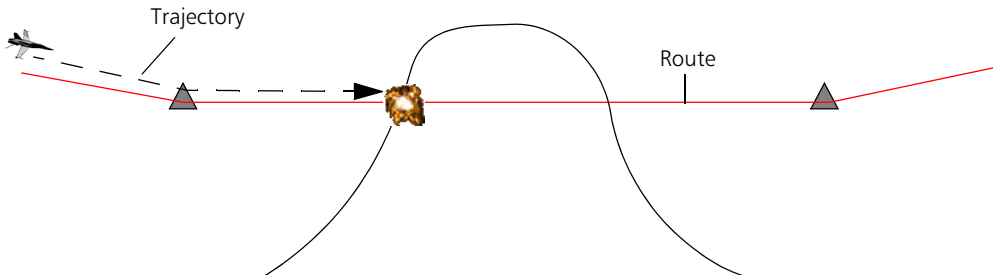


Figure 25-9. Fixed-wing route following

Fixed-Wing Movement Between Vertices

If two vertices in a route are not at the same altitude, a fixed-wing entity does not fly from the first vertex to the second in a smooth gradient. As soon as it passes the first vertex, it immediately descends to the altitude of the next vertex and then continues to fly towards it at that altitude. Therefore, it is possible that you could create a route that does not intersect the terrain, but an entity flying along that route could crash because a location in the terrain exceeds the altitude of a destination point, as illustrated in (Figure 25-10).

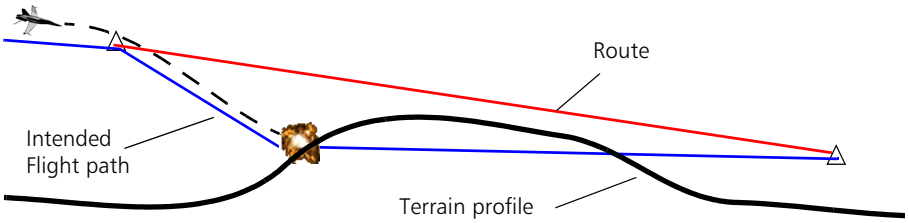


Figure 25-10. Fixed-wing trajectory between vertices



If you want a plane to change altitude using a smooth gradient, use one of the Fly Altitude tasks. However, these tasks do not let you follow a route.

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 26-27) 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 25-11 illustrates this behavior.

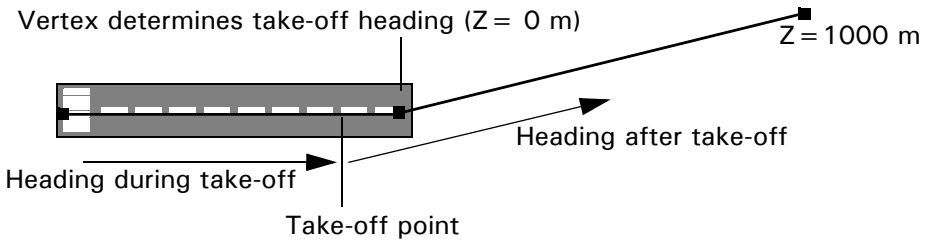


Figure 25-11. Defining a runway for take-off

25.6.6. How Fixed-Wing Entities Land

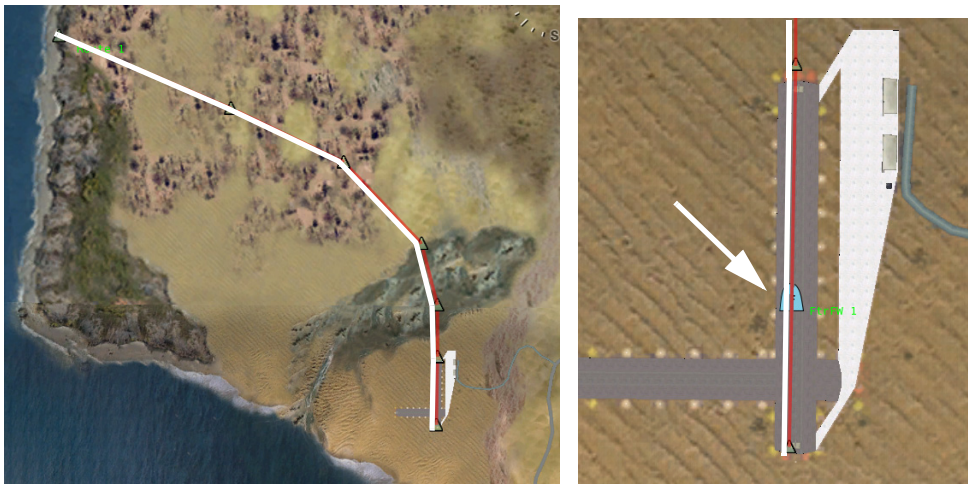
You can have aircraft land using the Fixed Wing Land task (please see "Fixed-Wing Land," on page 26-26), or as described in this section. The underlying behavior of the Fixed Wing Land task is the same as in this section. Using the Fixed Wing Land task automates the process of setting up the approach 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 10 meters of the terrain and it is moving within 10 percent of its minimum lift speed, it will try to land.

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 25-11 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 25-12. 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 16-14.)

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.
- 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.

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 25-13). 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 49-3.)

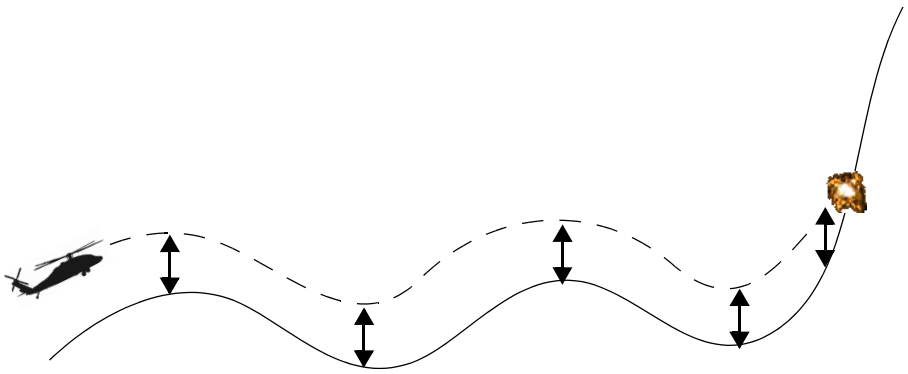


Figure 25-13. Helicopter terrain avoidance

25.7.1. Controlling Rotary-Wing Orientation

A rotary-wing entity can move laterally (sideways) without changing its orientation. In other words, it does not have to face in the direction it is moving. Rotary-wing entity movement controllers have two parameters that configure the speed at which a rotary-wing entity can move laterally. Rotary-wing entities rotate to try to align themselves with their velocity vector under the following circumstances:

- If the current speed is greater than the speed specified by **vel-to-align-actual**.
- If the speed of the entity is not greater than **vel-to-align-actual**, but the desired speed of the entity (the velocity it is attempting to achieve as a result of position or velocity commands) is greater than **vel-to-align-desired**.

If you do not want a rotary-wing entity to align itself with its direction of travel, set **vel-to-align-actual** and **vel-to-align-desired** to speeds that are greater than the speed at which you expect the entity to travel. You can set these parameters in the OPD Editor.



26. Tasks for Entity-Level Scenarios

This chapter describes the tasks supplied with VR-Forces that apply to entity-level scenarios.

Task Procedures 26-4
Animated Movement 26-5
Arm Mine at Depth 26-6
Attack Aircraft with Guns 26-6
Civilian Idle 26-7
Civilian Protest 26-7
Come to Stop 26-7
Convoy Along 26-8
Convoy To 26-9
Crowd Along Line 26-9
Crowd Around Location 26-10
Crowd Around Object 26-10
Crowd in Front of Entity 26-11
Deploy Sonobuoy 26-11
Deploy Sonobuoys Along Route 26-12
Disembark 26-13
Disembark All 26-13
Disperse Crowd 26-13
DI-Guy Animation 26-14
Drop Naval Depth Charge 26-14
Drop Naval Depth Charge at Location 26-15

Drop Naval Mine 26-16
Drop Naval Mines Along Route 26-17
Embark 26-18
Execute Close Air Support 26-21
Explode Charge at Depth 26-23
Fire at Target 26-23
Fire Cruise Missile 26-24
Fire for Effect Tasks 26-25
    Firing Naval Guns 26-26
Fixed-Wing Land 26-26
Fixed Wing Takeoff 26-27
Flee From Entities 26-27
Fly Altitude 26-28
Fly Heading 26-29
Fly Heading and Altitude 26-30
Follow Entity 26-31
    How Fixed-Wing Entities Follow Entities 26-31
Lase Target 26-32
Launch Anti-Submarine Missile (Vertical) 26-32
Launch Counter Measures 26-33
Launch Smoke 26-33
Launch Torpedo 26-34
    Anti-ship (Fixed Depth) 26-34
    Anti-Submarine 26-35
Lower Periscope 26-35
Move Along Route 26-36
Move Into Formation 26-37
Move to Altitude 26-38
Move to Depth 26-38
Move to Location 26-39
    Adding Multiple Move to Location Tasks to a Plan 26-40
Move to Location (Plan Along Roads) 26-40
Move to Location (Plan Path) 26-41
Move to Waypoint 26-42
Move to Waypoint (Plan Along Roads) 26-43
Move to Waypoint (Plan Path) 26-44

Orbit 26-45
Patrol Route 26-45
Patrol Between 26-46
Pattern Hold (Location) 26-46
Pattern Hold (Waypoint) 26-47
Place IED 26-48
Protest Around Location 26-49
Protest Along Line 26-49
Suppressive Fire Tasks 26-50
    Provide Suppressive Fire 26-51
    Provide Suppressive Fire at Location 26-52
Raise Periscope 26-52
Release Bomb Tasks 26-53
Rotary-Wing Land 26-54
Sail Heading 26-54
Send Radio Set 26-55
Send Radio Task 26-56
Send Text Message 26-57
Sonar Dip 26-57
Strafe Ground Target 26-58
Sweep Naval Mines 26-60
Taxi and Launch from Carrier 26-60
Taxi on Carrier 26-61
Turn to Heading 26-61
User Task 26-62
Wait Tasks 26-62
    Wait 26-62
    Wait Duration 26-63
    Wait Elapsed 26-63
Wander 26-63
Wander In Area 26-64
Reactive Tasks 26-64

26.1. Task Procedures

This chapter describes the tasks that simulation objects in entity-level scenarios can run. Tasks that apply to aggregate-level scenarios are described in Chapter 27, *Tasks for Aggregate-Level Scenarios*.

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 25-22 and "Rotary-Wing Entity Tasks and Behaviors," on page 25-29, 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.

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 64-28.

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).

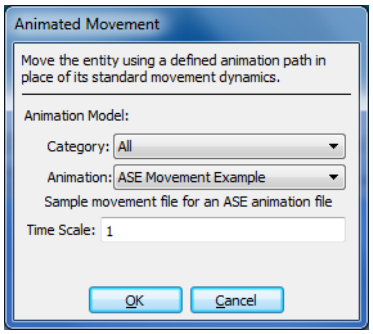


Figure 26-1. Animated Movement dialog box

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. 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 arm a mine:

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.

26.4. 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 issue the 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.

26.5. Civilian Idle

The Civilian Idle task causes a male or female civilian entity to stand in place and cycle through various standing and talking animations.

To assign a Civilian Idle task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Civilian Idle**.

26.6. Civilian Protest

The Civilian Protest task causes a male or female civilian entity to stand in place and cycle through various protest animations.

To assign a Civilian Protest task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Civilian Protest**.

26.7. 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.

26.8. 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.

26.9. 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 his option.

4. Click OK.

26.10. 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 cause a crowd to gather along a line:

1. Select the 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.

26.11. Crowd Around Location

The Crowd Around Location task causes a crowd unit to cluster around the specified location.

To cause a crowd to crowd around a location:

1. Select the 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.

26.12. 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 cause a crowd to crowd around a simulation object:

1. Select the 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.

26.13. 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 cause a crowd to crowd in front of an entity:

1. Select the 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.

26.14. 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. Deploy Sonobuoy is a system scripted task.



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 deploy a sonobuoy:

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.

26.15. 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. Deploy Sonobuoys Along Route is a system scripted task.



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 deploy sonobuoys along a route:

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.

26.16. 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. To disembark simulation objects immediately, that is, without using a task, please see "Embarking and Disembarking Simulation Objects," on page 19-7.



- When a simulation object disembarks, it simply goes to the disembarkation 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.

To disembark a simulation object:

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.

26.17. 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 disembark all simulation objects from a parent simulation object:

1. Select the parent simulation object.
2. Choose **Task** → **Embarkation** → **Disembark All**.

26.18. Disperse Crowd

The Disperse Crowd task causes a crowd to wander freely.

To disperse a crowd:

1. Select the crowd.
2. Choose **Task** → **Movement** → **Disperse Crowd**.

26.19. 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 Appearance," on page 30-15.



The DI-Guy Animation task applies only to lifeform entities.

To specify a DI-Guy animation:

1. Select the lifeform entity whose behavior you want to animate.
2. Choose **Task** → **DI-Guy** → **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.

26.20. Drop Naval Depth Charge

The Drop Naval Depth Charge task drops a depth charge at the current location of the entity. To execute this task, an entity must have a Naval Depth Charge Deployment weapon system. Drop Naval Depth Charge is a system scripted task.

To drop a naval depth charge:

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.

26.21. Drop Naval Depth Charge at Location

The Drop Naval Depth Charge at Location task drops a depth charge at a specified location. To execute this task, an entity must have a Naval Depth Charge Deployment weapon system. Drop Naval Depth Charge at Location is a system scripted task.



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 drop a naval depth charge at a location:

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.

26.22. Drop Naval Mine

The Drop Naval Mine task allows entities with a Naval Mine Deployment system to drop naval (underwater) mines at their current location. Drop Naval Mine is a system scripted task.

To drop a naval mine:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Drop Naval Mine**. The Drop Naval Mine dialog box opens.
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.

26.23. Drop Naval Mines Along Route

The Drop Naval Mines Along Route task allows entities with a Naval Mine Deployment system to drop naval (underwater) mines while they travel along a route. Drop Naval Mines Along Route is a system scripted task.

To drop naval mines along a route:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Drop Naval Mine**. 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.

26.24. 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 19-7.

Table 26-1 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 26-1: Embarkation support

Entity	Can Carry	HF
AH-6 Little Bird	1 human	
Aircraft Carrier	12 fighters or 9 fighters and 2 cargo planes. Three helicopters. 70 planes below deck.	x
Arleigh Burke-class Destroyer	1 aircraft	
Bell 206 Jet Ranger	4 humans	
Bell 406	6 humans	
BMP-2 AFV	7 humans	
BTR-80 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-46E Sea Knight	11 humans	
CH-47 Chinook	55 humans	

Table 26-1: Embarkation support

Entity	Can Carry	HF
CH-53E Super Stallion	55 humans	
EC 145	9 humans	
Fire Engine	6 humans	
GAZ-69 Utility Vehicle	4 humans	
Guided Missile Destroyer	2 helicopters	
CH-47 Chinook	55 humans	
HMMWV (all types)	4 humans	x
LCAC	9 HMMWVs or 4 trucks or 1 M1A2 or 1 generic ground vehicle	
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	
M939A2 5-Ton Truck	14 humans	
MD500	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	
MT-LB Multipurpose Armored Vehicle	11 humans	
Nimitz Class Carrier	48 aircraft, including fighter jets, helicopters, and reconnaissance.	
OH-58 Kiowa	1 human	
P-3 Orion	11 humans	
Rigid Hull Inflatable Boat	8 humans	
Rigid Hull Inflatable Boat - Civilian	6 humans	
SH-60 Seahawk	11 humans	
Space Shuttle	7 humans	
Technical Truck	4 humans	
Tractor Trailer Cab	2 humans	
Tractor Trailer with Cab	2 humans	
UH-60 Blackhawk	11 humans	

Table 26-1: Embarkation support

Entity	Can Carry	HF
UH-72 Lakota	9 humans	
V-22 Osprey	24 humans, 1 light ground vehicle	
VAB APC	10 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. VR-Forces includes entries on the Simulation Connections Configuration dialog box that start VR-Forces using the correct FED file and FOM Mapper for RPR FOM 1.0, and RPR FOM 2.
- 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 embark a simulation object:

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** → **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. Click OK.

26.25. 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. This is a system scripted 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 execute close air support:

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 26-2).
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.

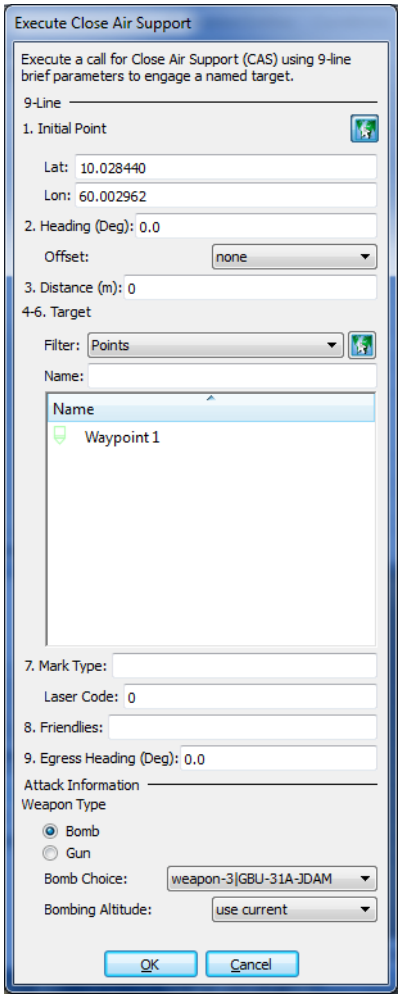


Figure 26-2. Execute Close Air Support dialog box

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.

26.26. 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 explode a depth charge:

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.

26.27. 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 order an entity to fire at a target:

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.)

26.28. Fire Cruise Missile

A cruise missile can fly directly to a target or follow a route.

To fire a cruise missile:

1. Select an entity that has cruise missiles.
2. Choose **Task** → **Engagement** → **Fire Cruise Missile**. The Fire Cruise Missile dialog box opens (Figure 26-3).

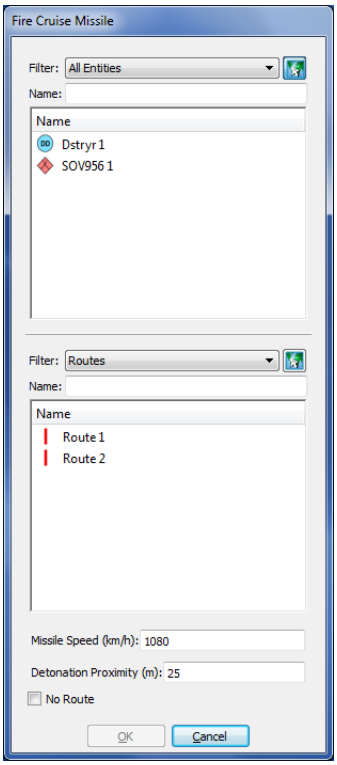


Figure 26-3. Fire Cruise Missile dialog box

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.

26.29. 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 18-25.) The back-end console window prints messages when you select targets that are outside of the entity's range.

26.29.1. 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.

26.30. Fixed-Wing Land

"How Fixed-Wing Entities Land," on page 25-28, 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. Fixed-Wing Land is a system scripted task.

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 land a fixed-wing aircraft:

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.

26.31. 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 25-27.) 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. Fixed-Wing Takeoff is a system scripted task.

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.

26.32. Flee From Entities

The Flee From Entities task causes an individual human entity or all the members of a crowd to flee from a specified entity. They flee until they are a specified distance away. For crowds, the task is complete when all members of the crowd have fled.

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 or crowd to flee from.
4. Optionally, specify the radius of the flight zone.
5. Click OK.

26.33. 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.

26.34. 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.

26.35. 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.

26.36. 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 31-16).

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 order an entity to follow a simulation object:

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.

26.36.1. 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. 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.

26.37. 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.

To lase a target:

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.

26.38. 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. This is a system scripted task.

To launch an anti-submarine missile:

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.

26.39. 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 9-19.)

To launch counter measures:

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.

26.40. 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 launch a smoke cloud:

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.

26.41. 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. The Launch Torpedo tasks are system scripted tasks.



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.

26.41.1. Anti-ship (Fixed Depth)

To launch a torpedo that moves at a fixed depth:

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.

26.41.2. Anti-Submarine

The anti-submarine torpedo homes in on the center of the target.

To launch an anti-submarine torpedo:

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.

26.42. 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. Lower Periscope is a system scripted task.

To lower a submarine's periscope:

1. Select the entity.
2. Choose **Task** → **Other** → **Lower Periscope**.

26.43. 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 Movement Between Vertices," on page 25-26 and "Rotary-Wing Entity Tasks and Behaviors," on page 25-29.)
- 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 direct a simulation object to move along a route:

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.

26.44. 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 direct a unit to move to a formation:

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.

26.45. 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 send an entity to an altitude:

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.

26.46. 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.

26.47. Move to Location

You can order a simulation object to move to a location on the terrain. 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 direct a simulation object to move to a location:

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 16-14.



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.

26.47.1. 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 16-14.
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.

26.48. 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. Move to Location (Plan Along Roads) is a system scripted task.

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 move a ground vehicle to a location using roads:

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.

26.49. 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.

This task is a system scripted task.

To move a surface or subsurface vehicle to a location with path planning:

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.

26.50. 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 send a simulation object to a waypoint or entity:

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.



When you task an air platform entity to move to an object, be sure to take into account the altitude setting for the object and the terrain the entity has to pass over to complete the task.

26.51. 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. Move to Waypoint (Plan Along Roads) is a system scripted task.

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 direct an entity to move to a waypoint using the road network:

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.

26.52. 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 based 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.

This task is a system scripted task.

To move a surface or subsurface vehicle to a waypoint with path planning:

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.

26.53. 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 specify 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 points.
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.

26.54. 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.

To order a simulation object to patrol a route:

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.



When you task an air platform entity to follow a route, be sure to take into account the altitude of the vertices and the terrain the entity has to pass over to complete the task.

26.55. 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 order a simulation object to patrol between two objects:

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.



When you task an air platform entity to patrol between objects, be sure to take into account the altitude of the objects and the terrain the entity has to pass over to complete the task.

26.56. Pattern Hold (Location)

The Pattern Hold (Location) task assigns a fixed-wing or rotary-wing entity to execute a standard hold pattern at the specified location. This is a system scripted task.

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 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. Click OK.

26.57. Pattern Hold (Waypoint)

The Pattern Hold (Waypoint) task assigns a fixed-wing or rotary-wing entity to execute a standard hold pattern at the specified waypoint. This is a system scripted task.

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. Select the waypoint.
4. Specify the heading, speed, and altitude for the hold pattern.
5. Specify the turn direction for the hold pattern.
6. Click OK.

26.58. 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. Place IED is a system scripted task.

To assign a place IED task to a human entity:

1. Select the entity.
2. Choose **Task** → **Engagement** → **Place IED**. The Place IED dialog box opens (Figure 26-4).

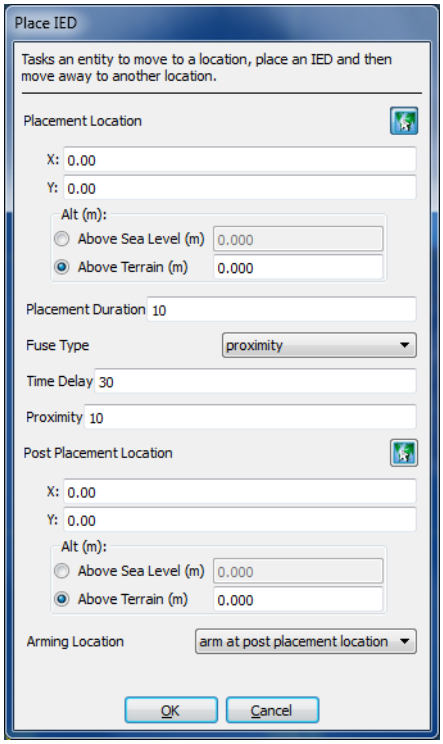


Figure 26-4. Place IED dialog box

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. Enter 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.

26.59. 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 cause a crowd to protest around a location:

1. Select the crowd.
2. Choose **Task** → **Movement** → **Protest**. The Protest Around Location dialog box opens.
3. Specify a location.
4. Click OK.

26.60. 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 cause a crowd to protest along a line:

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.

26.61. 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 hysteresis 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. This number of suppression levels was chosen arbitrarily. Higher levels of suppression are reached when the suppression value passes higher thresholds. The suppression level of the entity, a value between 0 and 3, can be viewed in the state data of the Information dialog box.

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 the Effects of Suppressive Fire," on page 70-17.

26.61.1. 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 26-50.



The target of suppressive fire must be at least 25 meters away.

To order suppressive fire on a point, a line, or in an area:

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.

26.61.2. 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 26-50.

The target of suppressive fire must be at least 25 meters away.

To order suppressive fire on a location:

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.

26.62. 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. Raise Periscope is a system scripted task.

To raise a submarine's periscope:

1. Select the entity.
2. Choose **Task** → **Other** → **Raise Periscope**.

26.63. 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 9-15.
- "Lase Target," on page 26-32.
- "Laser Code," on page 30-26.
- "Synchronize Laser Code," on page 30-42.

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.

26.64. 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 land a rotary-wing aircraft:

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.

26.65. 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.

26.66. 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 send a set data request radio message:

1. Select the simulation object that will send the message.
2. Choose **Task** → **Radio**→ **Send Radio Set** → *category* → *set_data_request*, where *category* is one of the categories of set data requests and *set_data_request* is one of the set data requests for that category. A dialog box for the set data request opens.
3. 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.
4. If the selected set data request requires input, complete the dialog box as described in Chapter 30, *Setting Simulation Object State*.
5. 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 Communication Models," on page A-8.)
6. Click OK.

26.67. 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 send a task radio message:

1. Select the simulation object that will send the message.
2. Choose **Task** → **Radio**→ **Send Radio Task** → *category* → *task*, where *category* is one of the categories of tasks and *task* is one of the tasks in that category. A dialog box for the task opens.
3. 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.
4. If the selected task requires input, complete the dialog box.
5. To send the message to all simulation objects configured to receive messages from this entity's radio, select the Broadcast check box. (For details about configuring the radio model, please see "Configuring Communication Models," on page A-8.)
6. Click OK.

26.68. 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 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 31-13.)

To send a text message:

1. Select the simulation object that will send the message.
2. Choose **Task** → **Radio** → **Send Text Message**. The Send Text dialog box opens.
3. In the Name text box, type the name of the simulation object that will receive the text message, or select the simulation object in the list or on the terrain.
4. In the Message Text box, type the message.
5. 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 Communication Models," on page A-8.)
6. Click OK.

26.69. 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. Sonar Dip is a system scripted task.

To dip sonar:

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.

26.70. 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 26-2 shows how arguments to the Execute Close Air Support task map to the Strafe Ground Target task.

Table 26-2: 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.

To strafe a ground target:

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 26-5).

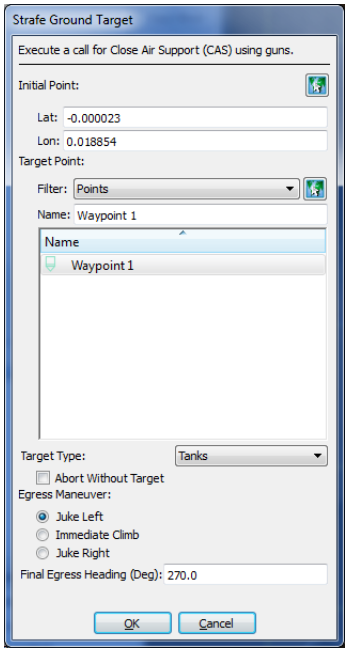


Figure 26-5. Strafe Ground Target dialog box

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.

26.71. 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. Sweep Naval Mines is a system scripted task.

To assign the 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.

26.72. Taxi and Launch from Carrier

The Taxi and Launch from Carrier task causes an aircraft to move to a takeoff route (catapult) and take off. The aircraft can taxi directly to the takeoff route or can follow a route to get to it. You can specify that the aircraft fly a holding pattern after takeoff.

This task is only for use with aircraft carriers that are part of the terrain. The *./userData/scenarios/CarrierDemo/FlightDeckOps* scenario is an example of using this task.

To cause an aircraft to taxi and launch from a carrier:

1. Select the aircraft.
2. Choose **Task** → **Movement** → **Taxi and Launch from Carrier**. The Taxi and Launch from Carrier dialog box opens.
3. Optionally specify a route to taxi along.
4. Specify a route to act as the catapult.
5. Optionally, specify that the aircraft fly a holding pattern and specify the altitude.
6. Click OK.

26.73. Taxi on Carrier

The Taxi On Carrier task causes an aircraft to taxi on the deck of an aircraft carrier to a location or along a route and assume a specified heading.

! This task is only for use with aircraft carriers that are part of the terrain. The *./userData/scenarios/CarrierDemo/FlightDeckOps* scenario is an example of using this task.

To cause an aircraft to taxi to a location on an aircraft carrier:

1. Select the aircraft.
2. Choose **Task** → **Movement** → **Taxi on Carrier**. The Taxi on Carrier dialog box opens.
3. Click on the carrier to specify the location or type the coordinates in the dialog box.
4. Optionally, specify a route to follow to the location.
5. Optionally, specify that the aircraft turn to a heading and specify the heading.
6. Click OK.

26.74. 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 order an entity to turn to a heading:

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.

26.75. 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.

26.76. Wait Tasks

You can order a simulation object to wait indefinitely, for a specified period of time, or until a specified elapsed simulation time.



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.

26.76.1. Wait

To order a simulation object to wait indefinitely:

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, such as the result of a `When` statement being triggered.

26.76.2. Wait Duration

To order a simulation object to wait a specific amount of time:

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.

26.76.3. Wait Elapsed

To order a simulation object to wait until a specific elapsed simulation time:

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.

26.77. 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.

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.

26.78. Wander In Area

The Wander In Area task causes a crowd to wander in the specified area.

To assign a crowd to wander:

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.

26.79. 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 26-3 lists the reactive tasks provided with VR-Forces.

Table 26-3: Reactive tasks

Reactive Task	Description
Evade Missile	Causes fixed-wing entities to try to evade missiles.
Bingo Fuel Monitor	Monitors fuel on embedded entities and requests a recovery when bingo fuel level is reached.
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.
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.



27. Tasks for Aggregate-Level Scenarios

This chapter describes tasks that apply to aggregate-level scenarios.

Introduction to Aggregate-Level Tasks 27-4
Activate Jammer 27-4
Air-to-Ground Attack 27-4
Attack by Fire 27-5
Attack with Anti-Air Missile 27-5
Attack with Anti-Ship Missile 27-5
Attack with Land-Attack Missile 27-6
Attack with Torpedo 27-6
Automatic Air Defense 27-7
Biological Attack 27-7
Bomb Location 27-8
Breach Obstacles 27-9
Change MOPP Level 27-9
Change Posture 27-10
Chemical Attack 27-10
Construct Abatis 27-11
Construct Anti-Tank Ditch 27-12
Construct Barbed Wire 27-12
Construct Barricade 27-13
Construct Berm 27-13
Construct Booby Trap 27-14
Construct Bridge 27-15

Construct Dry Gap 27-16
Construct Fortified Area 27-17
Construct Fortified Line 27-18
Construct Minefield 27-18
Construct Strong Point 27-19
Deactivate Jammer 27-19
Destroy Bridge 27-20
Destroy Ditch 27-20
Destroy Explosive 27-20
Destroy Fortification 27-21
Destroy Obstacle 27-21
Disembark 27-21
Disembark All 27-21
Embark 27-21
FASCAM 27-22
Halt Movement 27-22
Improve Booby Trap 27-22
Improve Breach 27-23
Improve Bridge 27-23
Improve Ditch 27-23
Improve Fortification 27-24
Improve Obstacle 27-24
Indirect Fire 27-25
Move Along Route 27-26
Move Along Route Retrograde 27-26
Move to Location 27-26
Move to Location Retrograde 27-26
Move To Waypoint 27-26
Move to Waypoint Retrograde 27-26
Nuclear Attack 27-27
Patrol Route 27-27
Patrol Between 27-27
Road Construction Tasks 27-27
Send NBC Report 27-28
Send Obstacle Report 27-28

Send Radio Set 27-28
Send Radio Task 27-28
Send Text Message 27-28
User Task 27-29
Wait Tasks 27-29
Background Process List 27-29

27.1. 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. These tasks are cross referenced to their descriptions in Chapter 26, *Tasks for Entity-Level Scenarios*.

27.2. 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. It is a system scripted task.

➤ To activate a simulation object's jammer, choose **Task** → **Jamming** → **Activate Jammer**.

27.3. Air-to-Ground Attack

The Air-to-Ground Attack task requires an attack (ingress) route, an exit (egress) route, and a target point. It is a system scripted task.

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 specify an air-to-ground attack:

1. Select the entity 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.

27.4. 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 order attack by fire:

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.

27.5. Attack with Anti-Air Missile

This task is available to any simulation object that has anti-air missiles. It is a system scripted task.

To attack a unit with an anti-air missile:

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.

27.6. Attack with Anti-Ship Missile

This task is available to any simulation object that has anti-ship missiles. It is a system scripted task.

To attack a unit with an anti-ship missile:

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.

27.7. Attack with Land-Attack Missile

This task is available to entities that have land-attack missiles, such as the Arleigh Burke DDG. It is a system scripted task.

To attack a unit with a land-attack missile:

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.

27.8. Attack with Torpedo

This task is available to entities that have torpedoes, such as the Arleigh Burke DDG. It is a system scripted task.

To attack a unit with a torpedo:

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.

27.9. 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. It is a system scripted task.

To enable automatic air defense:

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.

27.10. Biological Attack

The Biological Attack task lets a simulation object create an area contaminated with biological agents. It is a system scripted task.

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 execute a biological attack:

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.

27.11. 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 27-1). It is a system scripted task.

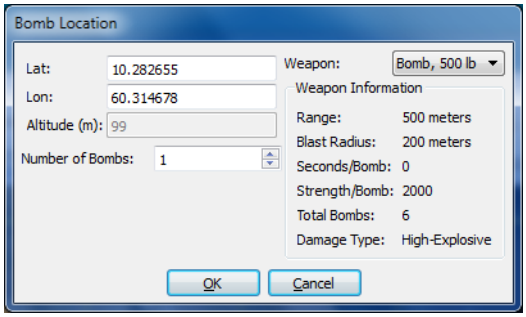


Figure 27-1. Bomb Location dialog box

To bomb a location:

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.

27.12. 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 breach an obstacle:

1. Select a combat engineering unit.
2. Choose **Tasks** → **Combat Engineering** → **Breach Obstacles**. The Breach Obstacles dialog box opens.
3. Click Define. The cursor changes to draw mode and the Create Breach dialog box opens or becomes available.
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 dialog box button now says Edit and the text next to it says Defined.
5. Click OK.

27.13. 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. It is a system scripted task.

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 change a unit's MOPP level:

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.

27.14. 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. Change Posture is a system scripted task.

To change a simulation object's posture:

1. Select the simulation object.
2. Choose **Task** → **Change Posture**. The Change Posture dialog box opens.
3. Specify a new posture.
4. Click OK.

27.15. Chemical Attack

The Chemical Attack task lets a unit create an area contaminated with chemical agents. It is a system scripted task.

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 execute a chemical attack:

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.

27.16. Construct Abatis

Combat engineering units can construct abatises. An abatis is specified as a point.



You can create abatises directly from the Hazards/Obstacles Palette. This task is used when you want a unit to construct this obstacle.

To construct an abatis:

1. Select a combat engineering unit.
2. Choose **Task** → **Combat Engineering** → **Construct Abatis**. The Construct Abatis dialog box opens (Figure 27-2). The Location is specified as "Not defined".



Figure 27-2. Construct Abatis dialog box

3. Click Define. The Create Abatis dialog box opens (Figure 27-3).

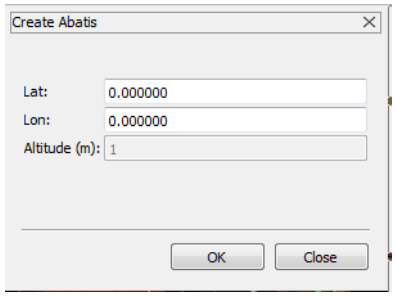


Figure 27-3. Create Abatis dialog box

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.

27.17. 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 Hazards/Obstacles Palette. This task is used when you want a unit to construct this obstacle.

To construct an anti-tank ditch:

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 27-2). 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.

27.18. 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 Hazards/Obstacles 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 27-2). 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.

27.19. Construct Barricade

Combat engineering units can construct barricades. A barricade is specified as a line.



You can create barricades directly from the Hazards/Obstacles 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 27-2). 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.

27.20. Construct Berm

Combat engineering units can construct berms. A berm is specified as a line.



You can create berms directly from the Hazards/Obstacles 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 27-2). 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.

27.21. 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 Hazards/Obstacles 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 27-2). 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.

27.22. Construct Bridge

Combat engineering units can construct bridges. A bridge is specified as a two-point line.



You can create bridges directly from the Hazards/Obstacles 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 **Tasks** → **Combat Engineering** → **Construct Bridge**. The Construct Bridge dialog box opens (similar to Figure 27-2). 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.

27.23. 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 Hazards/Obstacles 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 27-2). 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.

27.24. 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 Hazards/Obstacles 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 27-2). 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.

27.25. Construct Fortified Line

Combat engineering entities can construct fortified lines.

> You can create fortifications directly from the Hazards/Obstacles 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 27-2). 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.

27.26. Construct Minefield

Combat engineering entities can construct volcano minefields. Minefields are fixed size rectangles.

> You can create minefields directly from the Hazards/Obstacles 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 27-2). 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.

27.27. 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 Hazards/Obstacles 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 27-2). 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.

27.28. 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. Deactivate Jammer is a system scripted task.

➤ To deactivate a simulation object's jammer, choose **Task** → **Jamming** → **Deactivate Jammer**.

27.29. 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 **Tasks** → **Combat Engineering** → **Destroy Bridge**. The Destroy Bridge dialog box opens.
3. Select the bridge you want to destroy.
4. Click OK.

27.30. 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 **Tasks** → **Combat Engineering** → **Destroy Ditch**. The Destroy Ditch dialog box opens.
3. Select the ditch you want to destroy.
4. Click OK.

27.31. 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 **Tasks** → **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.

27.32. 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 **Tasks** → **Combat Engineering** → **Destroy Fortification**. The Destroy Fortification dialog box opens.
3. Select the fortification you want to destroy.
4. Click OK.

27.33. 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 **Tasks** → **Combat Engineering** → **Destroy Obstacle**. The Destroy Obstacle dialog box opens.
3. Select the obstacle you want to destroy.
4. Click OK.

27.34. Disembark

For details, please see "Disembark," on page 26-13.

27.35. Disembark All

For details, please see "Disembark All," on page 26-13.

27.36. Embark

For details, please see "Embark," on page 26-18.

27.37. 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 **Tasks** → **FASCAM**. The FASCAM dialog box opens (similar to Figure 27-2). 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.

27.38. Halt Movement

The Halt Movement task causes a simulation object to stop moving. It is a system scripted task.

To halt a simulation object's movement:

1. Select the simulation object.
2. Choose **Tasks** → **Movement** → **Halt Movement**. Movement stops immediately.

27.39. 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.

27.40. 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 **Tasks** → **Combat Engineering** → **Improve Breach**. The Improve Breach dialog box opens.
3. Select the breach you want to improve.
4. Click OK.

27.41. 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 **Tasks** → **Combat Engineering** → **Improve Bridge**. The Improve Bridge dialog box opens.
3. Select the bridge you want to improve.
4. Click OK.

27.42. 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.

27.43. 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.

27.44. 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.

27.45. 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. It is a system scripted task.

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.

27.46. Move Along Route

For details, please see "Move Along Route," on page 26-36.

27.47. 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. It is a system scripted task.

27.48. Move to Location

For details, please see "Move to Location," on page 26-39.

27.49. 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. It is a system scripted task.

27.50. Move To Waypoint

For details, please see "Move to Waypoint," on page 26-42.

27.51. 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. It is a system scripted task.

27.52. Nuclear Attack

The Nuclear Attack task lets a simulation object create an area contaminated with nuclear radiation. It is a system scripted task.

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 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.

27.53. Patrol Route

For details, please see "Patrol Route," on page 26-45.

27.54. Patrol Between

For details, please see "Patrol Between," on page 26-46.

27.55. Road Construction Tasks

The **Commands** → **Task** menu in plans contains the combat engineering tasks Construct Road, Improve Road, and Destroy Road. No entities are configured to use these tasks, so assigning them to a simulation object will fail. They are available for VR-Forces users who want to configure a simulation object to use them.

27.56. 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. It is a system scripted task.

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.

27.57. 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. It is a system scripted task.

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.

27.58. Send Radio Set

For details, please see "Send Radio Set," on page 26-55.

27.59. Send Radio Task

For details, please see "Send Radio Task," on page 26-56.

27.60. Send Text Message

For details, please see "Send Text Message," on page 26-57.

27.61. User Task

For details, please see "User Task," on page 26-62.

27.62. Wait Tasks

For details, please see "Wait Tasks," on page 26-62.

27.63. Background Process List

Background processes are not described in detail because they run automatically; no user action is required to execute them.

Table 27-1 lists the background tasks provided. These tasks are only available for units in *AggregateLevel.sms*.

Table 27-1: Background processes

Background Process	Description
Attack Enemies	Determines when to begin and stop attacking enemies.
Calculate Maximum Speed Modifier	Calculates a modifier that is applied to the maximum possible speed of a unit to modify its movement rate.
Halt When Engaged	Stop moving when attacking a hostile simulation object.
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 Sensors	Manages the state of a simulation object's sensors.
Perform Attrition	Determines when a unit is being attacked and modifies its health based on the current attrition rate.
Put on MOPP Gear	If a simulation object has MOPP gear, when it enters a hazardous area, it stops and changes to MOPP level 4.
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.



28. Creating Scripted Tasks

VR-Forces lets you create new tasks 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 scripted tasks. Chapter 29, *Writing Scripts for Scripted Tasks* explains how to write scripts for scripted tasks.

Introduction to Scripted Tasks 28-3
- Task Type Scripted Tasks 28-4
- The Reactive Task Type Scripted Task 28-5
- Background Process Type Scripted Task 28-5
- Script Metadata 28-5
- The Lua Scripting Language 28-5

Creating a New Scripted Task 28-6
- Specifying the Script ID 28-11
- Specifying the Task Menu Icon 28-11
- Specifying an Extended Name 28-12
- Specifying Task Parameters 28-12
- Specifying the Task Menu Location 28-16
- Specifying Action Categories 28-17
- Configuring a Script's Availability to a Simulation Object Type 28-18
- Specifying the Programming Language for a Scripted Task 28-23
- Creating Reactive Tasks 28-24
- Creating Background Processes 28-25

Saving Scripted Tasks 28-27

Editing a Scripted Task 28-28

Filtering the List of Scripted Tasks 28-28

Organizing Scripted Tasks into Folders 28-29
- Adding a Folder 28-29

Renaming a Folder 28-29
Deleting a Folder 28-29
Adding Scripted Tasks to a Folder 28-29
Removing a Scripted Task from a Folder 28-29
Exporting and Importing Scripted Tasks 28-30
Importing a Scripted Task Package 28-31
Copying a Scripted Task 28-32
Deleting a Scripted Task 28-32
Creating a System Scripted Task 28-33
Including System Scripted Tasks on the Task Menu 28-34
Specifying a Script Editor 28-36
Editing Lua Files 28-37

28.1. Introduction to Scripted Tasks

The scripted task feature in VR-Forces allows you to create new tasks for simulation objects without using the VR-Forces Toolkit and C++ API. Scripted tasks 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.

Scripted tasks are classified as system scripts or as scenario scripts. System scripted tasks are saved as part of the SMS and are available to all scenarios that use that SMS. Scenario scripted tasks 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 scripted tasks from one VR-Forces installation to another.

VR-Forces supports three types of scripted tasks, as follows:

- Task. Task scripted tasks can be listed on the Task menu and are 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.

Although creation of scripts requires that you have some programming skills, you do not need to know C++, use a compiler, or use the VR-Forces APIs to write these scripts.

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 28-36.

28.1.1. Task Type Scripted Tasks

Scripted tasks get 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. When you assign the task to a simulation object, either from the Task menu or in a plan, VR-Forces builds a dialog box for the task that includes all of the parameters specified for the script.

28.1.2. The Reactive Task Type Scripted Task

Reactive tasks are a special category of scripted tasks. Once enabled they 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 scripted tasks. Assume that all details about scripted tasks apply to reactive tasks unless noted otherwise. For conceptual details about reactive tasks, please see "Reactive Tasks," on page 25-11.

28.1.3. Background Process Type Scripted Task

The background process is a script that runs in the background for the simulation object types for which it was written. Strictly speaking it is not a task, but background processes are created and managed through the scripted task interface. Since a background task 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.

28.1.4. 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 the Task menu, and generate the task dialog box.

28.1.5. The Lua Scripting Language

Lua is a lightweight scripting language used by VR-Forces to implement the scripted tasks 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).

28.2. Creating a New Scripted Task

To create a new scripted task, you specify the metadata for the script and write the script code. A scripted task's metadata specifies the script's name, where it will be placed on the Task 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 scripted task is to specify a name for the script and add it to the list of scripted tasks. Once you do this, you can edit the metadata and the code as needed until you have completed the scripted task to your satisfaction.

All new scripted tasks are created as scenario tasks. If you want to create a new system scripted task, you save the scenario scripted task as a system scripted task. For details, please see "Creating a System Scripted Task," on page 28-33.



To create a scripted task, you must have a scenario open. This is true even if you plan to create a system scripted task rather than a scenario scripted task.

To create a new scripted task:

1. Open a scenario. (If you want to create a system scripted task, you must create it in the context of a scenario. Then you will promote it to a system task.)
2. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks dialog box opens (Figure 28-1).

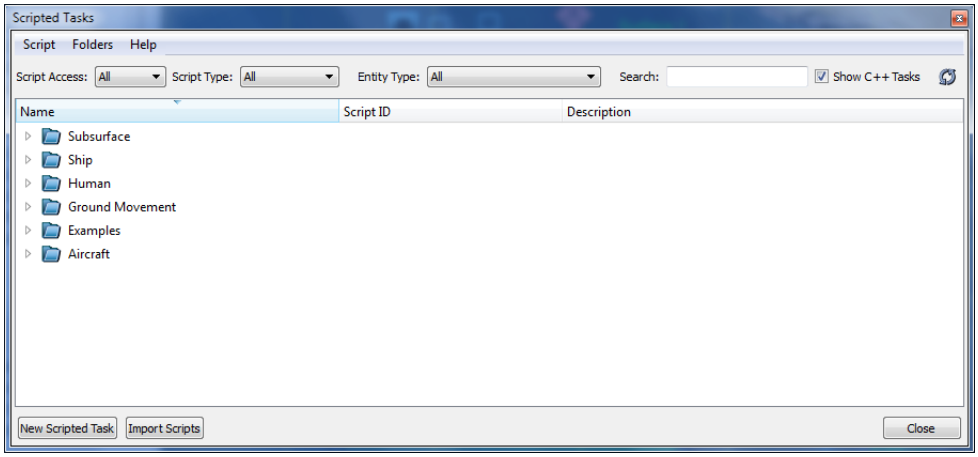


Figure 28-1. Scripted Tasks dialog box

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 Scripted Tasks into Folders," on page 28-29.)

4. Choose **Script** → **New Scripted Task**. The New Scripted Task dialog box opens (Figure 28-2).

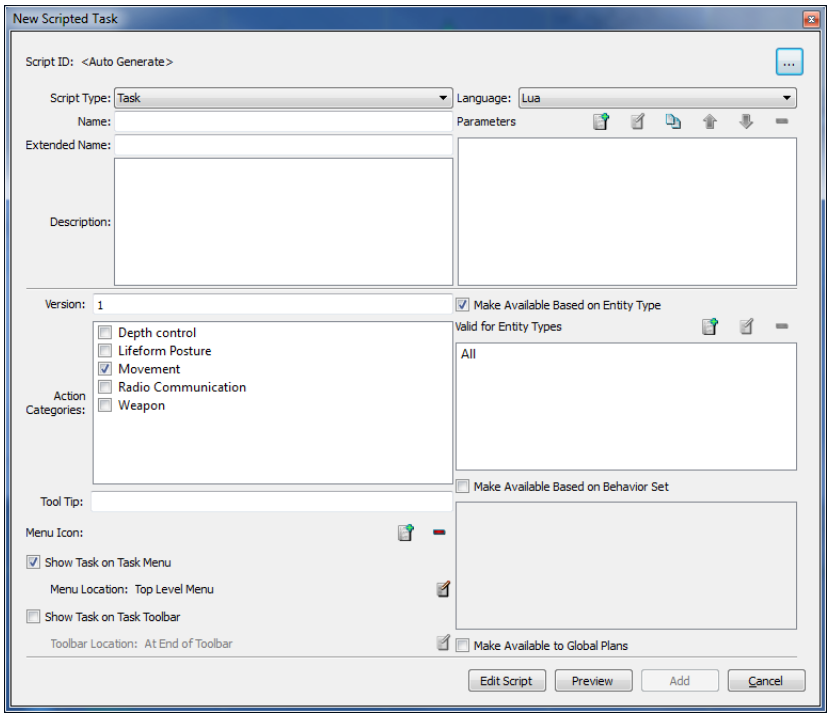


Figure 28-2. New Scripted Task dialog box

5. In the Script Type list, select the type of scripted task you are creating (Task, Reactive Task, or Background Process). The dialog box displays the correct set of metadata for the type of task you chose.

6. Complete the fields in the dialog box. They are described in Table 28-1 and in following sections.

Table 28-1: Script metadata

Input Field	Description
Script ID	A unique name for the scripted task. 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 28-11 for details.
Script Type	One of the following: ◆ Task. A scripted task that would typically be accessed through the Task menu. ◆ Reactive Task. A task that executes in response to specific conditions. ◆ Background Process. A task that executes continuously in the background. The fields on the New Scripted Task dialog box vary based on which type of task is being created.
Language	Specify Lua if you are writing a Lua script for this task. Specify C++ if you have written a new task controller and want to use the scripted task 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 Scripted Task," on page 28-23.
Name	The text that will be added to the Task menu for this script.
Extended Name	An alternative name to use on context-insensitive task lists if more than one task has the same name.
Description	A description of what the task does. This text is displayed on the task's dialog box. (Optional)
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	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.)
Tool Tip	Text for a tooltip when the task is selected on the Task menu. (Optional) (Does not apply to reactive tasks and background processes.)

Table 28-1: Script metadata

Input Field	Description
Menu Icon	The icon to display next to the task name on the Task menu. Please see "Specifying the Task Menu Icon," on page 28-11 for details. (Optional) (Does not apply to reactive tasks and background processes.)
Parameters	The input parameters that the task requires. Please see "Specifying Task Parameters," on page 28-12 for details. (Optional) (Does not apply to background processes.)
Make Available Based on Entity Type	If selected, the task 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 task. Editable only if Make Available Based on Entity Type is selected. This entry affects the context sensitivity of the Task menu. Please see "Specifying the Valid Entity Types for a Task," on page 28-19 for details. Default: All.
Make Available Based on Behavior Sets	If selected, the task 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 task is available on the Task 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 Scripted Tasks," on page 25-18.
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 scripted task that will be used by other scripted tasks, but which you do not want a user to assign directly to a simulation object.) (Does not apply to reactive tasks and background processes.)
Menu Location	The location on the Task menu where this scripted task will be listed. Please see "Specifying the Task Menu Location," on page 28-16 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 scripted task. If you want to display an icon, select a Task menu icon for this task, as described in "Specifying the Task Menu Icon," on page 28-11.
Toolbar Location	The location on the Task Toolbar for this task. You can assign more than one task to a toolbar button. For details, please see "Specifying Equivalent Tasks for a Task Toolbar Button," on page 28-22.

Table 28-1: Script metadata

Input Field	Description
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 scripted task will create.
8. Optionally, click Edit Script to begin writing the code for this scripted task.
9. Click Add. The script is added to the list in the Scripted Tasks window.
10. To save the script task as part of the scenario, save the scenario.
11. Write the code for the scripted task. For details, please see Chapter 29, *Writing Scripts for Scripted Tasks*.

28.2.1. 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 scripted task is executed. If a scripted task is used in a plan, it is referenced by its script ID. If you change the ID, you must ensure that it is unique. You cannot edit the ID of a system scripted task.

To specify the script ID:

1. In the New Scripted Task (or Edit Scripted Task) 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.
3. Click OK.

28.2.2. Specifying the Task Menu Icon

If you want to put an icon on the Task menu next to the menu text, you can specify one. (Does not apply to reactive tasks and background tasks.)

To specify a Task 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 the Task Menu icon for a script, click the Remove button () next to the Menu Icon label.

28.2.3. Specifying an Extended Name

Sometimes a task 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 on the **Commands** → **Task** menu in plans, all tasks are listed and 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 task, it is used on the **Commands** → **Task** menu to distinguish the task from other tasks that have the same name.

28.2.4. Specifying Task Parameters

Scripted tasks can take input parameters. When you specify parameters for a scripted task, VR-Forces automatically builds a dialog box that lets the user enter the parameter values. The parameter types that you can specify for a scripted task are as follows:

- Aggregate Formation. Provides a list of formations for units.
- Altitude MSL. Altitude above mean sea level.
- Bomb Resource. Creates a list of bomb resources.
- Boolean. Creates a check box.
- Choice (List). Specifies choices presented in a drop-down list.
- Choice (Option Button). Specifies choices presented as option (radio) buttons.
- DI-Guy Animation. Lets you select a DI-Guy animation.
- DI-Guy Appearance. Lets you select a DI-Guy appearance.
- Distance. Specifies a distance.
- Emitter. Entity emitter ID.
- Entity Type. Entity type enumeration.
- Force.
- Heading. Heading, in degrees.
- Integer. A whole number.
- 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.

- Offset Location. Specifies an offset to the side, behind, and above, as in the Follow Entity task.
- Real. A floating point decimal 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 Object (Multiple). Lets you select multiple simulation objects in a list window.
- Speed. Any kind of rate, such as speed, climb rate, and so on.
- String. An alphanumeric string.
- Time. Time in days:hours:minutes:seconds.
- Turn Rate. Turn rate in radians/second.

If a scripted task does not have any input parameters, it executes as soon as it is assigned to a simulation object, like the Wait command does.

To specify a parameter for a script:

1. Click the Add button ( ) above the Parameters list. The Add Parameter dialog box opens (Figure 28-3).

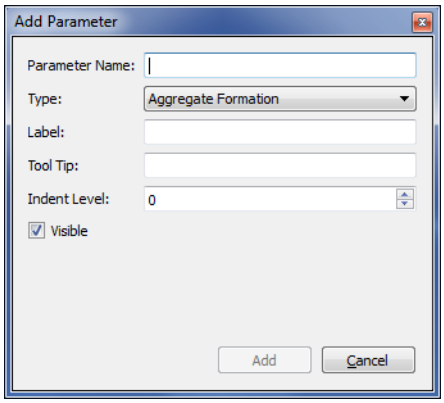


Figure 28-3. Add Parameter dialog box

2. Select the type of parameter from the Type list. If appropriate, the dialog box redisplays to request the details required for the variable type you selected.

3. Fill out the fields as follows:

   All of the parameters have the following fields:

   – 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 scripted task, 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.

   – Tool Tip. Optional text for a tool tip.

   – Indent Level. The number of pixels to indent the parameter on the task dialog box. This allows you to apply some formatting to the dialog box.

   – Visible check box. Specifies that the parameter be included on the scripted task dialog box. (You might want a parameter to be available in the code, but not have it set by a user when the scripted task is assigned.)

   Many parameters let you specify a default value or state.

   Some parameters let you specify a range for the acceptable values:

   – Range Bottom. The lowest acceptable value for this parameter.

   – Range Top. 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.

   For details about adding choices to a Choice parameter, please see "Specifying Choices for the Choice Parameters," on page 28-14. For details about adding categories to a filtered list, please see "Specifying Categories for a Simulation Object," on page 28-15.

Specifying Choices for the Choice Parameters

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.

To add a 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.)

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.

28.2.5. Specifying the Task 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). The default location is the top level of the menu. However, you will probably want to organize your tasks using the submenus provided by VR-Forces or in newly created submenus. The New Scripted Task dialog box displays the menu under which the scripted task is placed. It does not show the full path from the top level menu.



When you add a scripted task to a menu, it does not appear on the menu until you reload the scenario.

To specify a Task menu location for a scripted task:

1. Click the Edit button (). The Select Location on Task Menu dialog box opens (Figure 28-1).

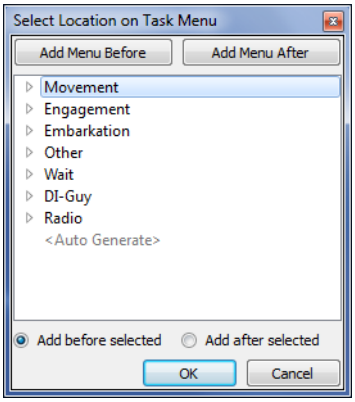


Figure 28-4. Select Location on Task Menu dialog box

2. To add the scripted task to an existing submenu:
   a. Expand the submenu on which you want to put the task.
   b. Select the task that you want the new task to be next to.
   c. To put the new task above the selected menu item, select the Add Item Before Selected option. To put the new task below the selected menu item, select the Add Item After Selected option.
   d. Click OK.

3. To create a new submenu for the scripted task:
   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 scripted task to the new submenu using the procedure in step 2.

28.2.6. 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 Scripted Task dialog box, these groups are represented as action categories. The action categories displayed in the Scripted Tasks 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 70-15.)

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.

28.2.7. Configuring a Script's Availability to a Simulation Object Type

Scripted tasks can be available to a simulation object based on the following criteria:

- The task is available for certain simulation object types. For details, please see "Specifying the Valid Entity Types for a Task," on page 28-19.
- The task is part of a Behavior Set. For details, please see "Specifying Availability by Behavior Set," on page 28-20.
- The task 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 Scripted Task dialog box. For details, please see "Specifying a Scripted Task in a System Definition," on page 28-20.

The default setting for new scripted tasks is that the task is available to all simulation objects.

Availability by simulation object type and by Behavior Set is configured in the New Scripted Task and Edit Scripted Task dialog boxes. Table 28-2 shows the interaction of these settings.

Table 28-2: 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 Entity Types for a Task

You can restrict a scripted task to specific simulation object types. If you do this, the task will not appear on the Task menu when you select a simulation object for which it is not valid. Just as with built-in VR-Forces tasks, there are cases, such as for global plans, where VR-Forces cannot determine the simulation object context and the task will be available on the Task menu even though it is not appropriate to the simulation object that you are tasking. In those cases it is up to you to know that a task is valid.

To specify the valid entity types for a scripted task:

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 28-5). You can build an enumeration by selecting options from the enumeration field lists or you can type in the enumeration directly.

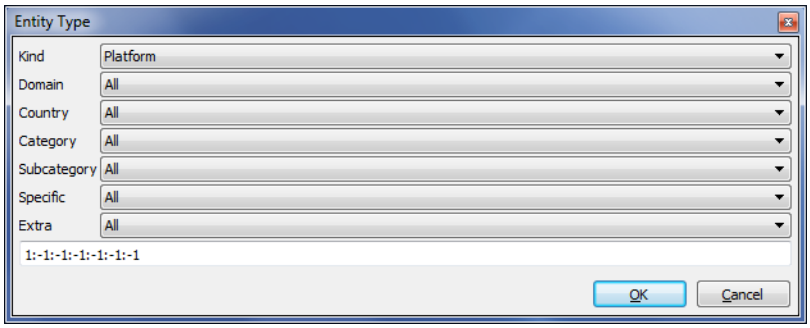


Figure 28-5. Entity Type dialog box

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 scripted task.

Specifying Availability by Behavior Set

You can restrict a scripted task to specific Behavior Sets. If you do this, the task will not appear on the Task menu when you select a simulation object whose force does not have the Behavior Set assigned to it.

To specify that a scripted task 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 scripted task 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 in a System Definition

Some scripted tasks are available to simulation objects because they are specified in a system that the simulation object uses. These tasks are not assigned to simulation object types as part of the scripted task meta data.



This method is used for many of the system scripted tasks supplied with VR-Forces.

Scripted tasks 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 unit-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** parameter:

```
(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")
    )
    ...
)
```

Specifying Equivalent Tasks for a Task Toolbar Button

There may be occasions when you want to have tasks with the same name that work differently depending on the simulation object you give the task 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 task by specifying the simulation object type that the task is available to. To simplify the Task Toolbar, you can map these different tasks to the same toolbar button. This is done by specifying that one task is equivalent to another task.

If you map multiple tasks 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. VR-Forces chooses the correct version of the task. If you select multiple simulation objects that would have different versions of the task, the toolbar button becomes disabled.

To specify an equivalent task for a Task Toolbar button:

1. Create a new scripted task.
2. On the New Scripted Task dialog box (or Edit Scripted Task dialog box), select Show Task on Task Toolbar.
3. Click the Toolbar Location edit button (). The Select Location on Task Toolbar dialog box opens.
4. Select the task that you want this scripted task to be equivalent to. The Equivalent To button becomes active.
5. Click the Equivalent To button. Text is added to the scripted task's entry in the list indicating the task it is equivalent to (Figure 28-6).

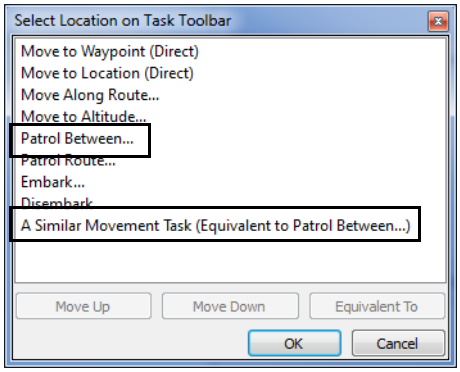


Figure 28-6. Select Location on Task Toolbar dialog box

Do not move the new scripted task to a different location on the toolbar. Moving it breaks the equivalency. Users will click the equivalent task's button to access this task.

6. Click OK.
7. Save the scripted task.

28.2.8. Specifying the Programming Language for a Scripted Task

The Language list in the New Scripted Task dialog box has two options, Lua and C++. In most cases you will choose Lua and write a script to implement the scripted task. However, since the scripted task interface automatically generates task dialog boxes and updates the Task menu, it can be a convenient way to add the GUI component for new tasks 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 to the menu and code a task dialog box.



The Language option only applies to Task 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 in the Scripted Tasks dialog box by selecting the Show C++ Tasks check box (Figure 28-7).

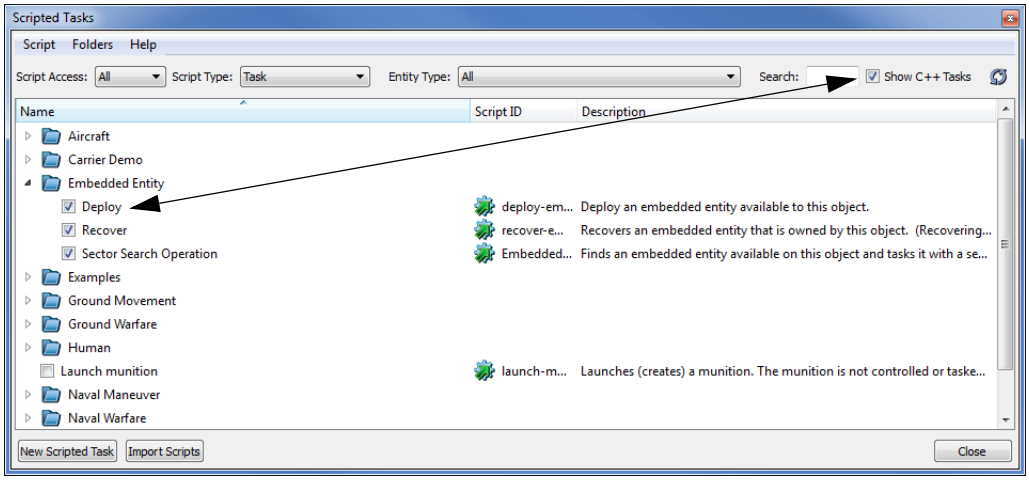


Figure 28-7. C++ tasks

28.2.9. Creating Reactive Tasks

The process for creating a reactive task is largely the same as for creating other scripted tasks. This procedure focuses on the differences.

To create a reactive task:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks dialog box opens (Figure 28-1).
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 Scripted Tasks into Folders," on page 28-29.)
3. Choose **Script** → **New Scripted Task**. The New Scripted Task dialog box opens (Figure 28-2).
4. In the Script Type list, select Reactive Task. The New Scripted Task dialog box redisplays to show the metadata fields for a reactive task (Figure 28-8).

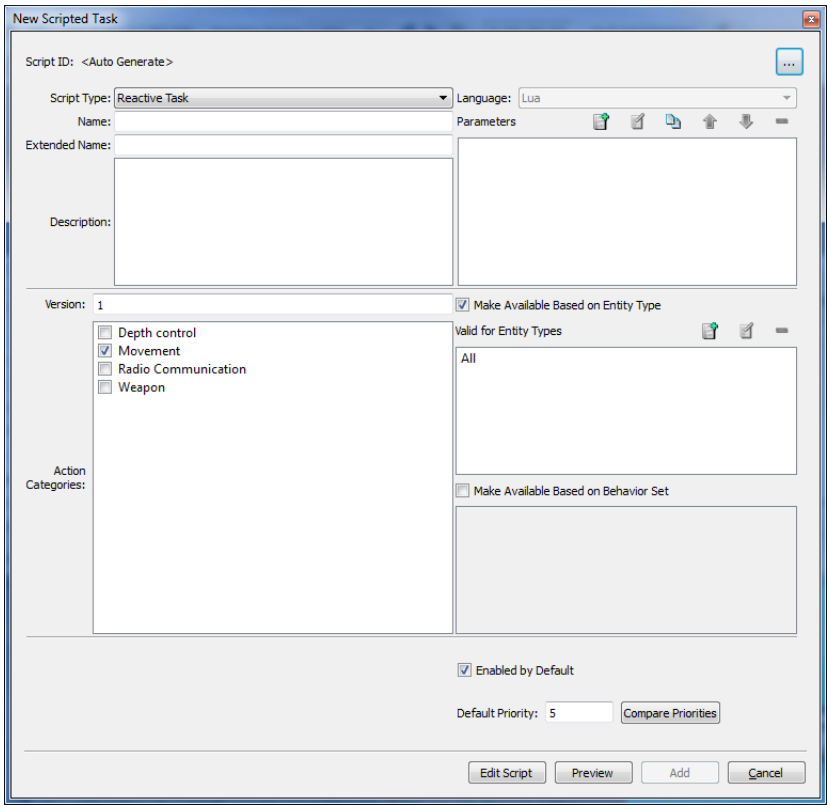


Figure 28-8. New Scripted Task dialog box, reactive task

5. Complete the fields in the dialog box. Fields that are common to all scripted tasks are described in Table 28-1.

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.

28.2.10. Creating Background Processes

The procedure for creating a background process is essentially the same as for other scripted task types. The main difference is that there are fewer fields to complete in the New Scripted Task dialog box. For details about writing a background process script, please see "Background Processes," on page 29-29.

To create a background process:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks dialog box opens (Figure 28-1).
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 Scripted Tasks into Folders," on page 28-29.)
3. Choose **Script** → **New Scripted Task**. The New Scripted Task dialog box opens (Figure 28-2).
4. In the Script Type list, select Background Process. The New Scripted Task dialog box redisplays to show the metadata fields for a background process (Figure 28-8).
5. Complete the fields in the dialog box. Fields that are common to all scripted tasks are described in Table 28-1.

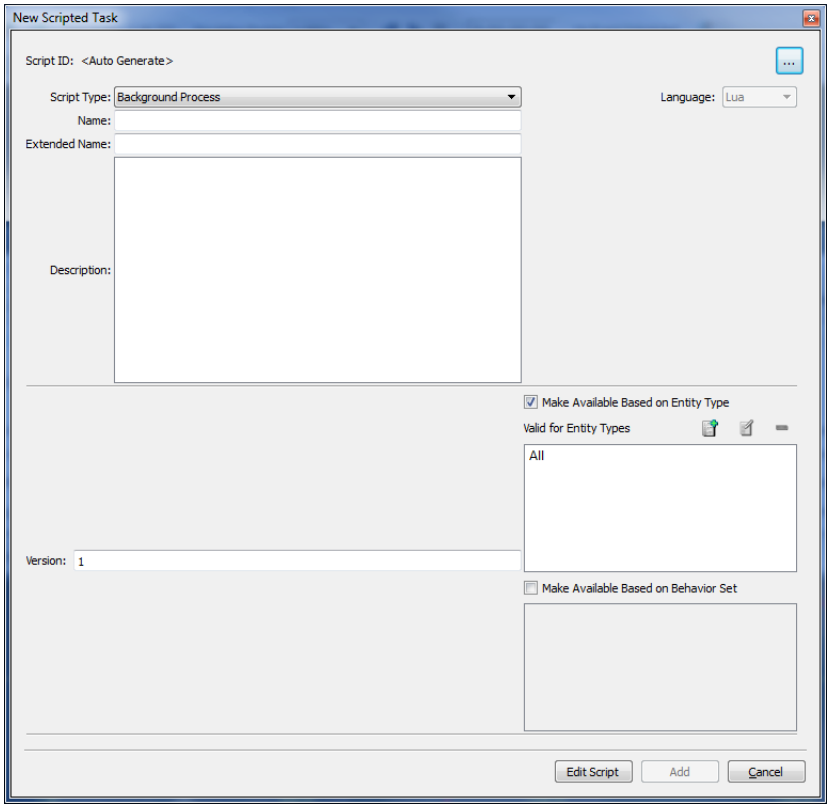


Figure 28-9. New Scripted Task dialog box, background process

6. Click Add.

28.3. Saving Scripted Tasks

When you create or edit a scripted task, its listing in the Scripted Tasks window is in italics, prefaced by an asterisk. When you save the scenario, or promote a scripted task to a system script, the scripted task gets saved and the italics and asterisk get removed.

It is likely that you will test scripted tasks by running a scenario and you will want to edit the scripted task as you find problems. If you create or edit a scripted task 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 scripted task after the scenario rewinds. The scripted task 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 scripted task. You can continue to run the scenario and rewind while you are working on the scripted task. Eventually you will have to save the scenario if you want to save your changes to the scripted task.

- To save a scenario scripted task, save the scenario.
- To save a system scripted task, after updating the task in the Edit Scripted Task dialog box, click Update.

28.4. Editing a Scripted Task

You can edit scripted tasks.

! If you edit a system scripted task, the changes will affect all scenarios that use the scripted task. 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** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
2. Select the scripted task that you want to edit.

---

i In the Script ID column the icon indicates the type of scripted task, Task (), Reactive Task (), or Background Process ().

---

3. Choose **Script** → **Scripted Task**, or double-click the script name. The Edit Scripted Task dialog box opens. (If you are editing a system scripted task, you are prompted to confirm that you want to edit it.)
4. Edit the metadata as desired.
5. Click Update.

28.5. Filtering the List of Scripted Tasks

You can filter the tasks listed in the Scripted Tasks 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, or all scripted tasks.
- 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 scripted task. Some scripted tasks do not have any simulation object types listed. These tasks, 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 scripted tasks to the different platforms.

- Show C++ Tasks: Toggles the display of tasks that are written in C++, but whose GUIs are created using the scripted task mechanism.

28.6. Organizing Scripted Tasks into Folders

You can create folders and organize scripted tasks by folder.

28.6.1. Adding a Folder

To add a folder:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
2. Choose **Folders** → **Add Folder**. The Folder Name dialog box opens.
3. Type a name for the folder.
4. Click OK.

28.6.2. Renaming a Folder

To rename a folder:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
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.

28.6.3. Deleting a Folder

You cannot delete a folder that has scripts in it.

To delete a folder:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
2. Select the folder that you want to delete.
3. Choose **Folders** → **Delete Folder**.

28.6.4. Adding Scripted Tasks to a Folder

➤ To add a scripted task to a folder, drag it onto the folder.

28.6.5. Removing a Scripted Task from a Folder

➤ To remove a scripted task from a folder, drag it away from the folder.

28.7. Exporting and Importing Scripted Tasks

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 scripted tasks from a script package. To change the contents you would have to create a new package.

If scripted tasks are organized in folders, when you save them the folder structure is also saved.

To export scripted tasks:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
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.

28.7.1. Importing a Scripted Task Package

If you import a script package that has multiple scripted tasks, you can choose which scripted tasks to import. Scripts get imported with the folder hierarchy that they were saved in.

To import scenario scripted tasks:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
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 28-10). It lists the scripted tasks saved in the script file.

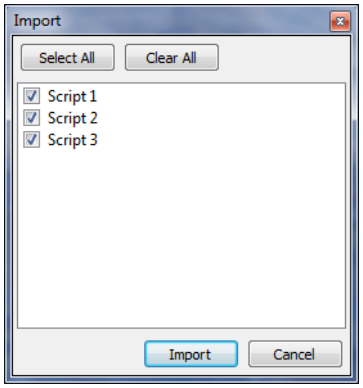


Figure 28-10. Import dialog box

5. Select the scripted tasks that you want to import.
6. Click Import.

28.8. Copying a Scripted Task

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 scripted task:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
2. Select the scripted task that you want to copy.
3. Choose **Script** → **Duplicate**. The New Scripted Task dialog box opens. If you copied a scenario scripted task, 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 scripted task, the system scripted task gets removed from the list of scripted tasks for this scenario and the new scripted task is added as a scenario scripted task. This lets you experiment with changes to a system scripted task without actually changing it. The original system scripted task will still be available to other scenarios.

4. Edit the script.
5. Click Duplicate. The script is added to the list.

28.9. Deleting a Scripted Task

If you delete a scripted task that is assigned to a simulation object in a scenario, the task assignment, whether as an independent task or as part of a plan, will fail. If the scripted task is used as a subtask in another scripted task, that scripted task will fail. VR-Forces does not warn you if either of these cases exist. It is your responsibility to consider the effects of deleting a scripted task.

To delete a scripted task:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
2. Select the scripted task that you want to delete.
3. Choose **Script** → **Delete Scripted Task**.

28.10. Creating a System Scripted Task

To create a system scripted task, you create a scenario scripted task and then save it as a system scripted task. System scripted tasks are saved in *./data/simulationModel-Sets/<model_set>/scripts*. A system scripted task 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 scripted task as a system scripted task:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
2. Select the scripted task that you want to save as a system scripted task.
3. Choose **Script** → **Promote to System Scripted Task**. You are prompted to confirm the change.
4. Click Yes.

28.10.1. Including System Scripted Tasks on the Task Menu

Most of the system scripted tasks included with VR-Forces are listed on the Task menu. The scripted tasks in the Examples folder and a few other tasks are not listed. You can specify whether or not particular scripted tasks are listed. You might want to do this to reduce clutter on the Task menu. Or you might want to limit the tasks available to players in a particular deployment of VR-Forces.

A system scripted task's inclusion on the Task menu is controlled by the following settings:

- The Show Scripted Task on Task Menu check box on the New Scripted Task and Edited Scripted Task dialog boxes.
- The check box next to the scripted tasks entry on the Scripted Tasks dialog box (Figure 28-11).
- Its membership in a behavior set. The effect of behavior sets on task availability is additive to the configuration discussed in this section. For details about using behavior sets, please see "Using Behavior Sets to Manage Scripted Tasks," on page 25-18.

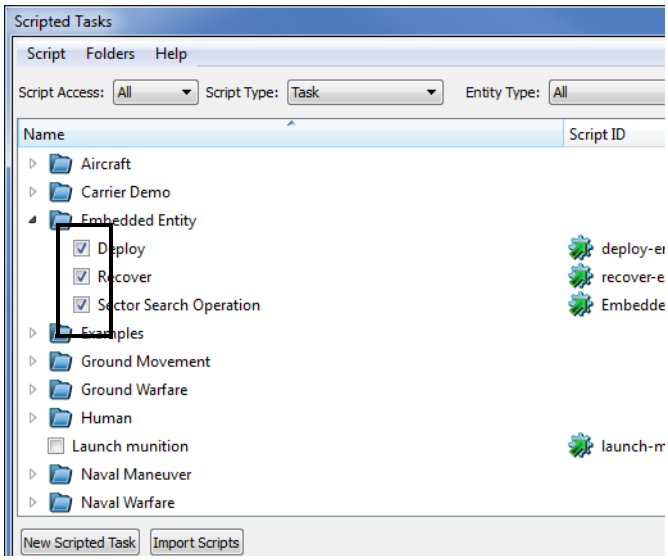


Figure 28-11. Show scripted task on Task menu check box

Table 28-3 shows when a scripted task is listed on the Task menu based on the configuration check boxes selected.

Table 28-3: System scripted task inclusion on Task menu

Show Scripted Task on Task Menu check box	System scripted task list check box	Task included on Task menu
☑	☑	Yes
☑	☐	No
☐	☑	Yes, for this scenario.
☐	☐	No

To specify the availability of a system scripted task on the Task menu:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
2. In the list of system scripted tasks, select the scripted task whose availability you want to change. Use Table 28-3 to determine which check box configuration you need for your desired Task menu status.
3. Choose **Script** → **Scripted Task**. The System Script Editing prompt opens. (If you have disabled this prompt, the Edit Scripted Task dialog box opens. Skip to step 5.)
4. In the System Script Editing prompt, click Yes. The Edit Scripted Task dialog box opens.
5. Select or clear the Show Scripted Task on Task Menu check box.
6. Click Update.
7. In the list of scripted tasks, select the check box next to the system scripted task you are editing to make it available. Clear the check box to make it unavailable.

28.11. 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 Scripted Task Options page (Figure 28-12).

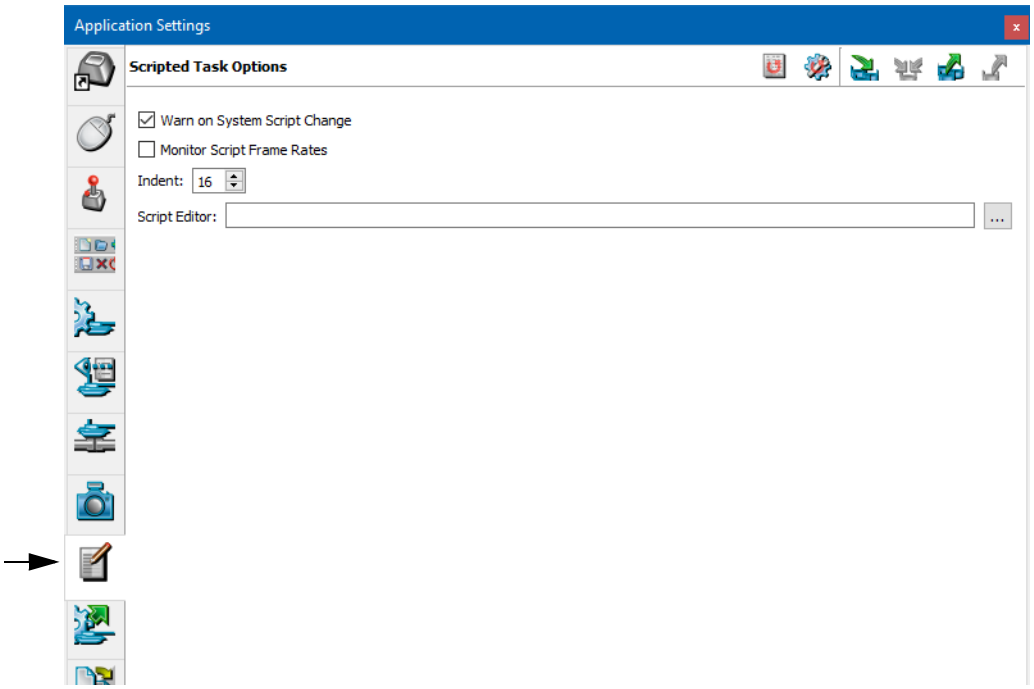


Figure 28-12. Scripted Task Options page

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.

28.12. Editing Lua Files

To edit the scripts for scenario scripted tasks you need to work through the Scripted task 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 scripted task rather than by editing the Lua file. However, VR-Forces includes some libraries of Lua functions that are independent of any of the scripted tasks. They are available to be included in your scripted tasks. 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 Scripted Tasks dialog box.

To edit a Lua file from the Scripted Task dialog box:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks window opens (Figure 28-1).
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.



29. Writing Scripts for Scripted Tasks

This chapter explains how to write Lua scripts for scripted tasks.

The VR-Forces Lua Interface 29-3
- Lua Classes 29-3
- Lua Modules 29-6

Script Loading and Execution 29-7
- Script Entry Points 29-7
- The Scripted Task Execution Sequence 29-9
- Limitations for Checkpointing Scripted Tasks 29-12
- Editing Scripted Tasks While a Scenario is Running 29-14

Tasks and Subtasks 29-15
- Monitoring the Status of Tasks and Subtasks 29-16
- Stopping Tasks 29-16
- Task Parameters 29-16

Geometry 29-18
- Location3D 29-18
- Vector3D 29-18
- VectorOffset3D 29-20
- VectorGeoc3D 29-20

Coding Messages for Translation 29-21

Error Detection 29-21

Interactive Scripts 29-22

A Basic Scripted Task 29-23
- Create and Move to Waypoint Metadata 29-24
- The Create and Move to Waypoint Lua Script 29-27

A Simple Reactive Task 29-28

Background Processes 29-29

29.1. 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 Start menu, or in *./doc/luadoc/index.html*.

29.1.1. 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 Entity 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 29-18.
- *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.
- *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.)
- *SimObject*. *SimObject* is a class that represents simulation objects in 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.
- *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 29-1 lists reserved words (in addition to those reserved by Lua), which should not be used except for their intended purpose.

Table 29-1: 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

29.1.2. 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.

29.2. Script Loading and Execution

This section describes script entry point functions and the execution sequence for scripted tasks. It also describes how VR-Forces checkpoints scripted tasks and executes them when a checkpointed scenario is run.

29.2.1. 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 Scripted Tasks," on page 29-12.) 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.

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.

29.2.2. The Scripted Task Execution Sequence

A Lua script has several distinct sections, including the entry points described in "Script Entry Points," on page 29-7 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 29-1 illustrates the control flow for a scripted task.

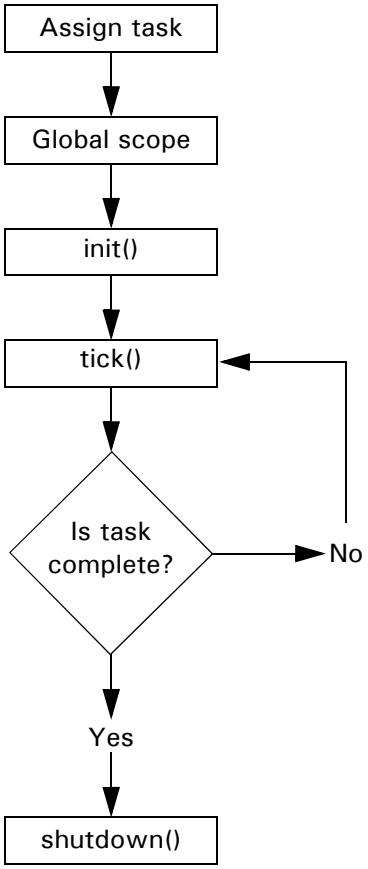


Figure 29-1. Scripted task control flow

The Script Execution Sequence for Reactive Tasks

Figure 29-2 illustrates the script execution sequence for reactive tasks. It is distinguished from that for scripted tasks by the addition of the checking process.

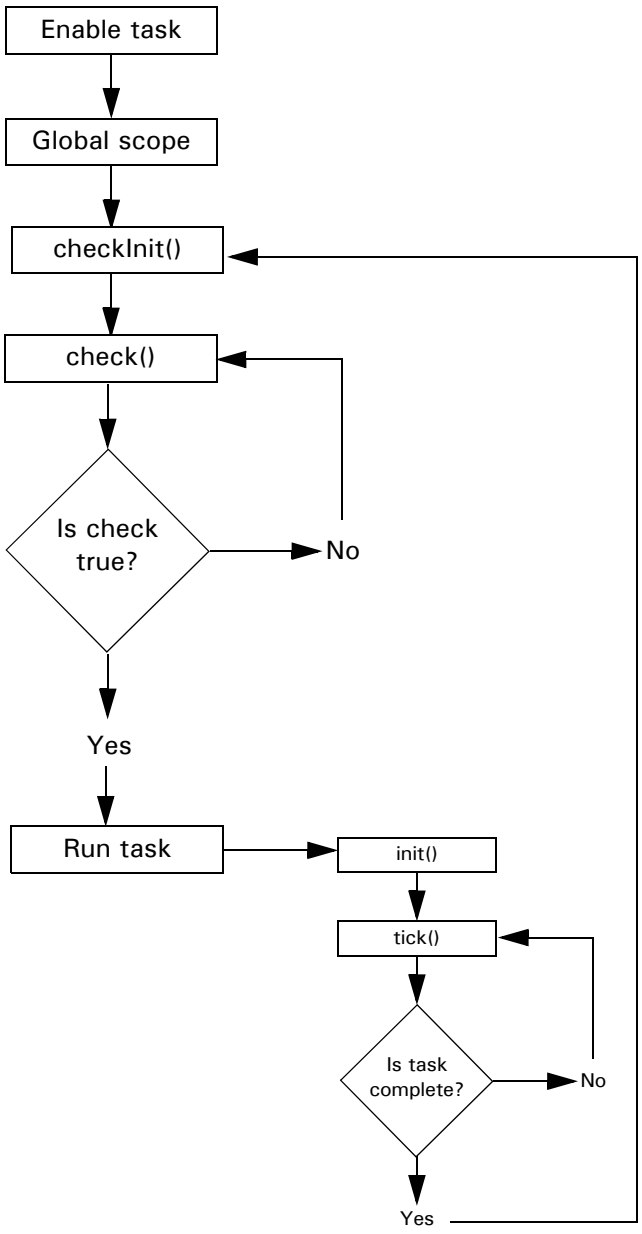


Figure 29-2. Reactive task control flow

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 Scripted Tasks," on page 29-12. For general details about checkpointing, please see "Saving a Scenario," on page 7-27.) 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
```

29.2.3. Limitations for Checkpointing Scripted Tasks

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 29-3).
- 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.

29.2.4. Editing Scripted Tasks While a Scenario is Running

You can add, remove, and edit scripted tasks during scenario creation or while a scenario is running. If a scripted task is not being used by any simulation objects, editing or deleting it has no effect on the scenario. If you change a scripted task that is being executed by a simulation object, when you save the change the scripted task may be stopped, restarted, or continue to execute depending on what has changed.

Table 29-2 lists the effect on scripted tasks and scripted tasks running as subtasks if you edit the code or metadata:

Table 29-2: Side effects of editing scripted tasks

Change to scripted task	Result
Change code or any metadata except the script ID. Entity type is still supported.	The task restarts. Changes to the Action Category take effect when scenario restarts.
Entity types supported changes.	The task stops for simulation objects that no longer support it.
Script ID changes.	The task continues to execute. Any other 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 scripted task, then it is added to or removed from the Task menu for the appropriate simulation object types. (This change takes place immediately.)

29.3. 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 Scripted Task 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.

29.3.1. 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).

29.3.2. 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).

29.3.3. 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.

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 29-3 lists the parameter types you can specify when you create a scripted task and the corresponding Lua types in the taskParameters table.

Table 29-3: Task parameters and Lua types

Parameter Type	Lua Type
Aggregate Formation	String
Altitude MSL	Number (meters)
Bomb Resource	String (resource name)
Boolean	Boolean
Choice (List)	Number (Index of the option chosen)
Choice (Option Button)	Number (Index of option chosen)
DI-Guy Animation	String
DI-Guy Appearance	String
Distance	Number (meters)
Emitter	Number (Index of the emitter system)
Simulation Object Type	String (DIS type enumeration, colon separated)
Force	String
Heading	Number (radians)
Integer	Number
Location (with altitude)	Location3D
Location (without altitude)	Location3D
Munition Resource	String (resource name)
Offset Location	VectorOffset3D
Real	Number
Resource	String (resource name)
Simulation Object (Single)	SimObject
Simulation Objects (Multiple)	Table of SimObjects
Speed	Number (meters / second)
String	String
Time	Number (seconds)
Turn Rate	Number (radians / second)

29.4. 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.

29.4.1. *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.

29.4.2. *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 29-4). 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.

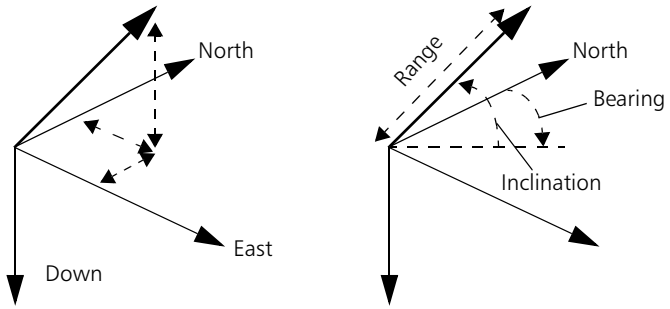


Figure 29-3. Vector3D: North-East-Down, or Bearing-Inclination-Range

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 29-3 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.

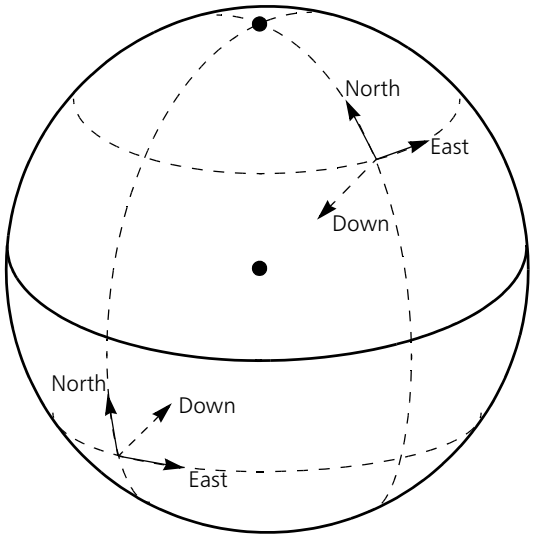


Figure 29-4. Coordinate systems with different origins

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.

29.4.3. *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.

29.4.4. *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.

29.5. Coding Messages for Translation

VR-Forces has a utility, translationFileCreate, that can extract the user-visible text in a scripted task 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*).

29.6. Error Detection

The VR-Forces Lua implementation can detect syntax errors and runtime errors.

When you save the script for a scripted task, it is checked for syntax errors. If an error is found, it is reported in an error message. In Figure 29-5, the `if` statement in line 7 is missing the `then` keyword.

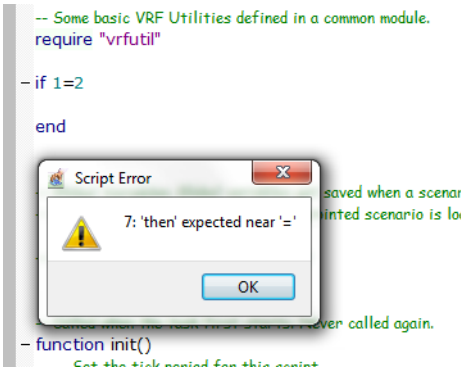


Figure 29-5. Lua syntax error



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 scripted task, the back-end will generate an error.

When you run a scripted task, 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 Scripted Tasks dialog box (Figure 29-6). Therefore, you may want to keep the dialog box open while you are testing new or edited scripts.

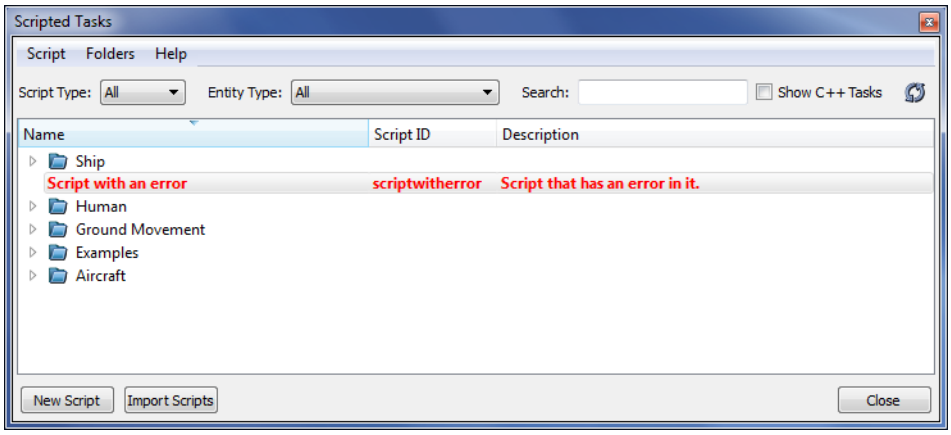


Figure 29-6. Lua error message

To view an error message:

1. In the Scripted Tasks 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.

29.7. 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.

29.8. 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.

29.8.1. Create and Move to Waypoint Metadata

Figure 29-7 shows the metadata for the Create and Move to Waypoint scripted task.

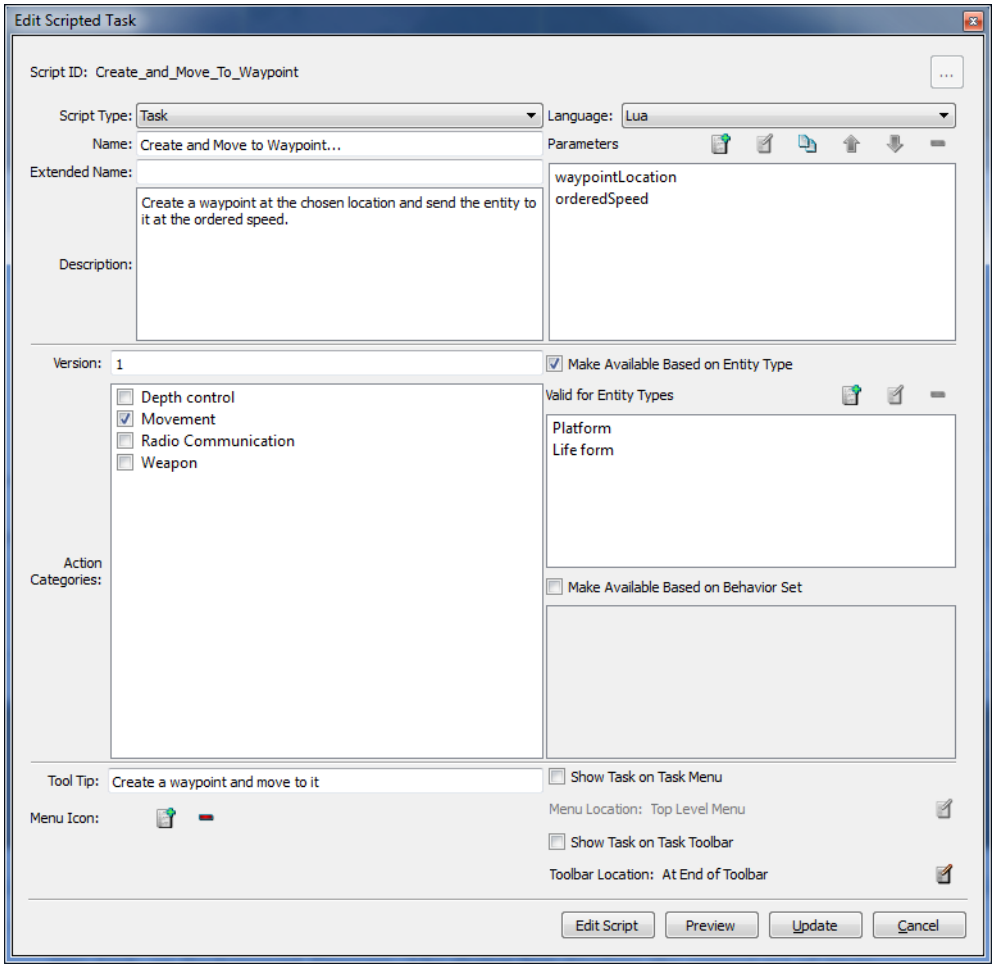


Figure 29-7. Create and Move to Waypoint scripted task

To view the metadata:

1. Choose **Simulation** → **Scripted Tasks**. The Scripted Tasks dialog box opens.
2. Expand the Examples folder (Figure 29-8).

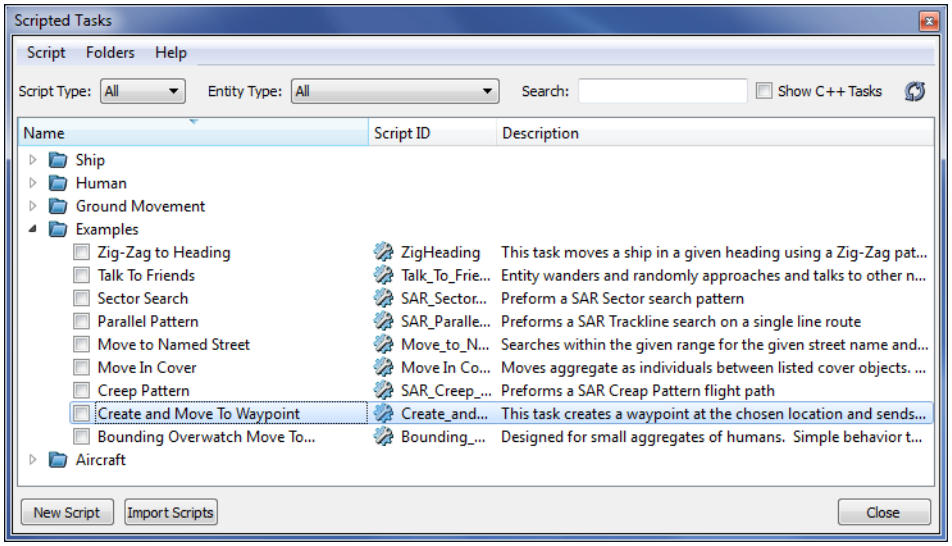


Figure 29-8. Scripted Tasks dialog box

3. Select Create and Move to Waypoint.
4. Choose **Script** → **Scripted Task**. The Edit Scripted Task dialog box opens (Figure 29-7).

Note how the Name column of the Create and Move to Waypoint entry in the Scripted Tasks dialog box matches the Menu Text in the Edit Scripted Task dialog box. The Script ID and Description fields are also drawn from the scripted tasks metadata.

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 29-9 shows the dialog boxes that specify the parameters.

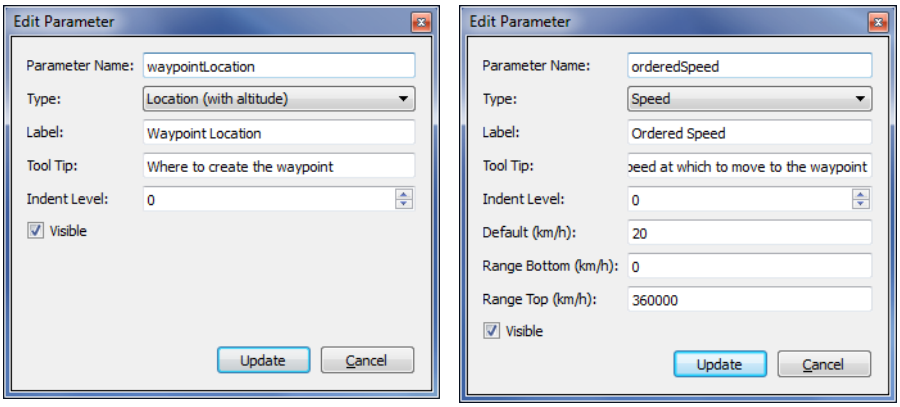


Figure 29-9. Parameter definitions

Figure 29-10 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.

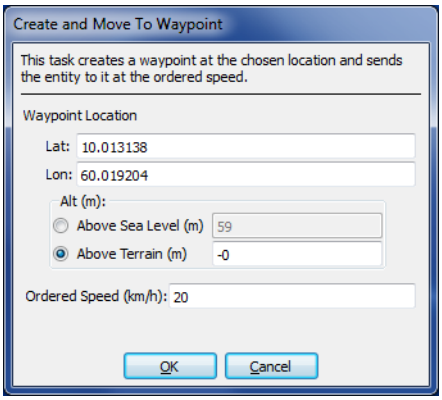


Figure 29-10. Create and Move to Waypoint dialog box

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.

29.8.2. The Create and Move to Waypoint Lua Script

The entire Lua script for this scripted task (without comments) is as follows:

```
require "vrfutil"
moveToWaypointSubtaskId = -1
function init()
  vrf:setTickPeriod(0.5)

  waypoint =
    vrf:createWaypoint({location=taskParameters.waypointLocation,
    force=this:getForceType()})

  vrf:sendSetData(this, "set-speed",
    {speed=taskParameters.orderedSpeed})
end
function tick()
  if moveToWaypointSubtaskId == -1 then
     moveToWaypointSubtaskId = vrf:startSubtask("move-to",
      {control_point=waypoint:getName()})
  end
  if vrf:isSubtaskComplete(moveToWaypointSubtaskId) then
     vrf:endTask(true)
  end
end
function shutdown()
  vrf:deleteObject(waypoint)
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(). The first time the task is ticked, it sends the Move to Waypoint subtask. Thereafter, each time the task is ticked, it checks to see if the subtask is complete. When the subtask is complete, 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 29-3. The createWaypoint() function 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 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.

29.9. 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
```

29.10. 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 Scripted Tasks 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
```



30. Setting Simulation Object State

This chapter explains how to set simulation object state, such as speed, force, and target.

Setting Simulation Object State and Attributes 30-4
Activation 30-5
Active Sonar Mode 30-6
Activity 30-6
Aiming 30-7
Altitude 30-7
Ammunition 30-8
Ammunition Pacing/Tracking 30-8
Appearance 30-9
Armed 30-9
Capabilities 30-10
Collision Avoidance Types 30-11
Concealed 30-12
Contamination 30-12
Counter Measures Auto Launch 30-12
Current Speed 30-13
Destroyed 30-13
Detonation Fuse Type 30-14
DI-Guy Appearance 30-15
DI-Guy Hand Item 30-15
Disembarked 30-16

Embarked 30-16
Emitter 30-17
Engagement Results 30-18
Equipment Pacing/Tracking 30-19
Food, Water, Fuel, Oil and Lubricant 30-19
Food, Water Fuel, Oil and Lubricant Pacing/Tracking 30-20
Force 30-21
Formation 30-22
Heading 30-23
Health 30-23
IFF 30-24
Jam Targets 30-25
Lase Autonomous 30-25
Laser Code 30-26
Location 30-26
MOPP Level 30-27
Morale 30-27
Navigation Preferences 30-28
Notify Level 30-28
Ordered Speed 30-29
Percent Complete 30-29
Posture (Unit) 30-30
Posture (Human) 30-31
Preferred Targets 30-32
Radar Mode 30-33
Reorganize 30-33
Resources 30-34
Resources Pacing/Tracking 30-34
Restore 30-35
Resupply Mode 30-35
Rules of Engagement 30-36
    How Fire-When-Fired-Upon Works 30-36
Sector of Responsibility 30-37
Sensor Aim 30-38
Sensor Zoom 30-38

Sonar Depth 30-39
Spot Reports 30-40
Superior 30-41
Supplying 30-41
Surrendered 30-41
Synchronize Laser Code 30-42
Target 30-42
Tasked by Superior 30-43
Tracer Use 30-43
Unit State 30-43
Weapons Pacing/Tracking 30-44

30.1. Setting Simulation Object State and Attributes

Some set data requests only apply to entity-level scenarios. Others only apply to aggregate-level scenarios. These set data requests indicate the applicable simulation type with an entity ( ) or unit ( ) icon in the margin.

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, except for the Altitude, IFF, Location, and Heading dialog boxes, which show the current values for the selected simulation object.



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 tasks are available on the Sets Toolbar and on the Last Selected Objects Panel.

30.2. Activation

The Activation set data request lets you activate (Display) or deactivate (hide) tactical graphics. This is strictly a visual effect. The 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 35-18 and "Tactical Graphic Active," on page 31-14.

To activate or deactivate a tactical graphic:

1. Select the tactical graphic.
2. Choose **Set** → **Disposition** → **Activation**. The Activation dialog box opens (Figure 30-1).

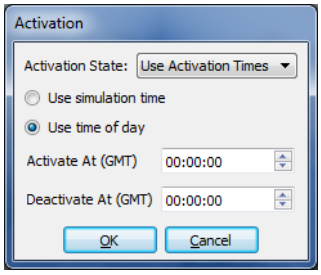


Figure 30-1. Activation

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 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.

30.3. 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.

30.4. Activity

You can give a simulation object in an aggregate-level scenario 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.

30.5. 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.

30.6. 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.

30.7. Ammunition

You can specify the amount of each type of ammunition that a simulation object in an aggregate-level scenario 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.

30.8. 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.

30.9. 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-010-2010-RC1.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 30-31.



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.

30.10. 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.

30.11. Capabilities

The capabilities that are available for a simulation object are determined by its entry in *./appData/settings/vrfGui/SISO-REF-010-2010-RC1.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.

30.12. 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 change 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.

30.13. 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 for a simulation object:

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.

30.14. 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.

30.15. 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.

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.

30.16. 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 30-29) 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.

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.

30.17. 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 a simulation object's state to destroyed:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Destroyed**. The simulation object is immediately destroyed.

30.18. 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 30-9.)

> IEDs, car bombs, and suicide bombers have radios. Therefore, you can detonate them using the Send Radio Set task.

To configure 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.

30.19. DI-Guy Appearance

The DI-Guy Appearance 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 Appearance set data request applies only to lifeform entities.

To set an entity's DI-Guy appearance:

1. Select the entity.
2. Choose **Set** → **Appearance** → **DI-Guy Appearance**, or click the DI-Guy Appearance button () on the Sets Toolbar. The DI-Guy Appearance dialog box opens.
3. Select an appearance from the list.
4. 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.
5. Click OK.

30.20. 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** → **DI-Guy 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.

30.21. 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**.

30.22. 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** → **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 a embarkation offset.
5. Click OK.

30.23. 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 70-11, for an explanation of how to configure individual and multiple emitters in the object parameter database.
- Please see the online help for the OPD Editor for a description of emitter parameters.

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.

30.24. 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.

30.25. 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.

30.26. Food, Water, Fuel, Oil and Lubricant

You can set the resources for a simulation object in an aggregate-level scenario.

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.

30.27. 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, and fuels, 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.

30.28. 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 a simulation object:

1. Choose **Set** → **Disposition** → **Force**. The Force dialog box opens.
2. Select a force from the list.
3. Click OK.

30.29. 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/simulationModel-Sets/<model_set>/vrfSim/formation*. Formation files have the extension *.frm*. In addition to the named formations provided by MÄK, 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 70-8.

30.30. 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 16-17.

To set a simulation 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.

30.31. 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 22-2.

To set a simulation 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.

30.32. 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.

To set the IFF transponder for a fixed-wing entity that is configured with an IFF controller:

1. Select the entity.
2. Choose **Set** → **Sensors** → **IFF**. The IFF dialog box opens (Figure 30-2).

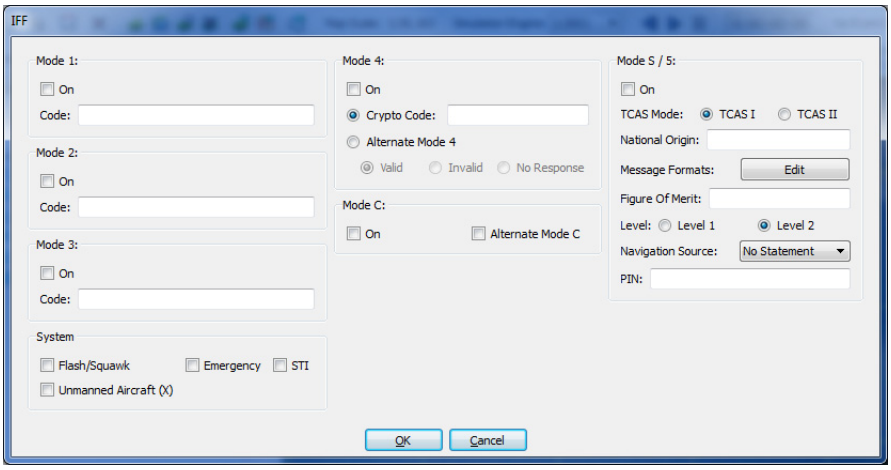


Figure 30-2. IFF dialog box

3. Complete the IFF dialog box.

30.33. Jam Targets

The Jam Targets set data request orders an aircraft to jam specific targets or all simulation objects within its jamming beam. This set data request does not affect the simulation. It sends update messages that can be used by other applications in a simulation.

To jam targets:

1. Select the entity that will send out the jamming beam.
2. Choose **Set** → **Sensors** → **Jam Targets**. The Jam Targets dialog opens.
3. In the Mode group box, select All Entities in Beam or Selected Entities.
4. If you selected Selected Entities, select the targets you want to jam.
5. Click OK.

30.34. 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.

30.35. 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 9-15.

To specify an entity's laser code:

1. Select the entity.
2. Choose **Set** → **Laser Designator** → **Laser Code**. The Laser Code dialog box opens.
3. Type a number between 111 and 8888, using only the digits 1-8.
4. Click OK.

30.36. 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 16-19. For ground entities, specifying an altitude lets you place an entity at different levels (floors) in building features.

To set the location for a simulation object:

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 16-14.
5. Click OK.

30.37. 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.

30.38. 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.

30.39. 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.

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.

30.40. Notify Level

You can set the notification level for entity 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.

30.41. 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.



- 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 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 30-13).

To set the ordered speed of a simulation 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.

30.42. Percent Complete

Percent Complete lets you set the degree of completion for combat engineering objects in aggregate-level scenarios.

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.

30.43. Posture (Unit)

You can set the posture of a simulation object in an aggregate-level scenario. The posture changes instantly. To change posture over time, use the Change Posture task.

To set a simulation 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.

30.44. 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 30-1 lists the position and movement relationships for each posture.

Table 30-1: 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** → **Appearance** → **Posture**. The Posture dialog box opens.
3. Select a posture from the list.
4. Click OK.

30.45. 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.

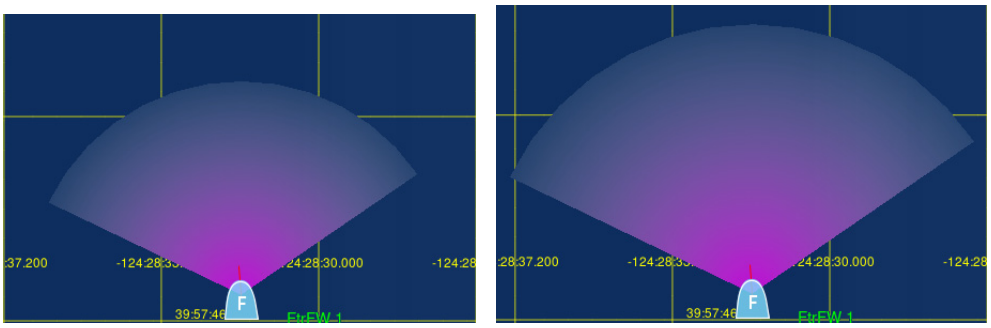
To specify preferred targets:

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.

30.46. 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 30-3 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.)



Search Track

Figure 30-3. Radar mode beams

For more information about emitter beams, please see "Emitter," on page 30-17 and "Configuring Emitters," on page 70-11.

To set an entity's 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.

30.47. 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.

30.48. 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 value:

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.

30.49. 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** → **Resource Pacing/Tracking**. The Resources Pacing/Tracking dialog box opens.
3. Select Resources.
4. Click OK.

30.50. 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 a simulation object:

1. Select the simulation object.
2. Choose **Set** → **Disposition** → **Restore**.

30.51. Resupply Mode

Resupply Mode specifies one of the following resupply modes for simulation objects in an aggregate-level scenario:

- 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 22-14.
- 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** → **Set 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.

30.52. Rules of Engagement

Rules of engagement specify the conditions under which a simulation object will fire at the enemy.



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 for a simulation object:

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.

30.52.1. 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.

30.53. 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 70-2.

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 30-4 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.

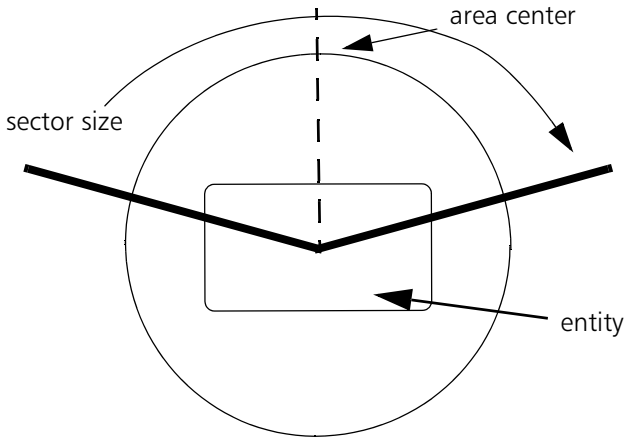


Figure 30-4. Sector of responsibility parameters

To set a simulation 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.

30.54. 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 46-10.

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.

30.55. 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 46-10.

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.

30.56. 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.

30.57. Spot Reports

You can set spot reports on or off for all simulation objects (global setting) from the Spot Report Options 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 Options page. For more information about spot reports, including setting them globally, please see "Displaying Simulation Objects Based on 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 for a simulation object:

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. This option can improve performance.
   - 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.

30.58. 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 a simulation 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.

30.59. 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.

30.60. 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.

30.61. 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.

30.62. 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.

30.63. 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 a simulation object to be tasked by its superior:

1. Select the simulation object.
2. Choose **Set** → **Other** → **Tasked by Superior**.

30.64. 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 User Tracers list.
4. Click OK.

30.65. Unit State

The Unit State request lets you change an entity-level unit from the aggregated state to the disaggregated state and vice versa.

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.

30.66. 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.



VI. Plans

VI.1. Introduction to Plans

VR-Forces has two kinds of plans, global plans and individual plans. A plan 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.

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.

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.

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.

Plans are discussed in detail in Chapter 32, *Writing Plans*.



31. Plan Concepts

This chapter explains how to combine tasks, set data requests, global commands, and conditions into plans.

Introduction to Individual Plans and Global Plans 31-2
Conditional Statements 31-4
The If Statement 31-6
When Statements (Triggers) 31-7
While Statements 31-9
Conditional Tests 31-10
Viewing Plans 31-14
Considerations for Creating Plans 31-14
Considerations for Using Triggers 31-15
Moving In Formation 31-16
Using the Tasked-By-Superior Request in a Plan 31-16
Following Simulation Objects 31-16
Planning Tasks for Aircraft 31-17
Using Non-VR-Forces Simulation Objects in Plans 31-17

31.1. 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 31-1 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).

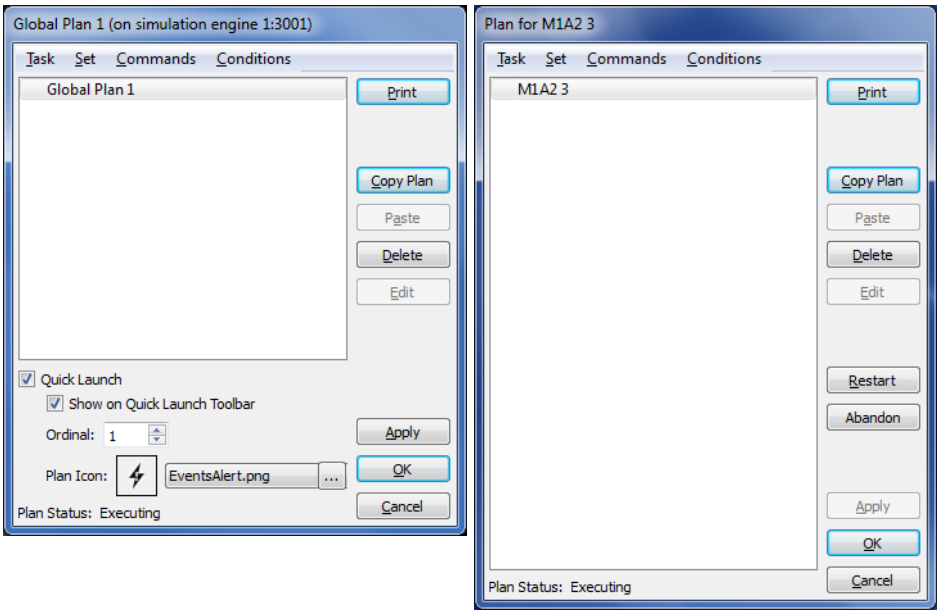


Figure 31-1. Global plan and individual plan

The Commands menu, which is common to both types of plans, contains global commands. These commands allow you to:

- Create and delete objects.
- Assign tasks and set data requests to simulation objects.
- Send console messages.
- Change hostility.
- Issue plans to simulation objects.
- Abandon and restart plans.
- Change time of day and weather settings.
- Trigger scenario events.
- Send view control messages.

Global commands must explicitly state the simulation object to which they apply.

31.2. 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 more information, please see "The If Statement," on page 31-6.
- `When`. A `When` statement is also called a trigger. A trigger statement lets you plan for a condition without knowing in advance when the condition will occur. A trigger specifies a condition that is evaluated continuously by VR-Forces. If the condition becomes true, VR-Forces executes the statements in the trigger block. For more information, please see "When Statements (Triggers)," on page 31-7.
- `While`. A `While` statement executes a block of statements repeatedly while the 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 more information, please see "While Statements," on page 31-9.

The conditions supported by VR-Forces 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 Line)
- 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? (Entity Detected)
- Has this simulation object received a radio 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? (Sim Time)
- 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 more information, please see "Conditional Tests," on page 31-10.

31.2.1. 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 `When` statement 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.

31.2.2. When Statements (Triggers)

A trigger consists of a `When` statement followed by a block of statements that are executed only if the condition in the `When` statement is true.

When VR-Forces reads a `When` statement, it registers the trigger and checks the condition. If the condition is true, control passes to the first statement in the `When` block. Task execution proceeds as follows:

- If the `When` block contains any tasks that are mutually exclusive with the currently executing task, the current task is discarded and the statements in the `When` block are executed.
- If the tasks in the `When` block are not mutually exclusive with the current task, or if the `When` block only has set data request statements, the `When` 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 31-15.
- After the tasks in the `When` statement block are completed, control returns to the next statement in the simulation object's plan.



The location of a `When` statement in a plan has an important affect on its functioning. For a discussion about where to place triggers, please see "Deciding Where to Put Trigger Statements," on page 31-15.)

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. Task execution proceeds as described in the previous paragraph.

Figure 31-2 illustrates the process when a trigger has a task that is mutually exclusive with the current task. After the tasks in the `When` statement block 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.

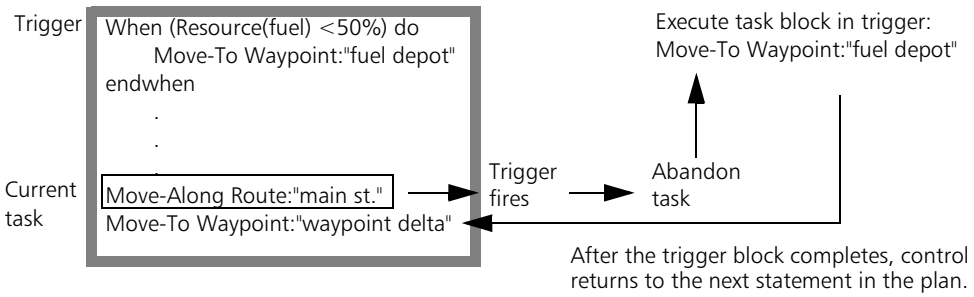


Figure 31-2. Flow of control for triggers

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 need to execute another `When` statement for that condition.

Unregistering Triggers

You cannot unregister triggers while a scenario is running. If you do not want a trigger to be executed, you have the following options:

- Before you start the scenario, remove the trigger from the plan.
- During a simulation, abandon the plan.
- During the simulation, remove the trigger from the plan and save the plan. The new plan will erase the old one. However, the plan will begin executing at the first statement, which may not be the behavior you want.



An alternative to using triggers in a plan is to write a reactive task. Reactive tasks reset after they run, which overcomes the problem of not being able to reregister a trigger. For information about reactive tasks, please see Chapter 28, *Creating Scripted Tasks*.

31.2.3. 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 `If`, `When`, and `While` statements.

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 the tasks in the block will never be completed.

31.2.4. 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 70-2.



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 31-3. The condition in M1A2 1's plan is true when BMP 1 has a target, not when M1A2 1 targets BMP 1.

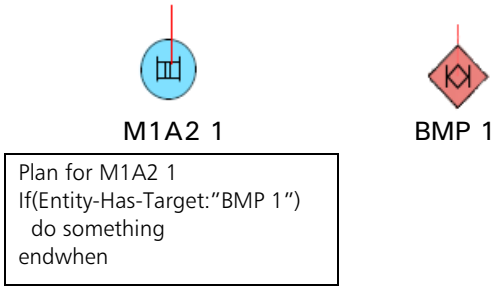


Figure 31-3. Entity Has Target condition

> 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` boolean 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.

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 33-4.

Entity Under Fire

Entity Under Fire returns true if the simulation object is under fire.



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.

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/else` 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 using comparison operators.

Receive Text Message

The Receive Text Message condition tests to see if this simulation object has received a text message (sent by the Send Text Message task) that contains the text specified in the Receive Text Message dialog box.

Scenario Event

The Scenario Event condition returns true if the specified scenario event is in process or has completed.

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 35-18 and "Activation," on page 30-5.

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.

You can nest and combine conditional expressions.

31.3. Viewing Plans

You can view a simulation object's plan and observe its progress through the plan.

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. Choose **Objects** → **Plan**, or press **p**. The Plan window opens.

31.4. 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.

31.4.1. 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 Trigger Statements

VR-Forces does not evaluate a trigger condition until it executes a `When` statement and registers the trigger. If you want to ensure that a `When` statement always gets read by VR-Forces, put it at the beginning of a plan before any non-trigger statements. If you put a trigger later in a plan, it is possible that it will never be registered. This could occur if a plan is abandoned. Of course, you might decide that you do not want a trigger to be registered 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 place 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.

Reregistering Triggers

A trigger can only fire once. It does not stay in effect for a re-occurrence of the triggering event. Under limited circumstances it is possible to work around this limitation. If a simulation object's plan only contains `When` statements, you could put a Restart Plan command at the end of each `When` block. In this way, after a trigger fires, it executes the commands in the trigger block and then it restarts the plan. This re-registers all of the `When` statements in the plan.

If a plan has commands outside of a trigger, then restarting the plan would also cause the simulation object to execute those commands, which might not be desirable. You might be able to work around this issue by putting these commands in a `When` block related to the simulation time.

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.

31.4.2. 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.

31.4.3. 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.

31.4.4. 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.

31.4.5. Planning Tasks for Aircraft

When you plan tasks for aircraft, the most important thing to remember is to pay attention to the terrain. Fixed-wing aircraft do not have terrain following capabilities. Rotary-wing aircraft have some terrain-following capacity, but it has some limitations. So if you assign aircraft an altitude that is lower than the terrain at some point along a route or the path they must follow to reach a waypoint, they will crash into the terrain. View the terrain profile for a route to see if it intersects the terrain.

You should also bear in mind that a fixed-wing aircraft's ability to follow a route depends on the speed of the aircraft, its size, and the distance and angle between the points of the route.

31.4.6. 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.



32. Writing Plans

This chapter explains how to combine tasks, set data requests, global commands, and conditions into plans.

Writing Plans for Simulation Objects 32-2
Adding a Task or Set Data Request to a Plan 32-3
Adding a Conditional Statement to a Plan 32-4
Editing a Statement 32-10
Deleting Statements from a Plan 32-10
Printing Plans 32-10
Saving Changes to a Plan 32-11
Creating Global Plans 32-11
Controlling When Global Plans Run (Quick Launch) 32-13
Opening an Information Dialog Box for a Global Plan 32-13
Adding Global Commands to a Plan 32-14
Adding a Global Task or Set to a Plan 32-15
Sending Console Messages 32-16
Creating Objects from Within a Plan 32-17
Deleting Objects from Within a Plan 32-17
Issuing a Plan 32-18
Sending View Control Messages 32-19
Writing Plans for Units 32-22
Writing a Plan for Multiple Simulation Objects 32-22
Writing Plans for Remote Entities 32-22
Copying Plans and Plan Statements 32-23
Restarting a Plan 32-23
Abandoning a Plan 32-24

32.1. 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 30-1, 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.

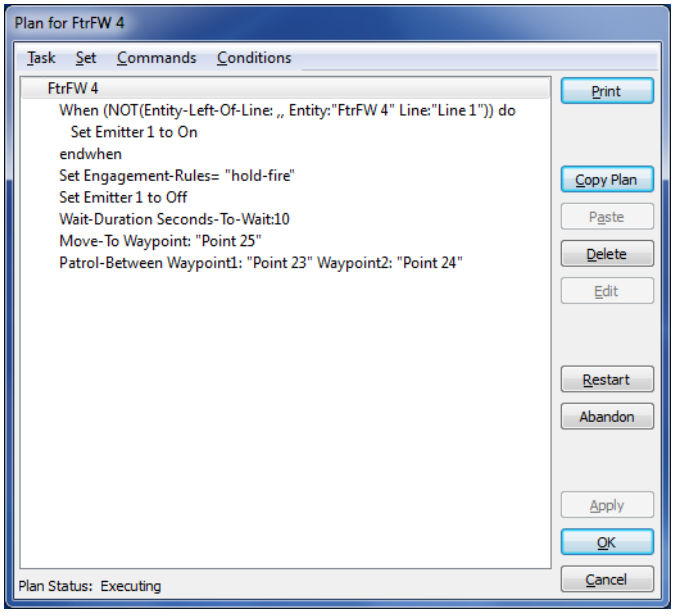


Figure 32-1. Plan window



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.

32.1.1. 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 – "*Object_name*" 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 Chapter 25, *Assigning Tasks*, Chapter 26, *Tasks for Entity-Level Scenarios*, and Chapter 27, *Tasks for Aggregate-Level Scenarios*.
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 Chapter 30, *Setting Simulation Object State*.



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.

32.1.2. Adding a Conditional Statement to a Plan

To add a conditional 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 a new statement. If the plan does not have any statements yet, the plan root – *Object_name* is already selected.
3. Choose **Conditions** → *condition*. The conditional expression dialog box opens.
4. Complete the conditional expression dialog box as described in "Building Statements in the Conditional Expression Dialog Box," on page 32-4.

! Once you make a selection from a list, you cannot change it. If you make an error building a conditional expression, you will have to Cancel out of the dialog box and start over again.

---

5. Click OK. A conditional statement block is added to the plan, with the syntax (the condition plus an "end" statement) that defines the bounds of the block of tasks that are executed if the condition is true. You need to add the task statements that you want to have executed. To add the task statements, follow the procedure in "Adding a Task or Set Data Request to a Plan," on page 32-3.

Building Statements in the Conditional Expression Dialog Box

The conditional expression dialog box is a dynamic dialog box that lets you build a conditional statement. 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 conditional statements using comparison operators, logical operators, and tests of simulation object status.

Using Prefix Notation to Establish Grouping and Precedence

To specify grouping, the Conditional Expression dialog box uses prefix notation, in which the operator comes first, followed by the operands. For example, a condition that you might verbalize as, "If entity 1 is in area A and entity 2 is in area B," is expressed in prefix notation as:

```
AND(entity 1 in area A, entity 2 in area B)
```

(For simplicity, this example does not use complete VR-Forces statement syntax.)

If you are not familiar with prefix notation, you might want to write out your conditional expressions before you begin entering them in the Conditional Expression dialog box.

Figure 32-2 illustrates how the Conditional Expression dialog box might redraw as you successively add the components of a condition statement. It creates the following conditional expression:



```
If(AND(Entity-In-Area Entity:"M1A2 2" Area:"base camp",
  resource(fuel)<25%))then
else
endif
```



The conditional expression dialog box resizes as you add conditions.

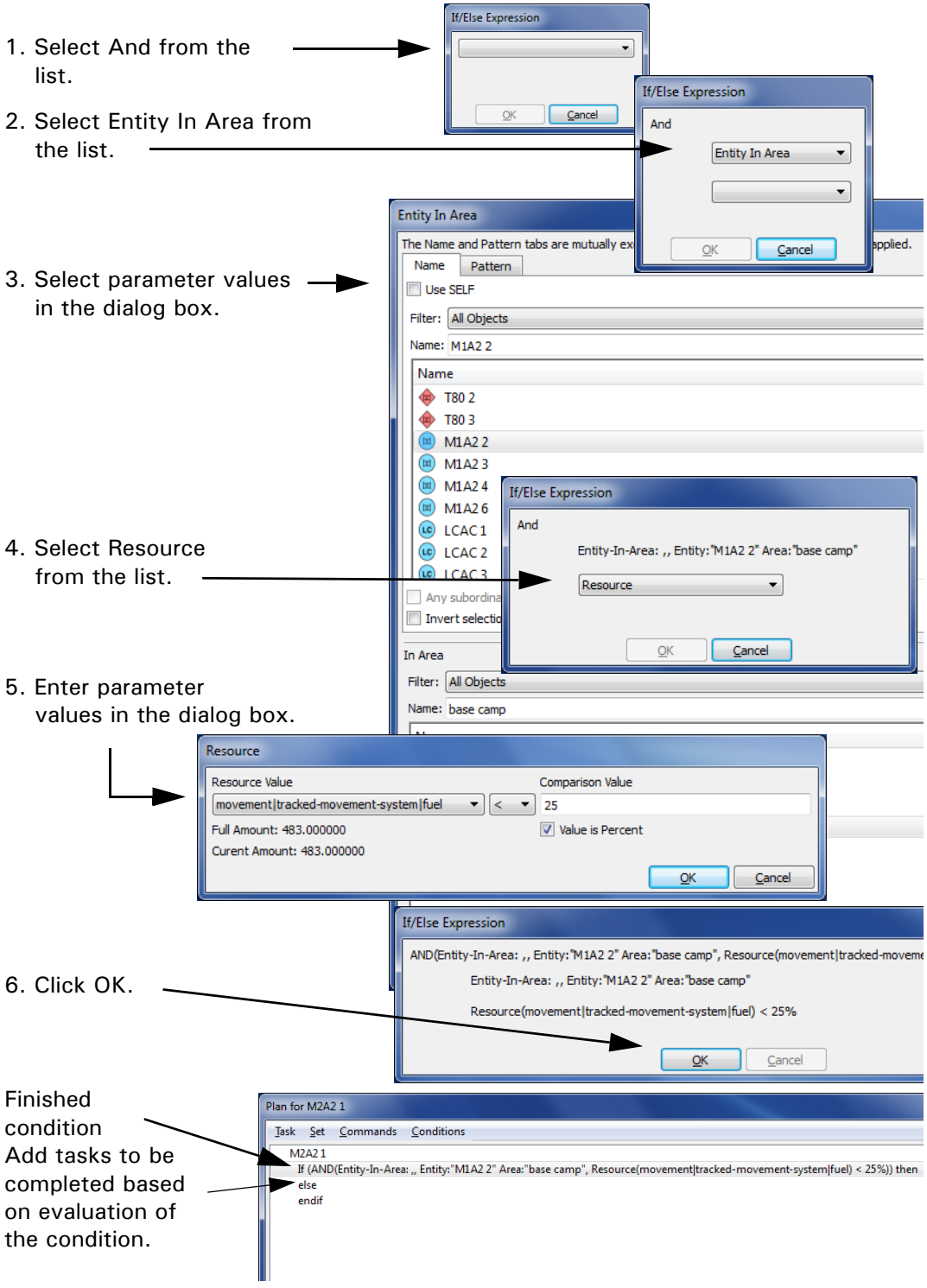


Figure 32-2. Example of building a conditional statement

Specifying Names or Patterns in Conditional Statements

When you add a conditional statement to a plan, you must specify the simulation objects for which the condition will be tested. You can specify:

- A specific named simulation object, for example, When simulation object M1A2 1 is in Area 1.
- That the condition applies to "Self", for example, When "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 Conditional Statement

To specify a named simulation object in a condition statement:

1. In a conditional statement dialog box, select the Name tab (Figure 32-3).

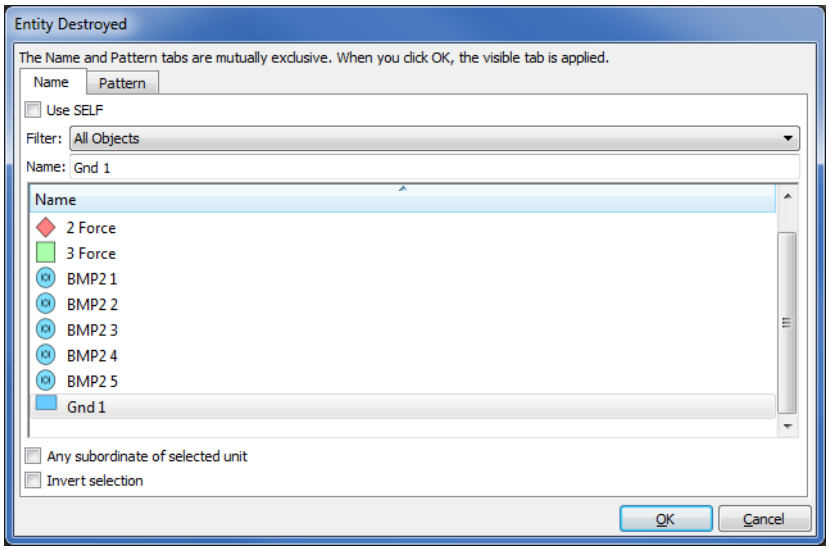


Figure 32-3. Name tab in condition dialog box

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, select the simulation object 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 Conditional 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 32-4).

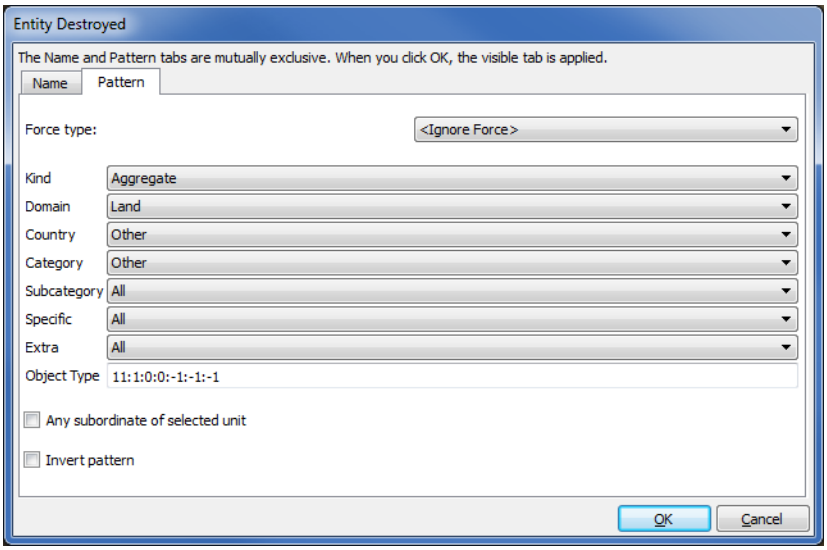


Figure 32-4. Pattern tab in condition dialog box

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.

32.1.3. Editing a Statement

You can edit simple statements in the Plan window. If you want to change a complex conditional statement, you must delete it and re-enter it.

To edit a plan statement:

1. Double-click the statement or select the statement and click Edit. 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.

32.1.4. 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 Delete 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`**.

32.1.5. 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. Click Print. A standard print dialog box for your operating system opens.
3. Complete the dialog box as you normally would.

32.1.6. 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.

32.2. 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 32-5). It lists each global plan and the simulation engine on which it is executed.

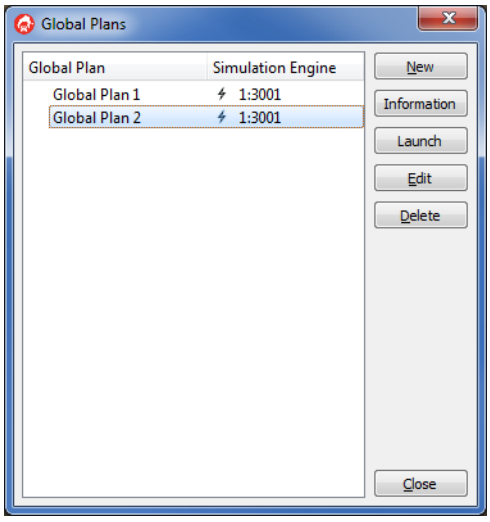


Figure 32-5. Global Plans window

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 32-6).

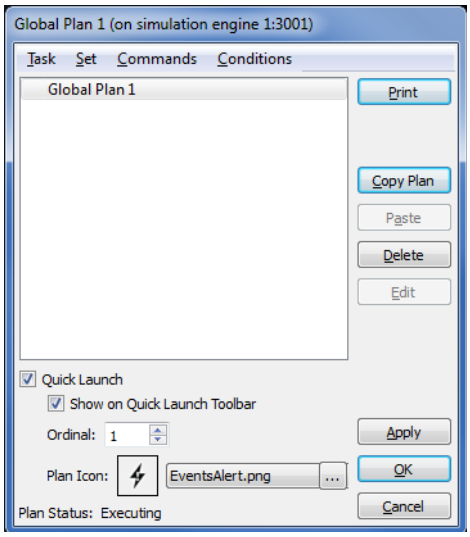


Figure 32-6. Edit Global Plan window

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 32-3 and "Adding a Conditional Statement to a Plan," on page 32-4. Add commands from the Commands menu as described in "Adding Global Commands to a Plan," on page 32-14.
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 32-13.
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.

32.2.1. 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 32-5).
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.

32.2.2. 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 32-5).
2. Select the global plan for which you want to open an Information dialog box.
3. Click Information.

32.3. Adding Global Commands to a Plan

The Commands menu submenus mimic the Task and Set menus on the VR-Forces menu bar and the options on the Simulation Objects Palette, Tactical Graphics Palette, and Hazards/Obstacles 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.

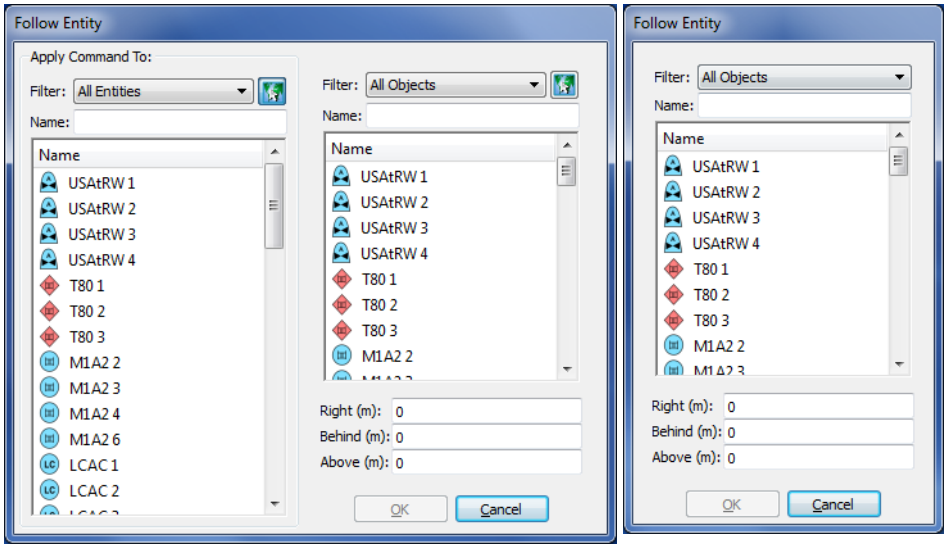
The best way to determine this is to create a simulation object of the type to which you want to give a task and see if the task or set data request is available on its context-sensitive Task or Set menu. You can also check Appendix C, *Systems and System Usage* to see if a simulation object type has the system used for a particular task.

32.3.1. Adding a Global Task or Set to a Plan

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. Global commands must be assigned explicitly to a target simulation object. Therefore, even though the list of tasks and sets on the Commands menu look the same as those in an individual plan or the main VR-Forces menu bar, the dialog boxes they use are different.

Figure 32-7 shows the Follow Entity dialog box displayed if you choose **Commands** → **Task** → **Movement** → **Follow Entity** in a global plan, and the Follow Entity dialog box displayed if you choose **Task** → **Movement** → **Follow Entity** in an individual plan. The right pane in the global command requires the same information as the local command. The difference is that in the global command dialog box you must specify the simulation object to which the command will apply. You can specify an existing simulation object or you can specify the name of a simulation object that does not yet exist, but which you expect to be created during the lifetime of the scenario, perhaps as the result of a Create command in a global plan.

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.



Global plan Individual plan

Figure 32-7. Global command task dialog box and individual plan task dialog box

To assign a global command:

1. In a Plan window or Global Plan window, choose one of the commands on the Commands menu or a submenu. The appropriate dialog box opens for the command.
2. If the dialog box requires you to specify the simulation object that will execute the command (in other words, if there is an Apply Command To panel in the dialog 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.



Remember that a global task command 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 global task 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. In the command-specific panel of the dialog box, complete the command parameters. Please see Chapter 25, *Assigning Tasks*, Chapter 26, *Tasks for Entity-Level Scenarios*, Chapter 27, *Tasks for Aggregate-Level Scenarios* and Chapter 30, *Setting Simulation Object State* for the details of assigning specific tasks and sets.

32.3.2. 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 or Global 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.

32.3.3. Creating Objects from Within a Plan

You can create any of the objects on the Simulation Objects, Tactical Graphics, and Hazards/Obstacles palettes from within a plan.

To create an object from a plan:

1. In a Plan or Global Plan window, choose **Commands** → **Create** → *object*, where *object* is an object listed on one of the submenus.
2. 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.)

32.3.4. 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 or Global 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**.

32.3.5. 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 31-2). 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 global plan or an individual plan, choose **Commands** → **Issue Plan**. The Issue Plan dialog box opens (Figure 32-8).

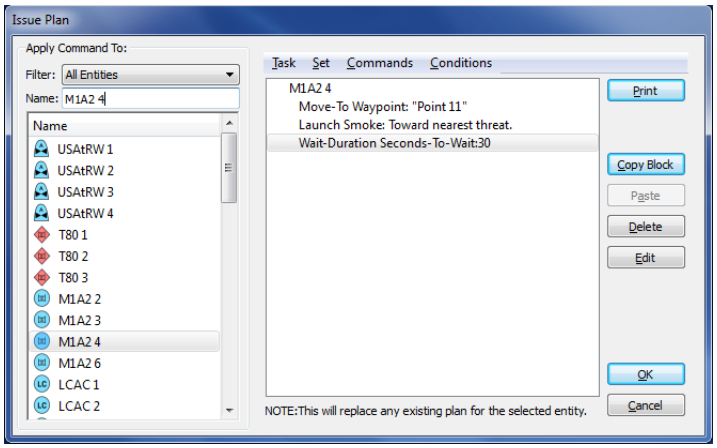


Figure 32-8. Issue Plan dialog box

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.
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.

32.3.6. 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 those in any other VR-Vantage applications in the simulation. When you send a command you can direct it to a specific observer or to all similarly named observers, for example, all Observer 1s.

The Observer View command works only with VR-Forces. It does not affect other VR-Vantage applications. If you send the Observer View command, you can specify a transition time to the new view, similar to the automated transition described in "Animating Transitions Between Observer Views," on page 44-30.



- 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.
- 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 MÄK 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 global plan or an individual plan, choose **Commands** → **Send View Control**. The Send View Control window opens.
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) (Figure 32-9).
   c. Click OK.

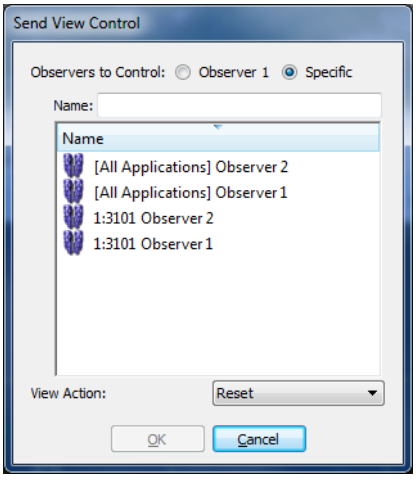


Figure 32-9. Observers to control

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. Send the selected observer view.
   - Attach mode. Send 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, select the observer view that you want to send (Figure 32-10). Optionally, specify a transition time, in seconds.

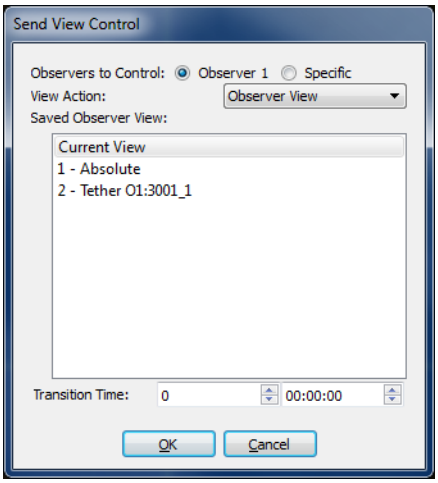


Figure 32-10. Observer view

5. If you selected absolute mode or an attach mode, complete the dialog box 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 (Figure 32-11). The default offset is 0,0,0.

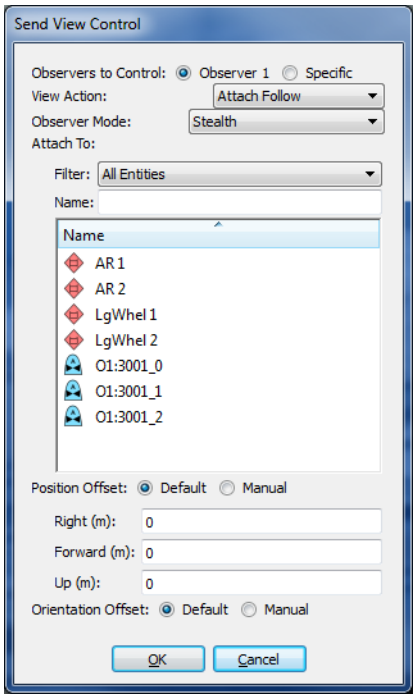


Figure 32-11. Attach mode

   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.

32.4. 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.

32.5. 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.

32.6. 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 "Configuring VR-Forces Components on Remote Entities," on page 70-18.

32.7. Copying Plans and Plan Statements

You can copy plans and individual plan statements and paste them into another simulation object's plan.

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 (*Object_name*). To copy a statement, select the statement. If you select a conditional statement, the entire statement block is selected.
3. Click Copy. If you selected the top line, it reads Copy Plan. If you select a statement, it reads Copy Block.
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 Paste.
7. Continue editing the plan.
8. Save the plan.

32.8. 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** → **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.

32.9. 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** → **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.



VII. 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 16, *Creating and Placing Objects*. For information about combat engineering objects, which are displayed similarly to tactical graphics, please see "Combat Engineering Objects," on page 22-9.



33. Introduction to Tactical Graphics

This chapter is a brief introduction to tactical graphics.

Introduction 33-2
Points 33-4
Phase Lines 33-4
Routes 33-5
Areas 33-5
Obstacles 33-5
Displaying Tactical Graphics 33-5
Showing and Hiding Tactical Graphics 33-7
Showing and Hiding Tactical Graphics Labels 33-8
Displaying Height-Above-Terrain Lines for Vertices 33-9

33.1. 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:

- 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.
  - Phase line. Infinitely long line defined by two points.
  - Route. A sequence of points in three-dimensional space.
  - Area. A polygonal area. Areas are considered to have volume, so that air and subsurface entities can interact with them.
  - 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.
  - 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.
  - Terrain Page-in Area. An area in which streaming or paged terrain is given priority for paging in.



In entity-level scenarios, waypoints, lines, routes, and areas are in the control objects category on the Tactical Graphics palette. Obstacles, linear obstacles, and pedestrian paths are on the Hazards/Obstacles palette.

In aggregate-level scenarios, waypoints, lines, routes, area, and obstacles are on the Tactical Graphics palette. Linear obstacles and pedestrian paths are not supported.

- 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.
- 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.
- 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 22-9.

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 18-3.

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 41, *Intervisibility*.
- "Terrain Profiles," on page 49-2.
- "Manually Paging-In Terrain," on page 51-6.
- "Using Disaggregation Areas," on page 23-12.
- Chapter 16, *Creating and Placing Objects*.
- Chapter 34, *Overlays*.

33.2. 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 33-1 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.

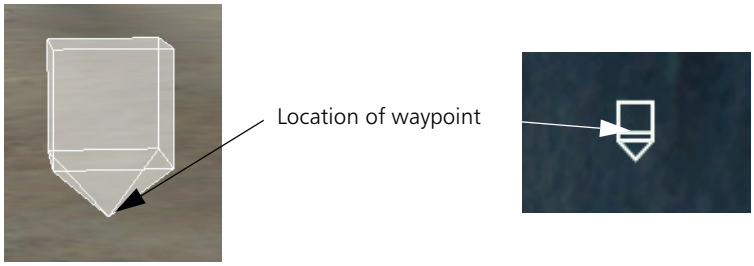


Figure 33-1. Waypoint symbol

33.3. 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 33-2 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 33-2. Phase line direction

33.4. 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.

33.5. 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?

33.6. 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.

Obstacles and linear obstacles are on the Hazards/Obstacles palette in entity-level scenarios. Obstacles (areal) are on the Tactical Graphics palette in aggregate-level scenarios. (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.

33.7. 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.

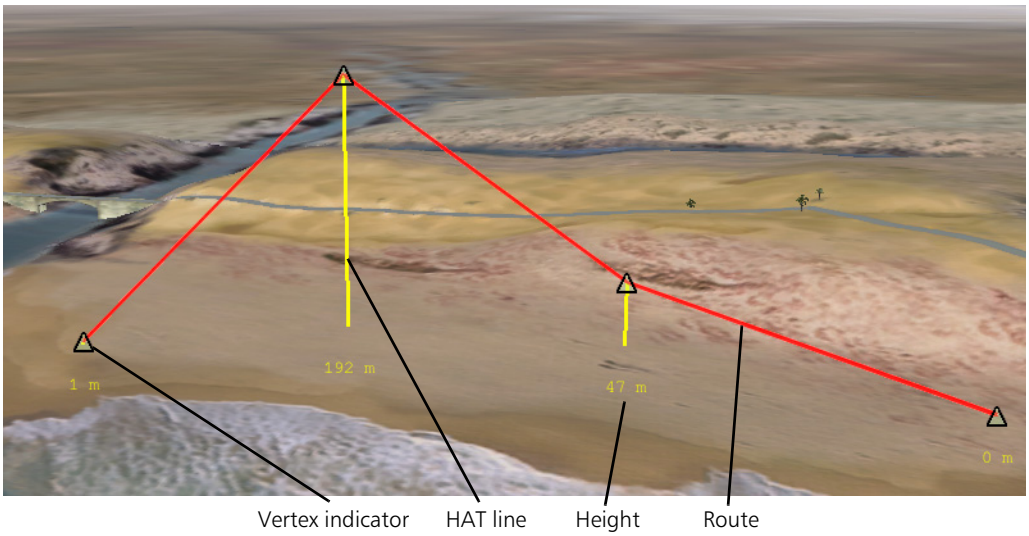


Figure 33-3. Tactical graphics with height-above-terrain lines (3D)

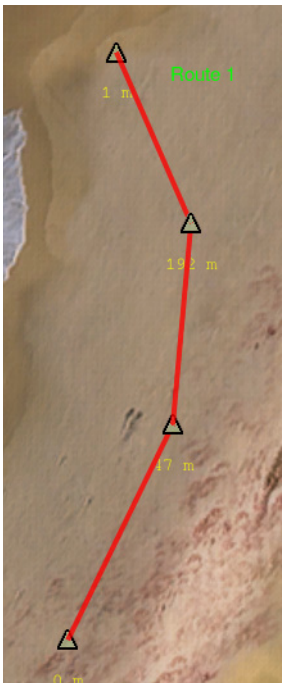


Figure 33-4. Tactical graphic (2D)

33.7.1. 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 33-5).

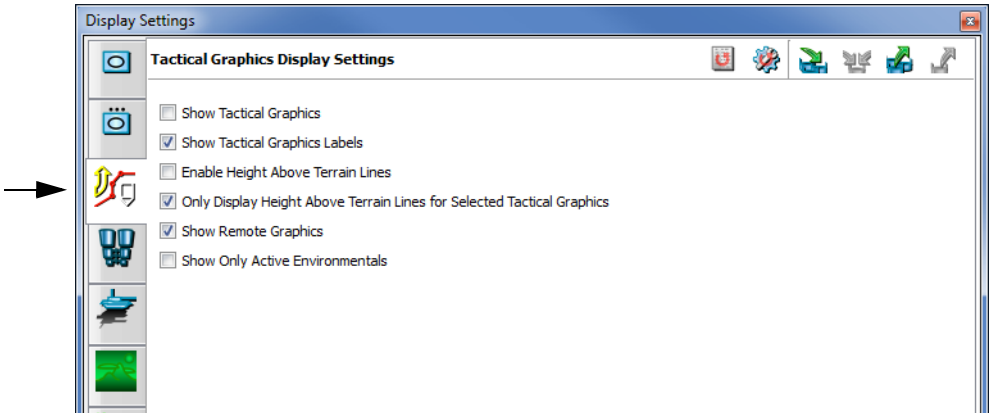


Figure 33-5. Tactical Graphics Display Settings page

3. Select or clear the Show Tactical Graphics check box.
4. To show only active tactical graphics, select Show Only Active Environmentals. For more information about active tactical graphics, please see "Active and Inactive Tactical Graphics," on page 35-18.

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 34-2).
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.

33.7.2. 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 33-3 and Figure 33-4. 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 18-15.



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 33-5).
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.

33.7.3. 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. 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) (Figure 33-3).

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 33-5).
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.



34. Overlays

This chapter explains how to create and edit overlays and tactical graphics.

Creating and Editing Overlays 34-2
- Creating an Overlay 34-3
- Selecting Overlays 34-3
- Locking an Overlay 34-4
- Hiding the Objects on an Overlay 34-4
- Changing an Overlay's Name 34-5
- Deleting an Overlay 34-5

Saving and Loading Tactical Graphics and Overlays 34-6
- Loading Tactical Graphics and Overlays 34-6

34.1. 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 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 tab 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 34-1).

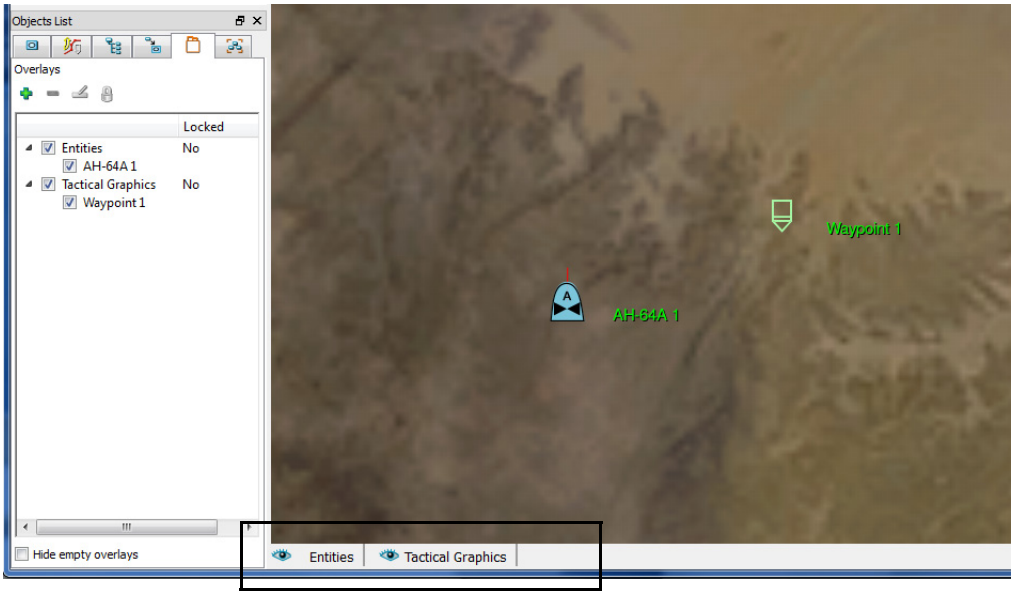


Figure 34-1. Overlays tab and Overlay Bar

34.1.1. Creating an Overlay

To create an overlay:

1. On the Objects List Panel, select the Overlays tab (Figure 34-2).

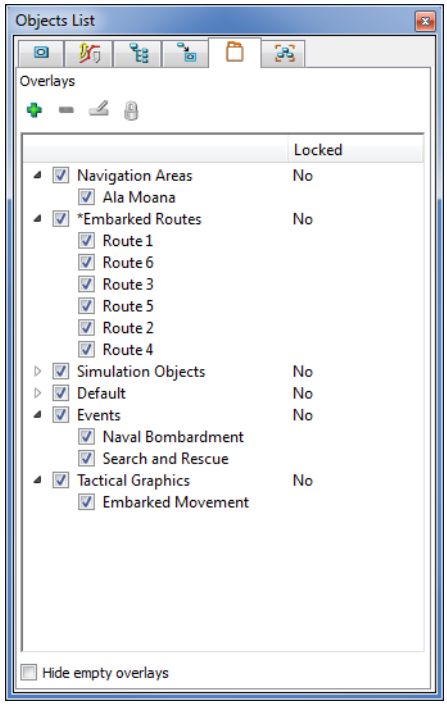


Figure 34-2. Overlays tab

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 34-5.

34.1.2. Selecting Overlays

➤ To select an overlay, click its name on the Overlays tab.

34.1.3. Locking an Overlay

When an overlay is locked, you cannot edit the objects on it. You can still add and delete objects.

To lock or unlock an overlay:

1. On the Objects List Panel, select the Overlays tab.
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.

34.1.4. 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.

To hide all of the objects on an overlay, do one of the following:

- Click the overlay label on the Overlay Bar.
- On the Objects List Panel, Overlay tab, clear the check box for the overlay that you want to hide.

➤ To hide individual objects on an overlay, on the Objects List Panel, Overlay tab, 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.

34.1.5. 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 tab.
2. Select the overlay that you want to rename and click the Rename button (), or double-click the overlay whose name you want to change. The name becomes editable.
3. Type a new name.
4. Press **`Enter`**.

34.1.6. Deleting an Overlay

To delete an overlay:

1. On the Objects List Panel, select the Overlays tab.
2. Select the overlay you want to delete.

! When you delete an overlay, all of the tactical graphics on it are also deleted.

---

3. Click the Delete button ().

34.2. Saving and Loading Tactical Graphics and Overlays

If you publish a tactical graphic, it is automatically saved to the order of battle 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** → **Save Overlay**. The Save Overlay dialog box opens.
2. Type a name for the overlay file.
3. Click Save.

34.2.1. Loading Tactical Graphics and Overlays

You can load overlays that you have saved from a previous session.

To load overlays:

1. Choose **File** → **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.



35. Creating and Editing Tactical Graphics

This chapter explains how to create and edit overlays and tactical graphics.

Creating Tactical Graphics 35-2
- Creating Freehand Lines 35-2
- Drawing Circles and Ellipses 35-3
- Drawing Boxes 35-3
- Creating Volumetric Tactical Graphics 35-4
- Adding Text to an Overlay 35-4
- Assigning Tactical Graphics to an Overlay 35-5
- Displaying a Terrain Profile When You Create a Line 35-6
- Publishing Tactical Graphics 35-7
- Creating a Computed Route 35-7
- Creating an Intel Collection Area 35-9
- Creating Routes for Aircraft 35-10

Attaching Tactical Graphics to a Simulation Object 35-11
- Attaching a Tactical Graphic Automatically 35-11

Deleting Tactical Graphics 35-11

Editing Tactical Graphics 35-12
- Editing Vertices 35-13
- Resizing Boxes and Text Objects 35-15
- Rotating an Ellipse 35-16
- Changing the Direction of a Line 35-16
- Changing the Color of a Tactical Graphic 35-16
- Changing Linear and Areal Style Properties 35-17

Active and Inactive Tactical Graphics 35-18

35.1. Creating Tactical Graphics

The generic procedures for creating tactical graphics are described in Chapter 16, *Creating and Placing Objects*. This section provides additional details about tactical graphic creation and placement.

35.1.1. 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 the Shapes category.
3. Select Freehand line or Freehand Line with Arrow.
4. Click where you want the line to start, and while holding down the left mouse button, draw a line on the screen.
5. Release the mouse button. The Create Freehand dialog box opens.
6. Edit the properties as desired.

35.1.2. Drawing Circles and Ellipses

Simulation objects do not interact with circles and ellipses. They are strictly for use as graphics.

To draw a circle:

1. Click the Tactical Graphics Palette.
2. Select the Shapes category.
3. Select Circle. The cursor changes to draw mode and shows a vertex icon.
4. 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.
5. Drag the mouse to set the radius of the circle.
6. Click to complete the circle.

To draw an ellipse:

1. Click the Tactical Graphics Palette.
2. Select the Shapes category.
3. Select Ellipse. The cursor changes to draw mode and shows a vertex icon.
4. 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.
5. 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.
6. Drag the mouse left and right and click to set the width of the ellipse.

35.1.3. 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 the Shapes category.
3. Select Box. A box is attached to the cursor
4. Click to place the box in the approximate location that you want.
5. Double-click any vertex. The vertex changes to show it is editable.
6. Drag the vertex to reshape the box.

35.1.4. Creating Volumetric 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).

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.

35.1.5. 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 the Shapes category. (Text is also in the Logistics, Intel, and Symbols categories.)
3. Select Text. The text "Text" is attached to the cursor.
4. Click to set the bottom center of the text. The Create Text dialog box opens.
5. Type the text you want in the Text box.
6. Edit any other properties as desired.

7. Click Create.

35.1.6. Assigning Tactical Graphics to an Overlay

By default, when you create a tactical graphic, it gets assigned to an overlay with the same name as the palette from which it is created. (You can change the default overlay in the 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 16-13. To move a tactical graphic to a different overlay, follow the procedure in "Moving a Tactical Graphic to a Different Overlay," on page 35-5.

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 tab of the Objects List panel.

35.1.7. Displaying a Terrain Profile When You Create a Line

You can display a terrain profile window while you create a line. You can configure terrain profiles to automatically display a terrain profile when a line is created ("Configuring Terrain Profiles," on page 49-3), 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 16-4. A Create Object tab is added to the window below the Props Palette (Figure 16-4).
2. Click the Create Object tab. A Create Object dialog box that is appropriate for the object opens (Figure 35-1).

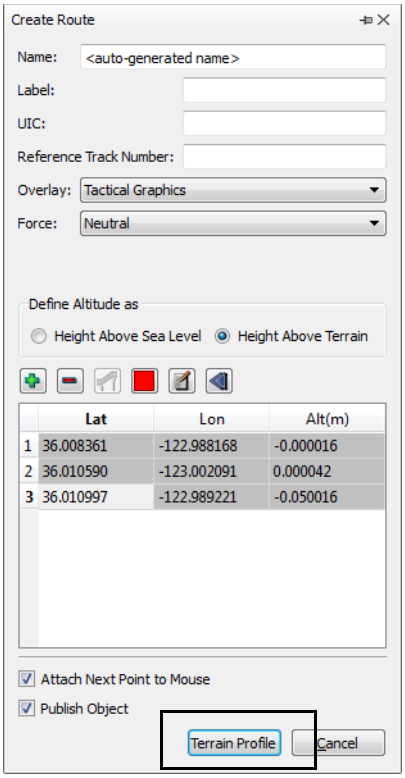


Figure 35-1. Terrain Profile button

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.

35.1.8. 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 order of battle 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.

35.1.9. 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 35-2).

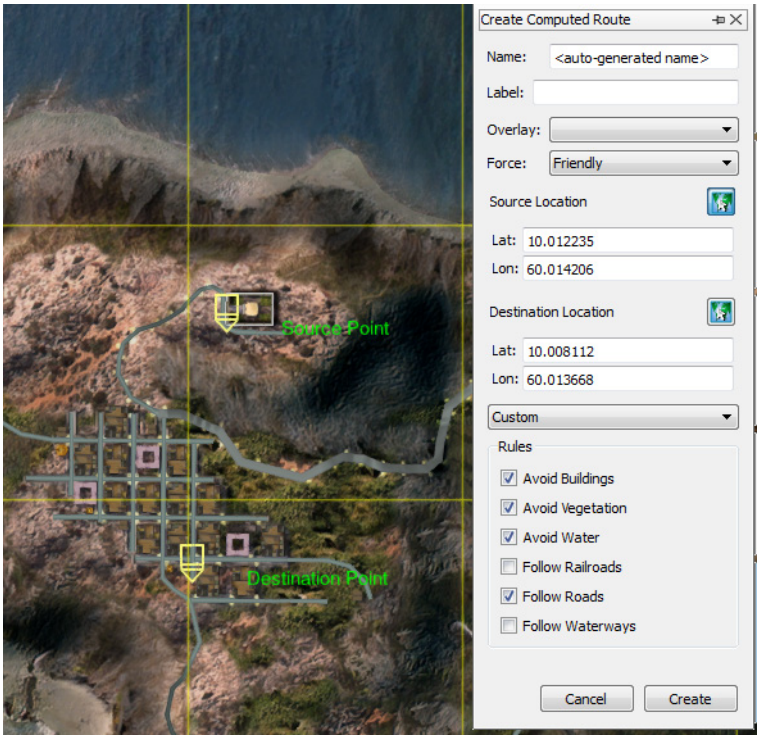


Figure 35-2. Creating a computed route

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.

35.1.10. 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 Intel category.
2. 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, click the Add button (+) next to the Area Modifiers label. The Choose Force dialog box opens (Figure 35-3).

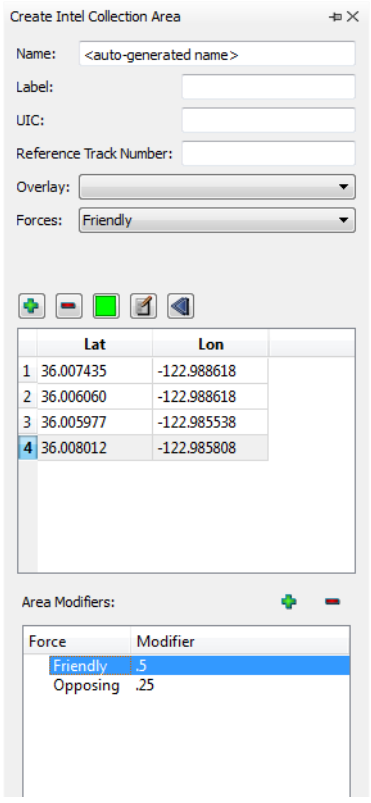


Figure 35-3. Create Intel Collection Area dialog box

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.

35.1.11. Creating Routes for Aircraft

When you create a route for a fixed-wing entity, remember that as an aircraft flies between the vertices of a route, it starts at the altitude of the first vertex and immediately rises or falls to the altitude of the next vertex as shown in the trajectory in Figure 35-4. (It does not change altitude smoothly between the vertices.) If a terrain feature in-between the two vertices is higher than the trajectory followed by the aircraft, the aircraft will fly into the terrain, even though a line between the two vertices would not pass through the terrain. There is no terrain following built into the route following process for fixed-wing entities.

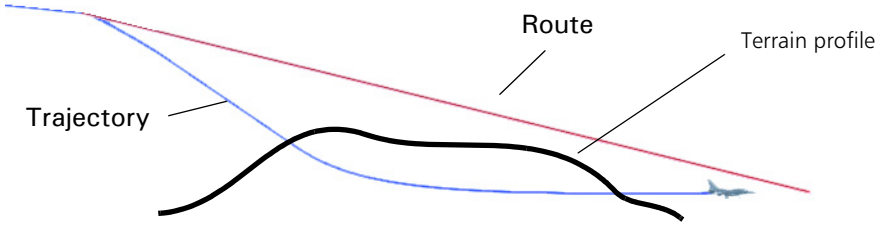


Figure 35-4. Fixed-wing trajectory between vertices

This caution applies to rotary-wing entities if there are abrupt changes in elevation in the terrain. Rotary-wing entities can follow the terrain, but they only look down, not ahead, so abrupt changes in terrain could cause a crash.

35.2. 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.

35.2.1. 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.)

To attach a tactical graphic to a 3D model when you create it:

1. Select a 3D observer mode, such as Stealth mode.
2. Select the tactical graphic that you want to create.
3. Move the cursor over the 3D model you want to attach to until you see a green box.
4. Click to place the first vertex (or point for point tactical graphics) on the 3D model.
5. Continue to create the tactical graphic as you normally would. The first point stays anchored to the model.

35.3. 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.

35.4. 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 35-13).
- Change its name.
- Change the altitude reference ("Setting Altitude in the Create Object or Edit Object Dialog Box," on page 16-15), "Specifying the Altitude for All of the Vertices in a Route," on page 16-16.
- Change the direction of a line ("Changing the Direction of a Line," on page 35-16).
- 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 35-17.)
- Move it to a different location ("Dragging an Object to a New Location," on page 16-19).
- Move it to a different overlay ("Moving a Tactical Graphic to a Different Overlay," on page 35-5).
- Change the color ("Changing the Color of a Tactical Graphic," on page 35-16).
- Resize boxes and text objects ("Resizing Boxes and Text Objects," on page 35-15).
- Rotate an ellipse ("Rotating an Ellipse," on page 35-16).



To edit a tactical graphic, its overlay must be unlocked.

35.4.1. 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 35-14.

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 35-5).
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 35-14.
7. Click Update.

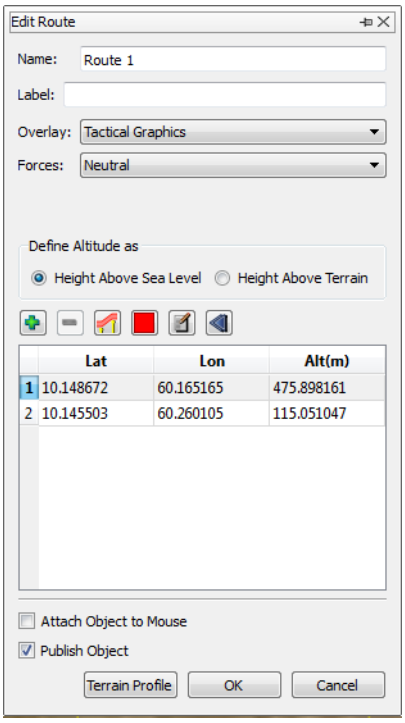


Figure 35-5. Edit Route dialog box

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.

35.4.2. 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.

35.4.3. 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.

35.4.4. 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.

35.4.5. 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. Click OK.

35.4.6. Changing Linear and Areal Style Properties

Lines and areas have a variety of properties that can be 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 or at the end of each segment of the line.

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 35-6).

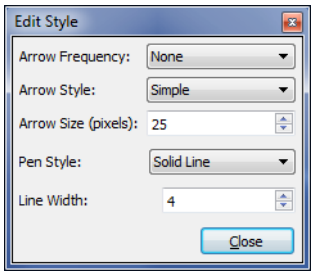


Figure 35-6. Edit Style dialog box

4. Change the properties by selecting options from the appropriate list or typing new values.
5. Click OK.

35.5. 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. A tactical graphic's state can be set 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..

Hiding 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.

For details about the Activation set data request, please see "Activation," on page 30-5 and "Tactical Graphic Active," on page 31-14.

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 35-7).

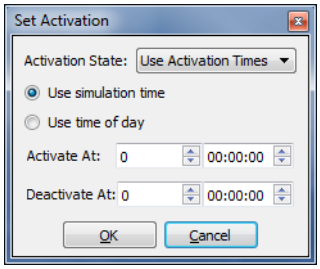


Figure 35-7. Set Activation

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 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.



36. 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 36-2
Viewing a List of Remote Graphics 36-4

36.1. Displaying Remote Graphics

VR-Forces can display graphics sent from other applications using the VR-Vantage Remote Draw API (Figure 36-1).

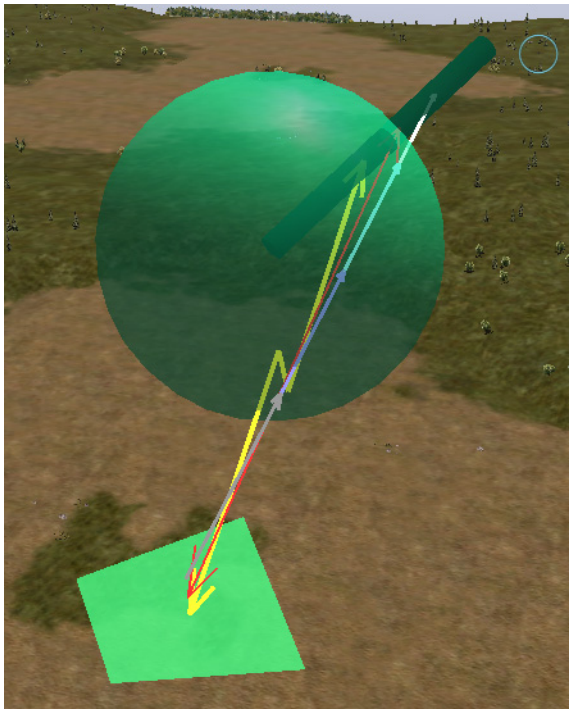


Figure 36-1. Remote graphics

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 36-2).

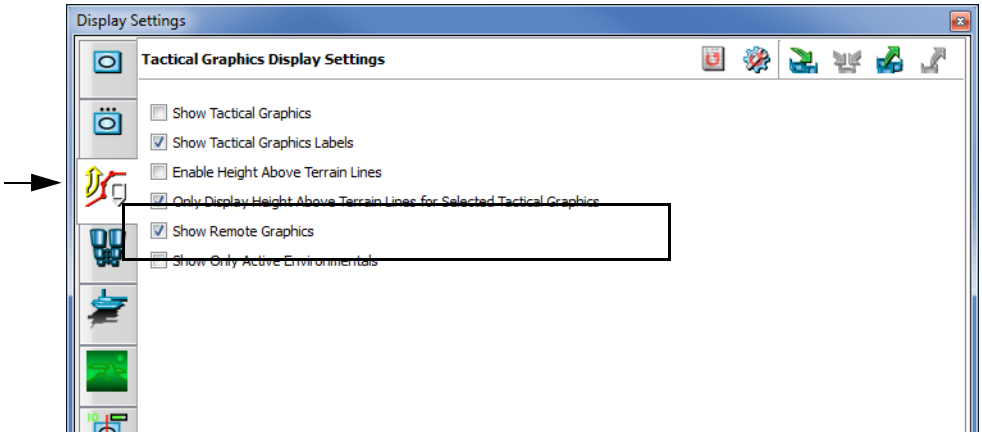


Figure 36-2. Tactical Graphics Display Settings page

3. Select or clear the Show Remote Graphics check box.



You can also enable and disable remote graphics by choosing **Settings** → **Remote Graphics**.

36.1.1. 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 36-3 illustrates the attributes for the Entity and Packet attribute group templates displayed in Figure 36-4.

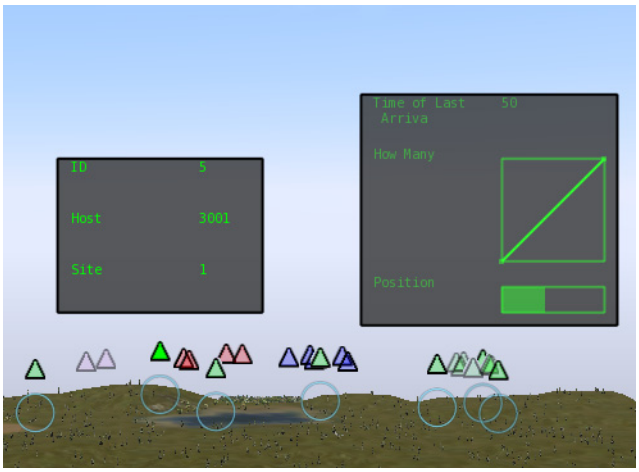


Figure 36-3. Remote graphics attribute group display

To view a list of remote graphics:

1. Choose **View** → **Remote Graphics Panel**. The Remote Graphics Panel is displayed (Figure 36-4). The Remote Graphics Sets tab lists remote graphics by set.

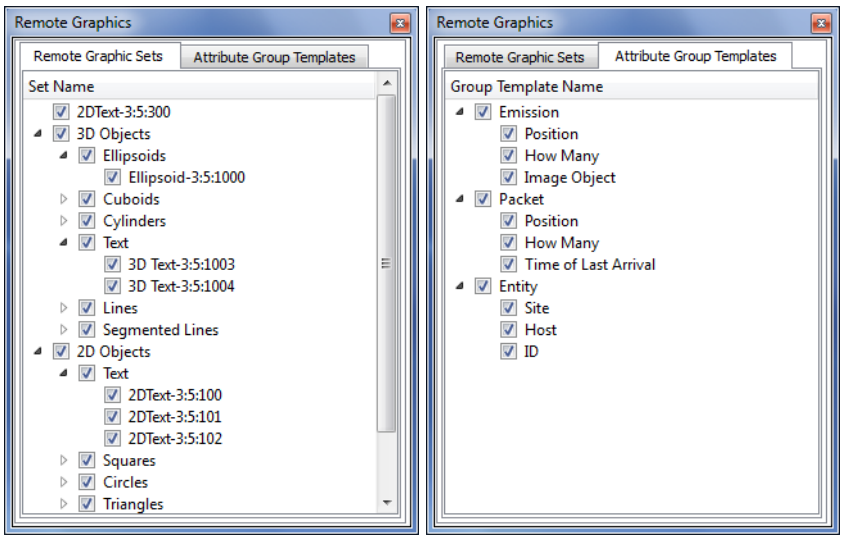


Figure 36-4. Remote Graphics Panel

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.



VIII. 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.



37. 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) 37-3
Specifying Models for Trailing Effects and Decal Effects 37-4
Displaying Track Histories 37-5
Configuring Track Histories 37-6
Displaying a Terrain Profile for a Track History 37-8
Viewing Fire and Detonation Effects 37-8
Viewing Fire and Detonation Lines 37-9
Viewing Unit Combat Graphics 37-10
Visualizing Tasks 37-12
Pinning Task Visualizations for a Simulation Object 37-13
Displaying Sensor Contact Lines 37-13
Enabling the Display of Sensor Contact Lines 37-14
Specifying which Simulation Objects Display Sensor Contact Lines 37-14
Configuring Sensor Contact Lines 37-15
Displaying Laser Designators 37-16
Displaying Supply Lines 37-17
Displaying Tactical Smoke 37-17

Displaying Electromagnetic Emissions 37-18
Displaying Radio Communication Lines 37-19
Enabling Radio Communications Graphics 37-20
Configuring the Lifetime and Display Mode 37-21
Configuring the Color of Radio Communication Lines 37-22

37.1. 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 37-1 illustrates some trailing effects.



Figure 37-1. 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 37-2).

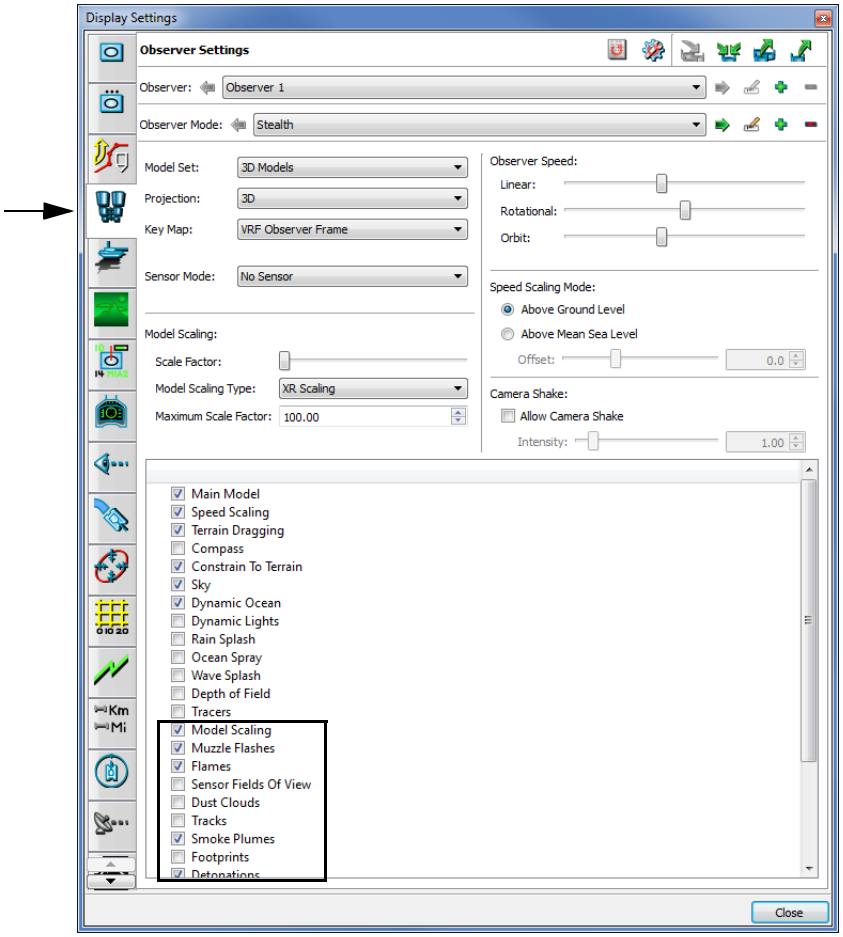


Figure 37-2. Display Settings dialog box, Observer Settings

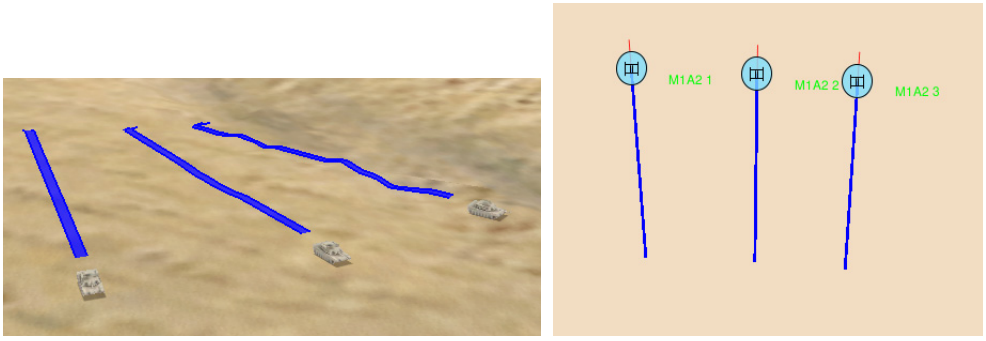
3. Select or clear the check box for each effect that you want to change.

37.1.1. 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 57-8.

37.2. 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 37-3 illustrates track histories.



3D 2D

Figure 37-3. 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.

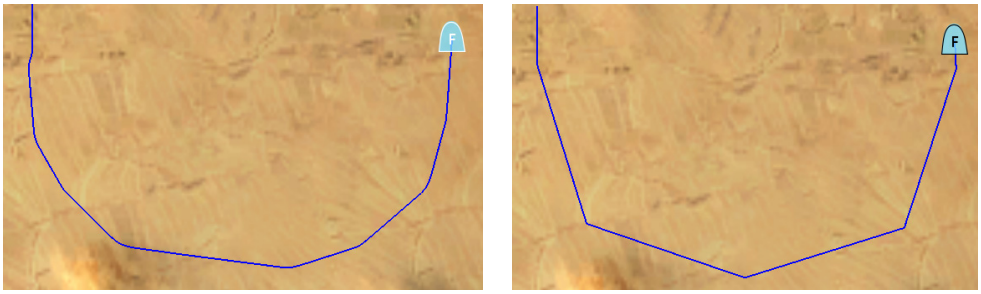
To enable or disable display of all track histories:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 37-2).
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.

37.2.1. 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 37-4 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 37-4. 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 37-5).

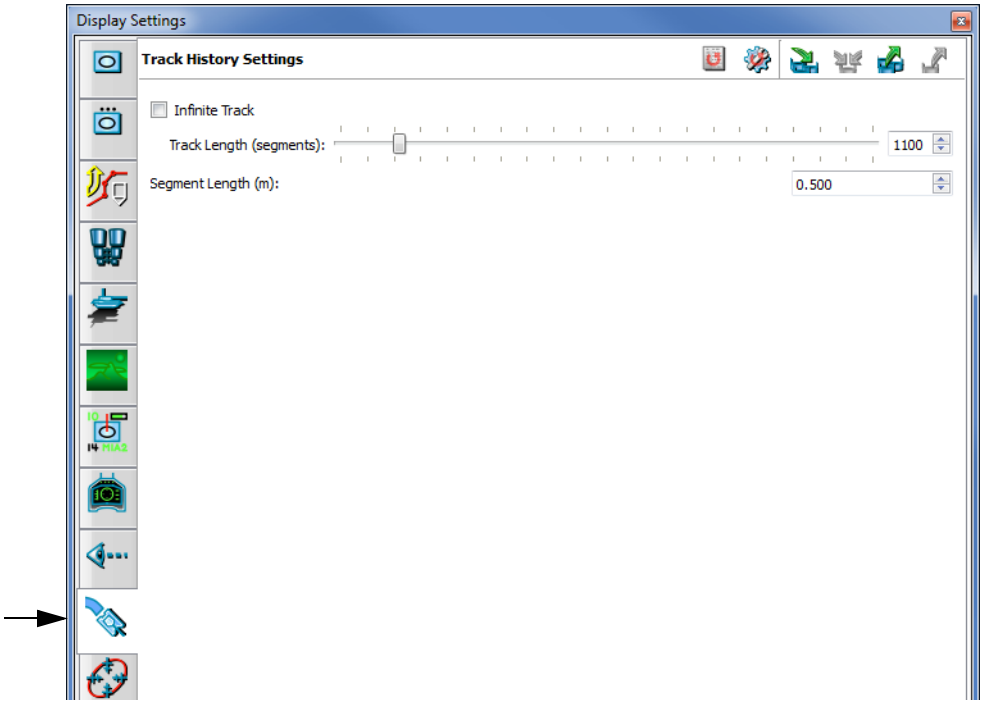


Figure 37-5. Track History Settings page

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.

37.2.2. 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 49, *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** → **Track History Profile**. The Terrain Profile dialog box opens. It updates as the simulation object moves.

37.3. Viewing Fire and Detonation Effects

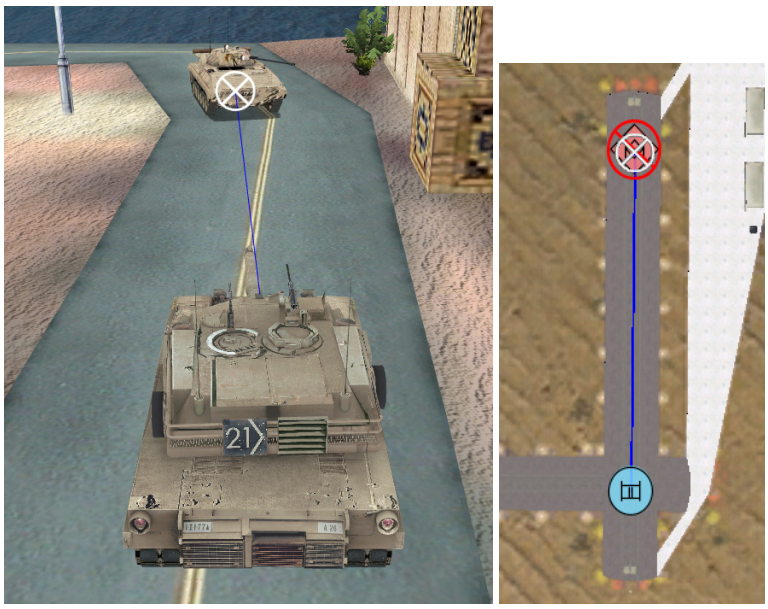
VR-Forces can display the following graphic effects when munitions are fired and detonated:

- 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 30-43.)

➤ To enable or disable display of a fire or detonation effect, choose **Observer** → *effect*, where *effect* is Flames, Smoke Plumes, Detonations, Muzzle Flashes, or Tracers.

37.4. 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 37-6).



3D 2D

Figure 37-6. 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**.

37.5. 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.

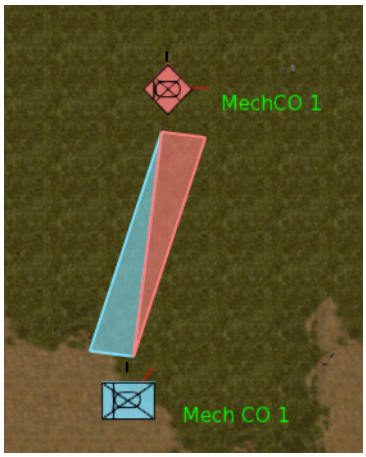


Figure 37-7. Attack indicators

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 37-8).

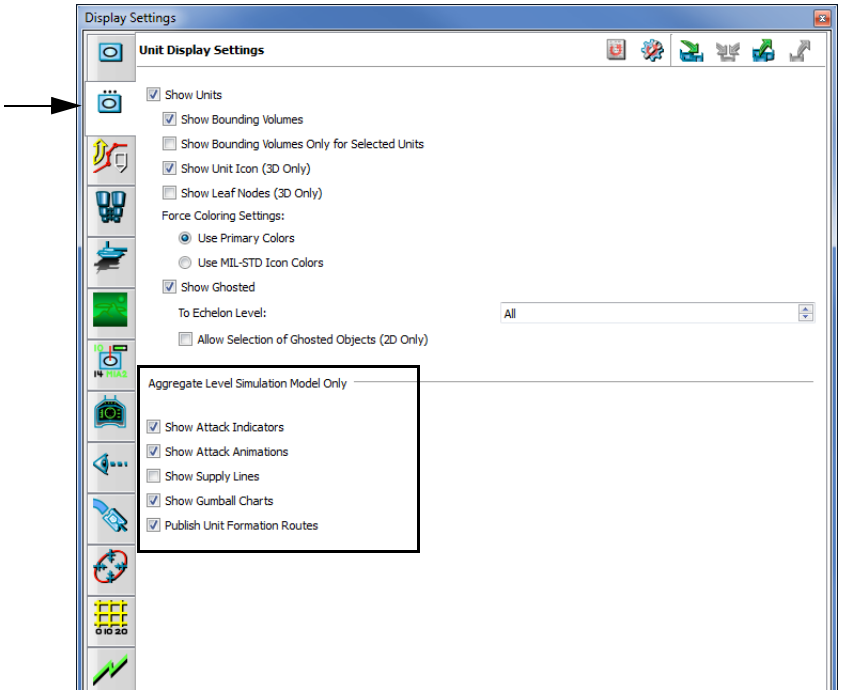


Figure 37-8. Unit Display Settings page

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.

37.6. Visualizing Tasks

VR-Forces can display graphics that help to show the task an entity is executing. Figure 37-9 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.

You can specify that task be visualized for the selected entity. You can also specify that tasks be visualized on a per entity basis. The per-entity setting is saved as part of the scenario.

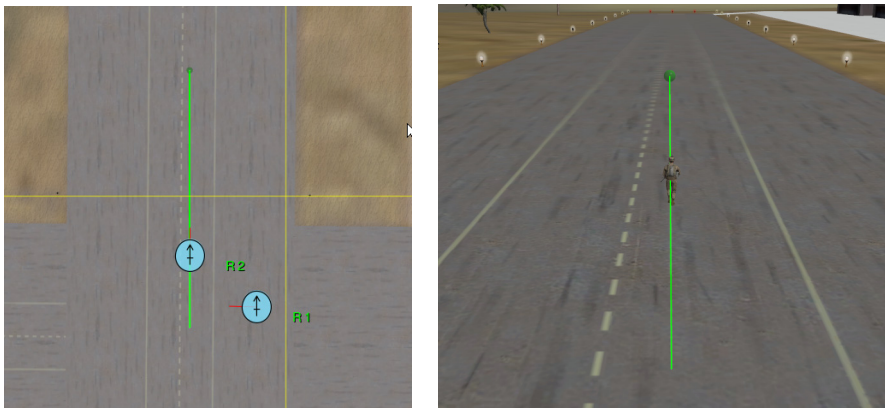


Figure 37-9. Task visualization

To specify that the selected entity show task visualization lines:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 18-10).
3. Select the Show Task Visualizations for Selected Entity check box.

- This setting is valid only if one entity is selected. If multiple entities are selected, they do not show visualization lines unless you have pinned them to the simulation object.
- You can also enable or disable sensor contact lines by choosing **Observer** → **Task Visualizations** or clicking the Task Visualizations button ( ) on the Observer Settings Toolbar.

37.6.1. Pinning Task Visualizations for a Simulation Object

By default, when task visualizations are enabled, they are only displayed for the selected simulation object. You can pin task visualization to a simulation object 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.

To pin task visualizations for a simulation object:

1. Select the simulation object for which you want to visualize tasks.
2. Choose **Objects** → **Pin Task Visualizations**.

37.7. Displaying Sensor Contact Lines

You can display lines between a simulation object and the objects it detects with its sensors (Figure 37-10). 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.

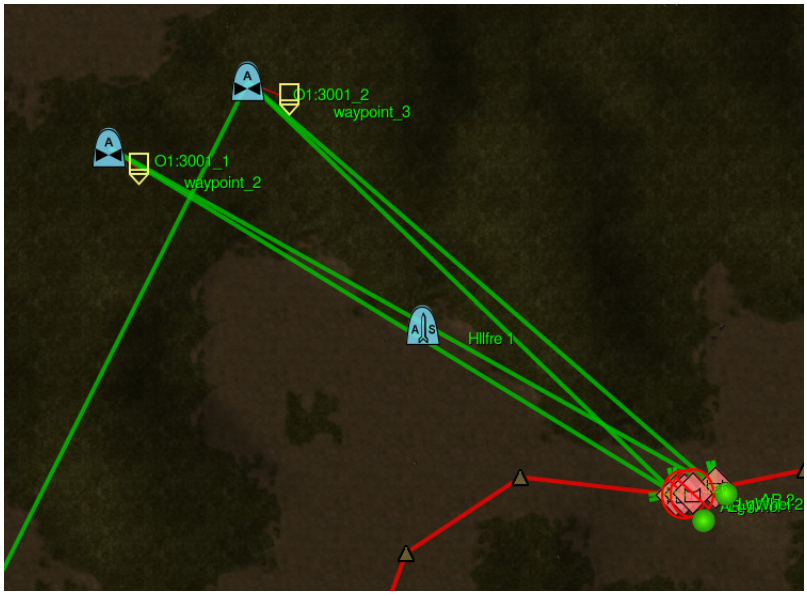


Figure 37-10. Sensor contact lines

37.7.1. 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.

37.7.2. 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** → **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 Settings page (Figure 37-11).

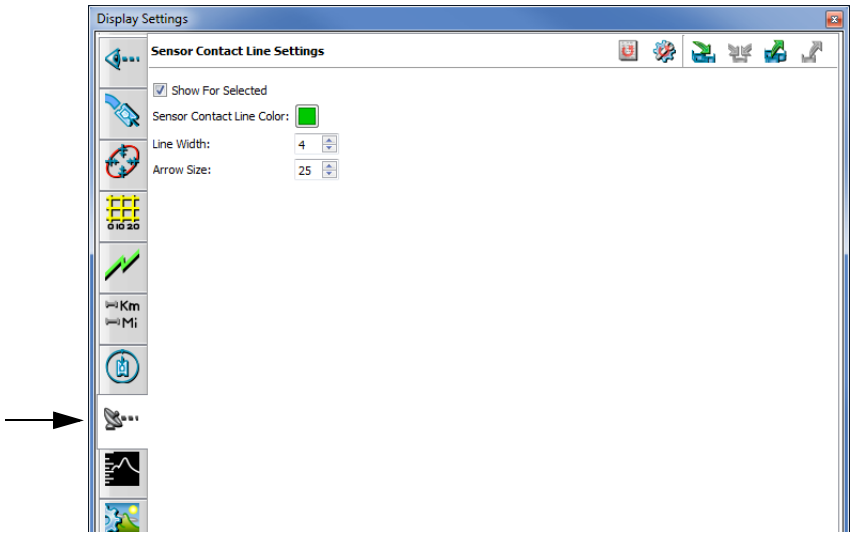


Figure 37-11. Sensor Contact Settings page

3. Select or clear Show For Selected.



This setting is saved with the GUI settings.

37.7.3. 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 37-11).
3. Change the values as desired.

37.8. 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 37-12).

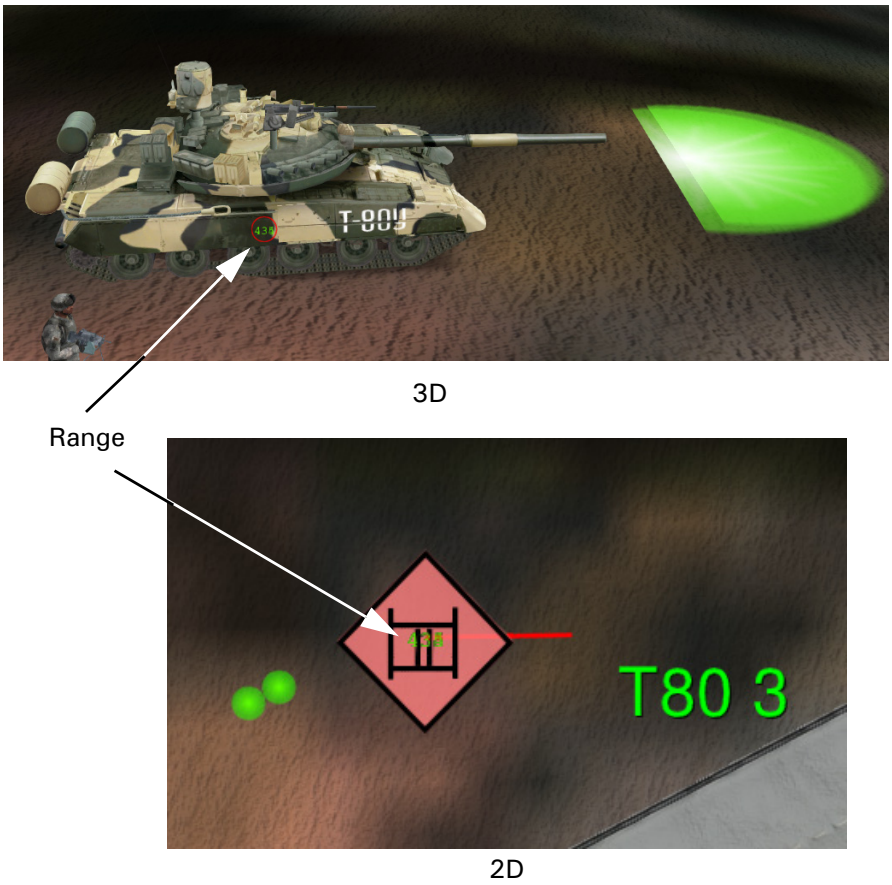


Figure 37-12. Laser designators

➤ To display laser designators, on the Display Settings Toolbar, click the Laser Designator button ( ).

37.9. 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 37-8).
3. Select or clear Show Supply Lines.

37.10. 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 or disable display of tactical smoke, choose **Observer** → **Tactical Smoke**.

37.11. Displaying Electromagnetic Emissions

VR-Forces can display electromagnetic emissions (emitter volumes) for entities that have emitter systems (Figure 37-13). 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 60-2.)

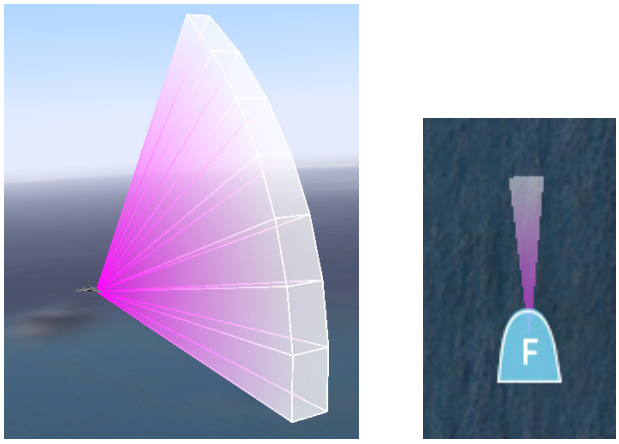


Figure 37-13. Emitter volumes

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 18-10).
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.

37.12. 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 MÄK libraries) sends a radio message to another simulation object. You can configure a communication line's color and lifetime. Figure 37-14 displays squawk indicators and a line-of-sight communication line.



Figure 37-14. 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 37-15).

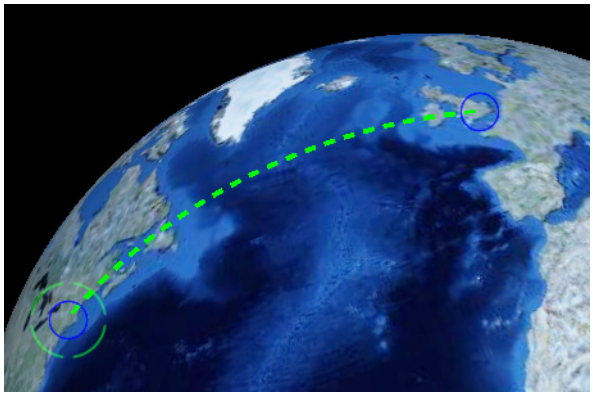


Figure 37-15. Curvature of the earth communication line

37.12.1. 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 37-16).

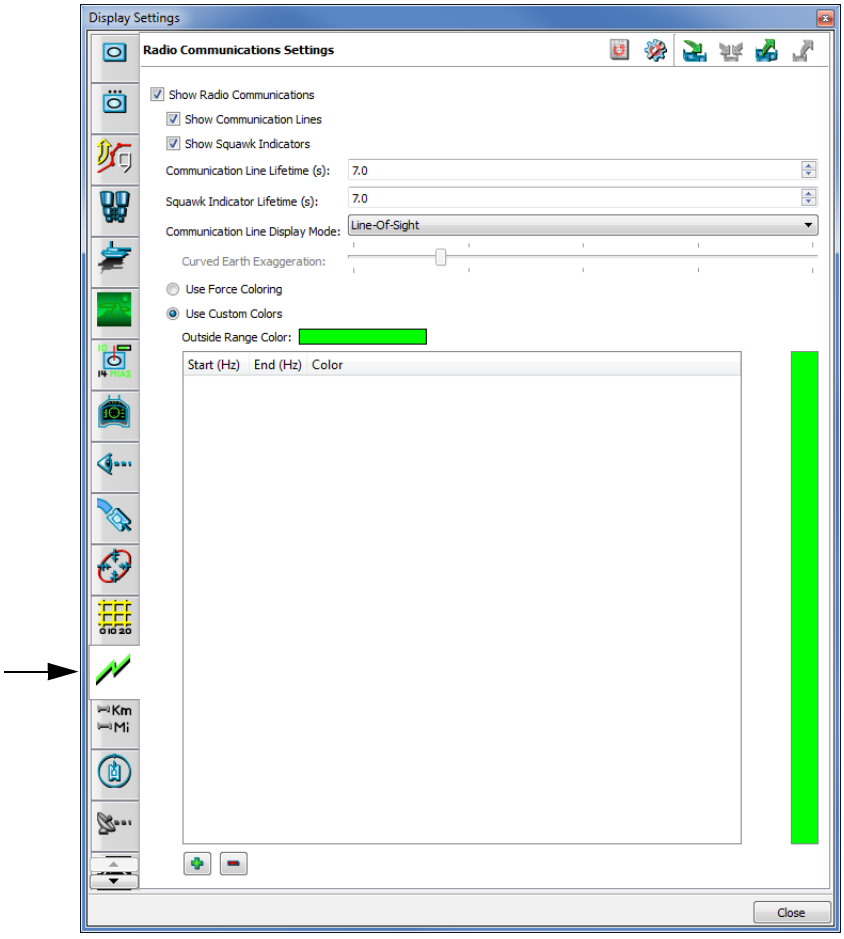


Figure 37-16. Radio Communications Settings page

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.

37.12.2. 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 37-16).
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.

37.12.3. 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 37-16).
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 37-17).

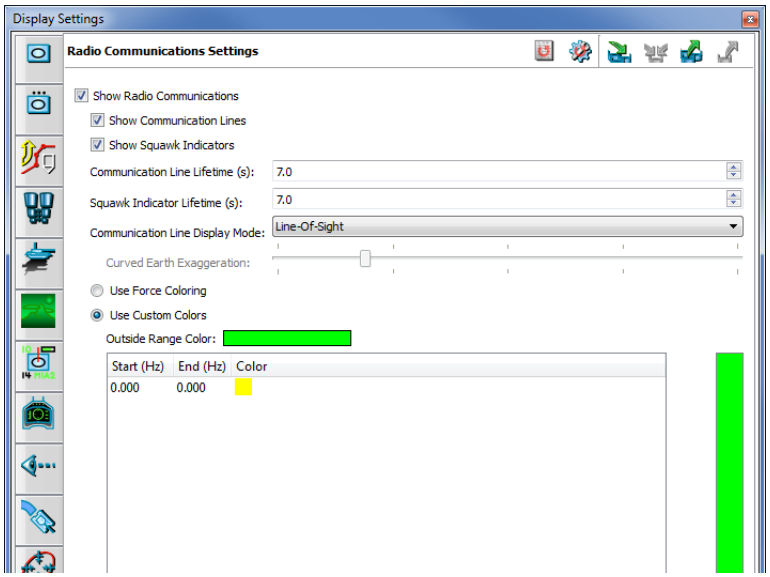


Figure 37-17. New frequency point

   c. Double-click the starting value of the range to make it editable (Figure 37-18).

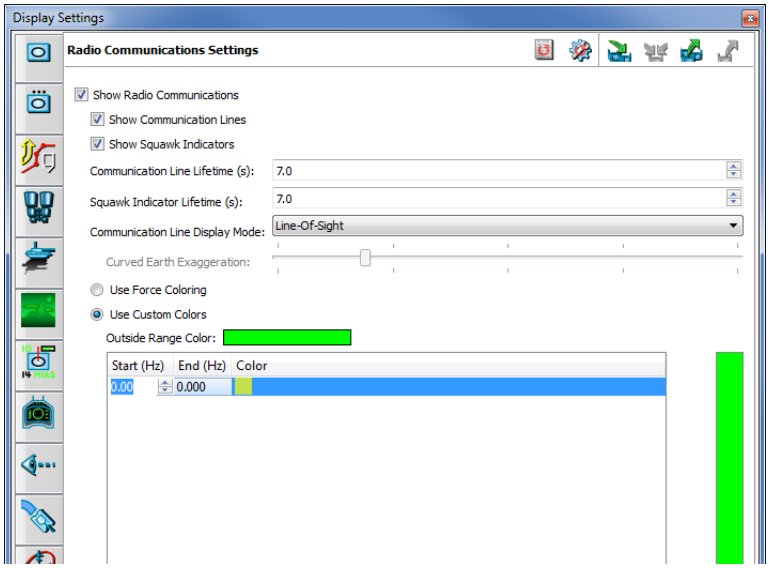


Figure 37-18. Editable frequency point

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.



38. Lighting Effects

This chapter describes the lighting effects supported by VR-Forces.

Displaying Lighting Effects 38-2
Enabling and Disabling Lighting 38-3
Configuring Ambient and Diffuse Lighting 38-4
Specifying the Emissive Lighting Scale 38-6
Displaying Dynamic Lighting 38-6
Displaying Ocean Planar Reflections 38-7
Configuring Ocean Planar Reflections 38-8
Lens Flare 38-9
Displaying Crepuscular (Sun) Rays 38-11
Configuring the Skybox Cube Map 38-12
Flashlight Mode 38-13
Disabling Lighting and Clouds 38-13
Displaying Shadows (3D Only) 38-14
Configuring Shadows 38-16

38.1. 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.

38.2. 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 38-1).

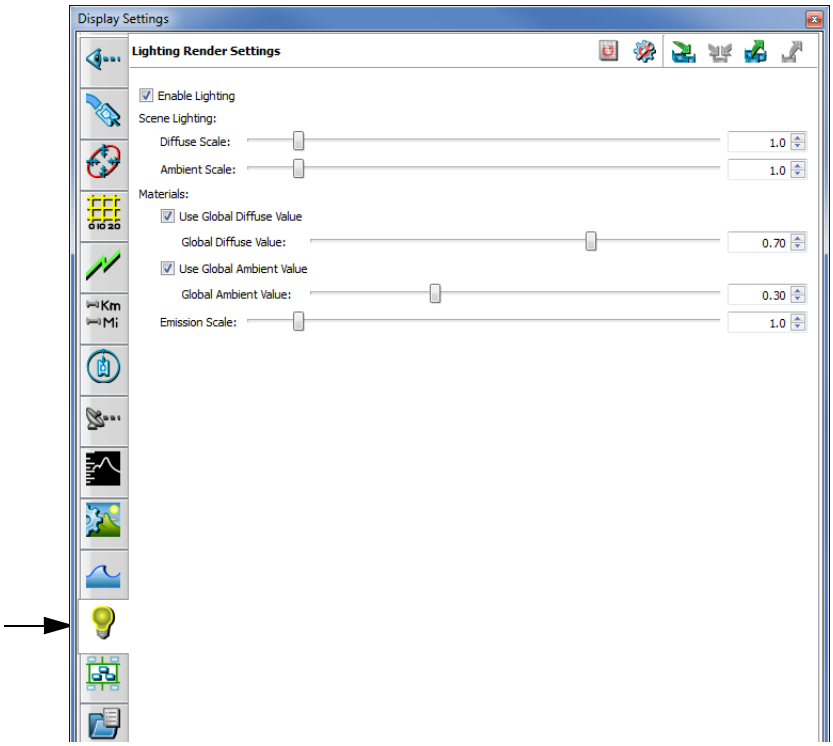


Figure 38-1. Lighting Render Settings

3. Select or clear the Enable Lighting check box.

38.2.1. 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 38-2).
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 38-2).

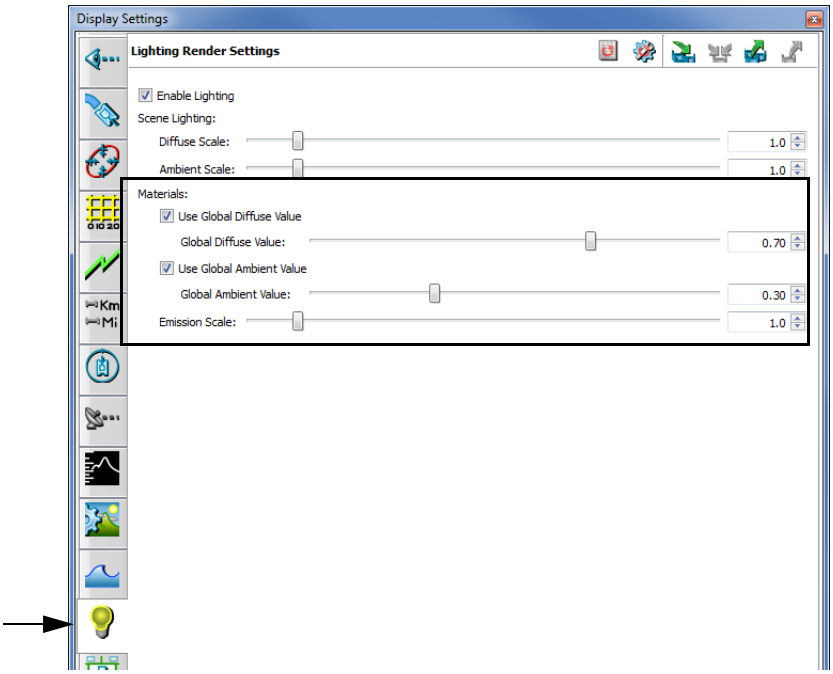


Figure 38-2. Ambient and Diffuse Value settings

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 38-2).
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.

38.2.2. 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 38-1).
3. Adjust the Emission Scale slider or change the value in the box.

38.3. 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 38-3 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 38-3. Dynamic Lighting enabled

- Light points that are part of the terrain, such as runway lights, automatically turn on at dusk and turn off at dawn.
- 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**.

38.4. Displaying Ocean Planar Reflections

VR-Forces can show the reflection of terrain and objects on water (Figure 38-4). To see planar reflections, dynamic ocean must be enabled. (For details about enabling dynamic ocean, please see "Enabling Marine Effects," on page 11-16.)

Displaying ocean planar reflections degrades performance.



Figure 38-4. Ocean planar reflection

To enable or disable planar reflections:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Render Settings page (Figure 38-5).

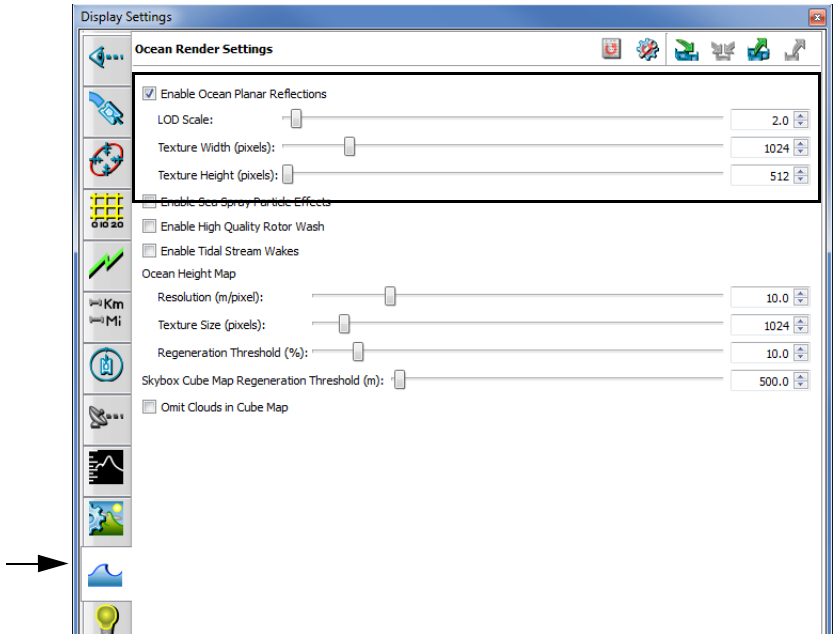


Figure 38-5. Ocean Render Settings

3. Select or clear the Enable Ocean Planar Reflections check box.

38.4.1. 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.

To configure ocean planar reflections:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Render Settings page (Figure 38-5).
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.

38.5. Lens Flare

Lens flare is a visual artifact often seen when looking at light sources such as the sun through a camera lens (Figure 38-6). VR-Forces can simulate lens flare when the observer looks towards the sun.



Figure 38-6. Lens flare

To enable or disable lens flare:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 38-7).

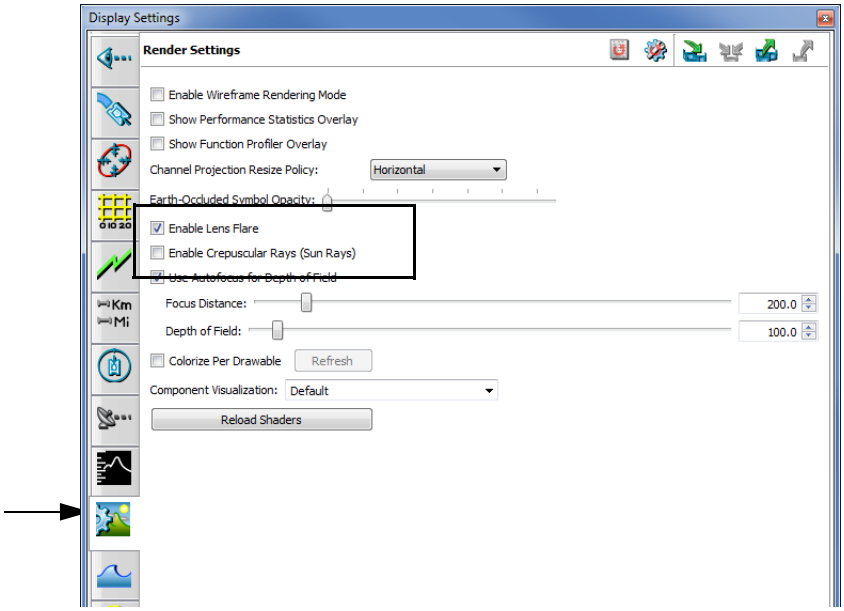


Figure 38-7. Render Settings page

3. Select or clear the Enable Lens Flare check box.

38.6. Displaying Crepuscular (Sun) Rays

VR-Forces can simulate the effect of crepuscular rays (sunbeams) beaming through clouds (Figure 38-8).

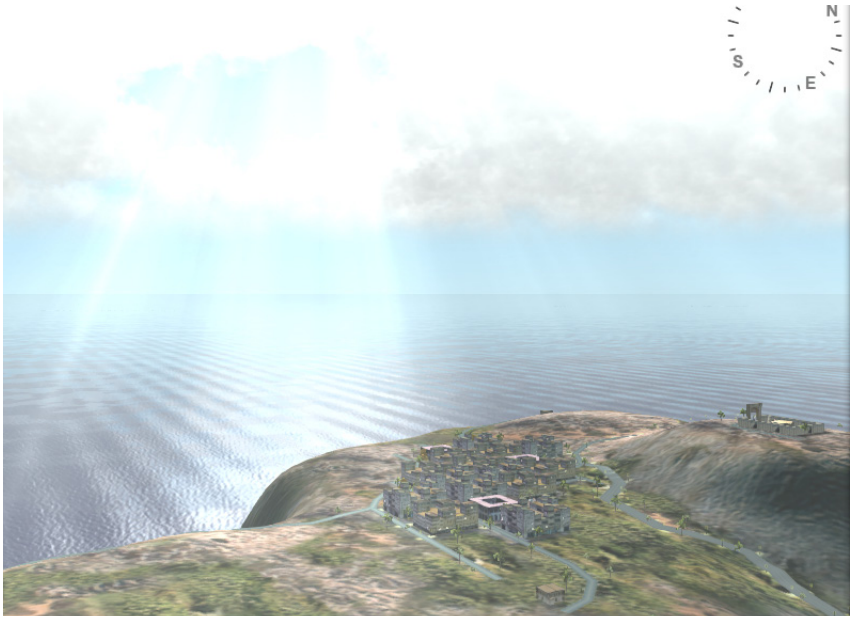


Figure 38-8. Crepuscular rays

To enable or disable crepuscular (sun) rays:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 38-7).
3. Select or clear the Enable Crepuscular Rays (Sun Rays) check box.

38.7. 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 Render Settings page (Figure 38-5).
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.

38.8. 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 38-9. 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.

38.9. 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**.

38.10. Displaying Shadows (3D Only)

VR-Forces can display shadows for entities, props, and terrain features.



Figure 38-10. 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 38-11).

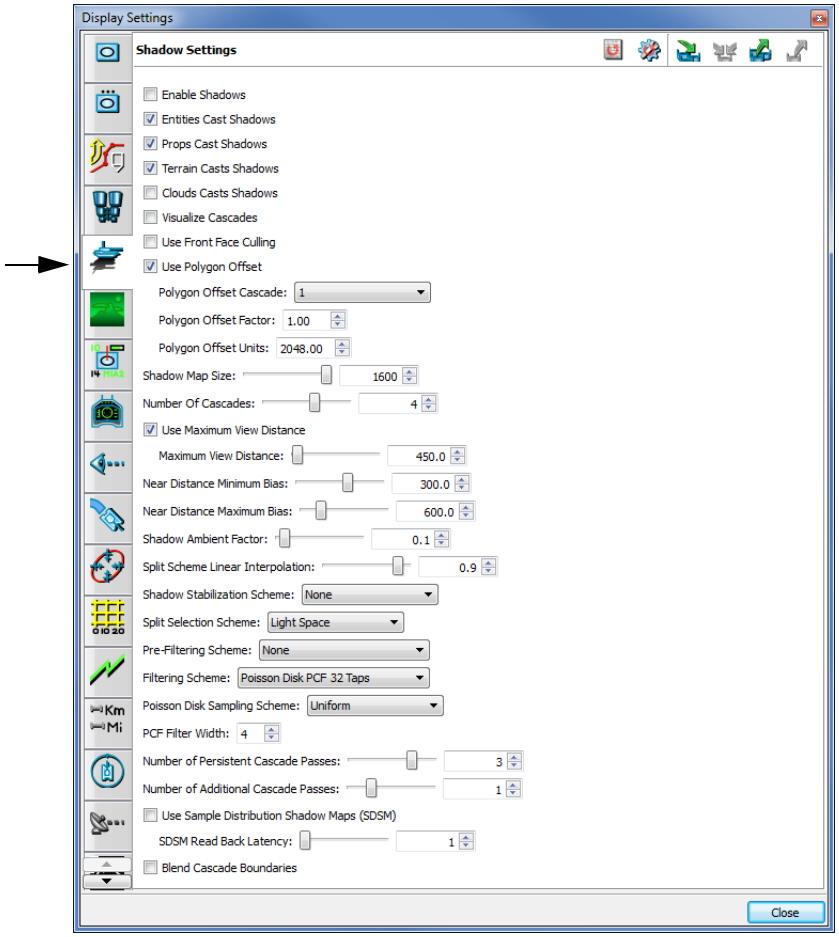


Figure 38-11. Display Settings, Shadow Settings page

3. Select or clear the Enable Shadows check box.



You can also enable and disable shadows by choosing **Settings** → **Shadows**.

38.10.1. Configuring Shadows

You can configure which objects cast shadows and parameters that affect performance and the quality of shadows. Shadow parameters include the following:

- Cascades. A scene is split into sections, with those closest to the camera showing the highest resolution shadows. You can configure the number of splits and how the quality of resolution changes as you move away from the camera.
- Polygon offset. These settings shift shadows to minimize Z-fighting.
- Shadow ambient factor. Hides Z-fighting and other visual artifacts.
- Shadow stability. Affects resolution when the camera moves.
- Filtering. These options affect visual quality.
- Sample Distribution Shadow Maps (SDSM). SDSM increases rendering quality at the expense of performance.

To configure how shadows are rendered:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Shadow Settings page (Figure 38-11).
3. To enable or disable shadows for object types, select or clear the Entities Cast Shadows, Props Cast Shadows, or Terrain Casts Shadows check boxes.
4. Change other parameters as desired.



39. Marine Effects

This chapter describes different effects for precipitation, and marine entities.

Displaying Water Effects 39-2
Enabling Sea Spray 39-4
Enabling High Quality Rotor Wash 39-4
Displaying Wakes and Spray Effects 39-5
Enabling Buoyancy for Surface Entities 39-6

39.1. Displaying Water Effects

When it is raining, VR-Forces can simulate raindrops hitting the observer's camera lens (Figure 39-1). 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 39-1. Rain effect

When dynamic ocean is enabled, VR-Forces can simulate sea spray (most visible in high seas) (Figure 39-2). It can also display rotor wash for hovering helicopters. You can display rotor wash at two levels of quality.

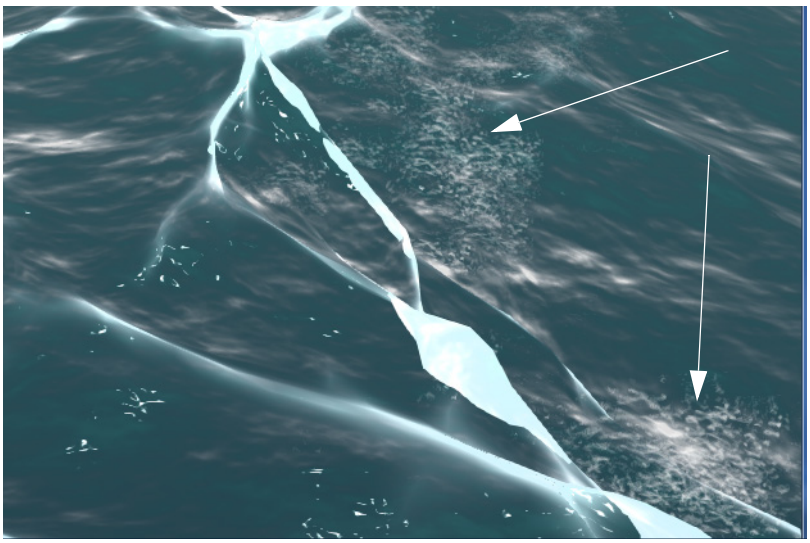


Figure 39-2. Sea spray effects

To enable or disable splash effects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 39-3).

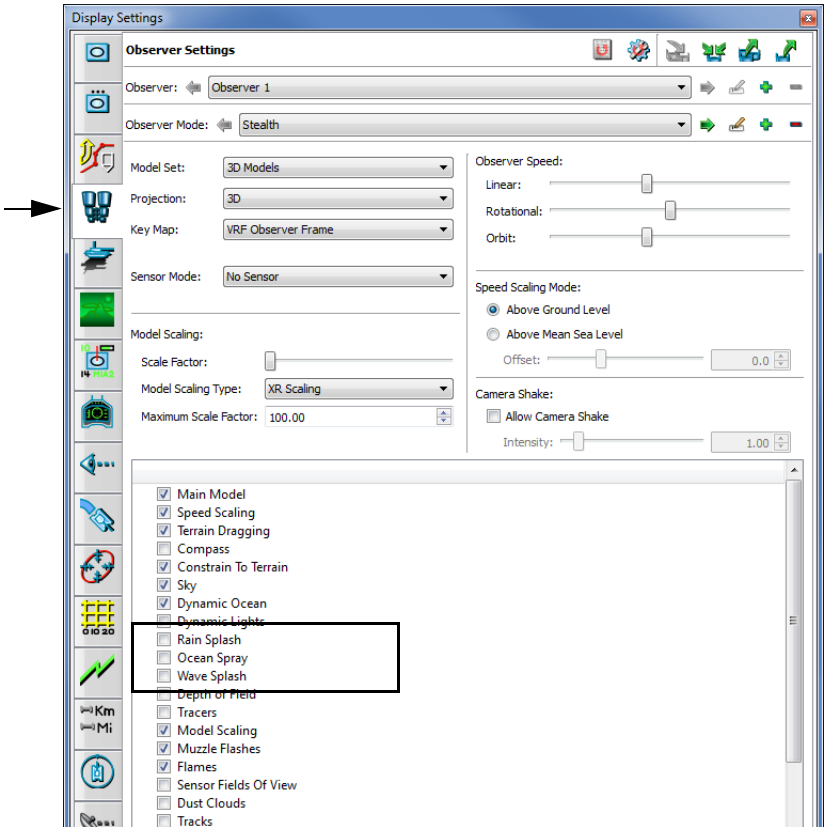


Figure 39-3. Observer Settings page

3. Select or clear the Rain Splash, Wave Splash, or Ocean Spray effect check boxes.

39.1.1. Enabling Sea Spray

To enable sea spray effects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Ocean Render Settings page (Figure 39-4).

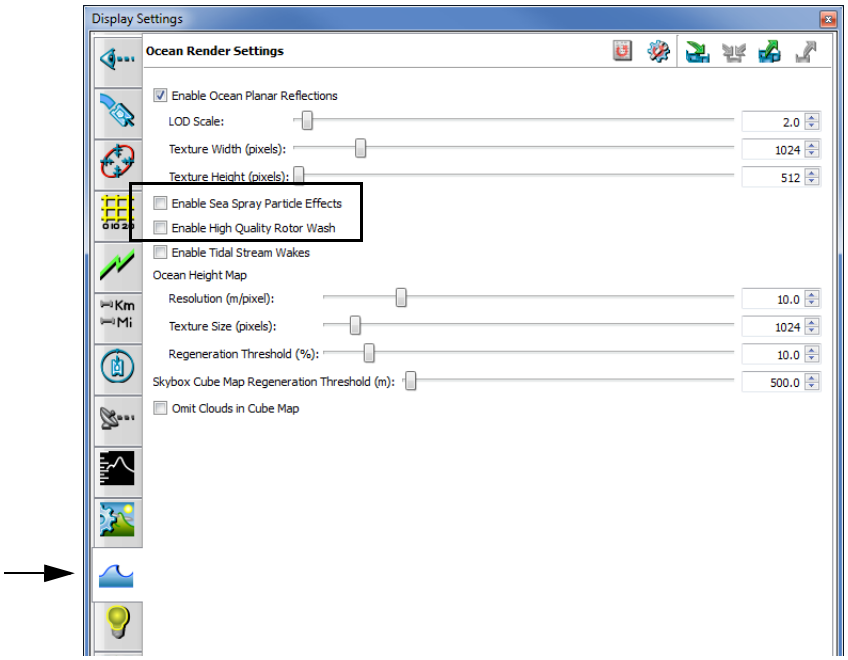


Figure 39-4. Ocean Render Settings page

3. Select Enable Sea Spray Particle Effects.

39.1.2. 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 Render Settings page (Figure 39-4).
3. Select Enable High Quality Rotor Wash.

39.2. 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 39-5).

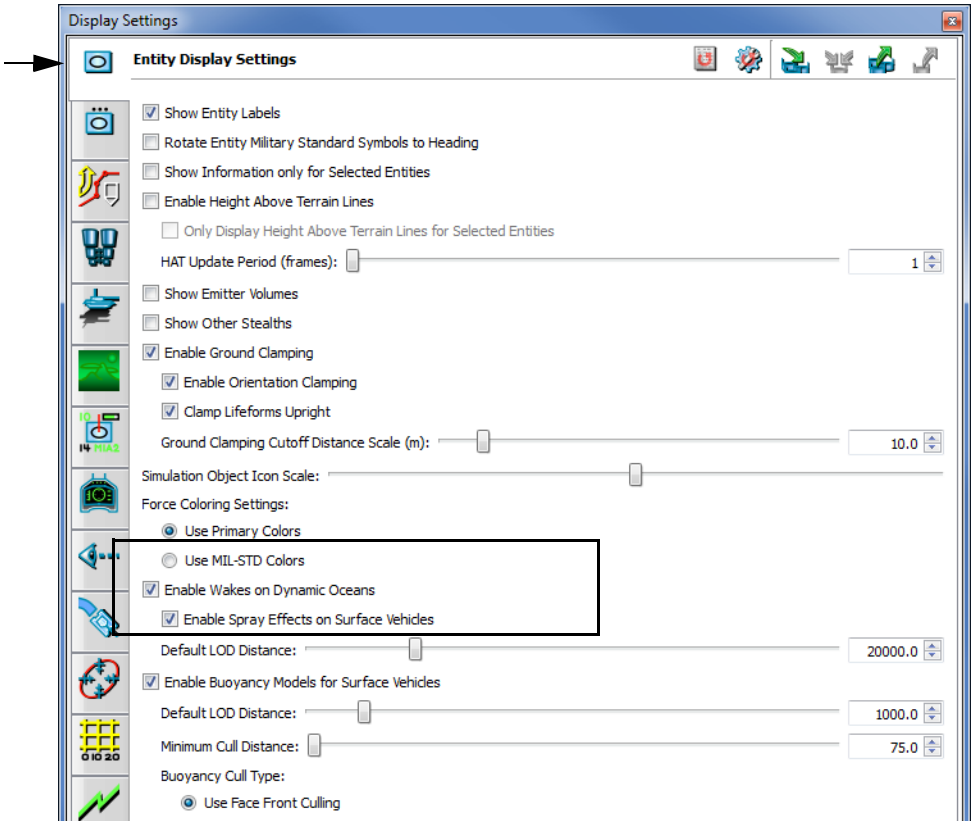


Figure 39-5. Entity Display Settings page

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.

39.3. 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., 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.

Given the buoyancy modeling constraints described in the previous paragraphs, you have two options for how VR-Forces decides how to cull processing of buoyancy models:

- 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 39-6, the shaded area is the area in which buoyancy models are calculated.

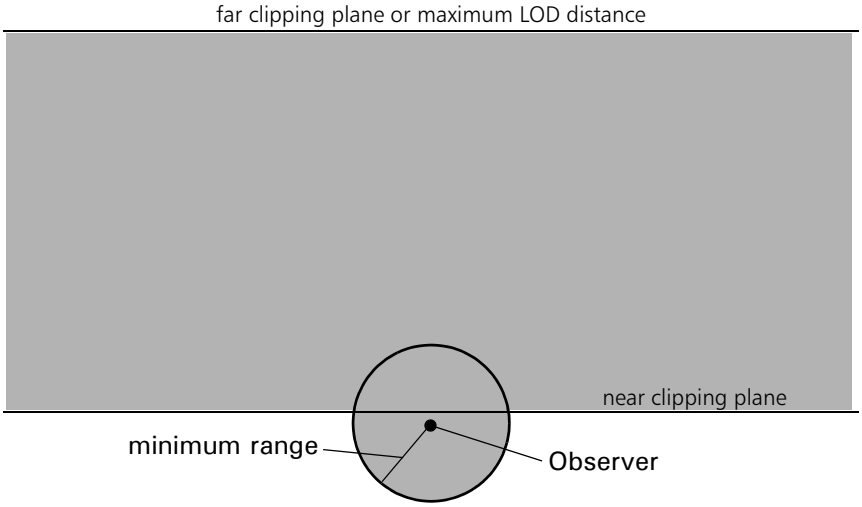


Figure 39-6. Face front culling

- View Frustum Culling. VR-Forces calculates buoyancy models for all entities within the minimum range and entities within the view frustum of any observer and no further than the far clipping plane or maximum buoyancy LOD range, whichever is closest. In Figure 39-7, the shaded area is the area in which buoyancy models are calculated.

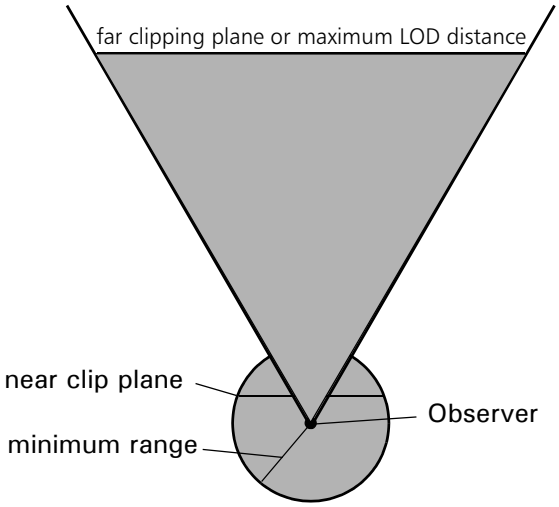


Figure 39-7. Frustum culling

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 39-5).
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. Specify the cull type, either Face Front Culling or View Frustum Culling. Please see the discussion of buoyancy for details.



40. Using Sensors

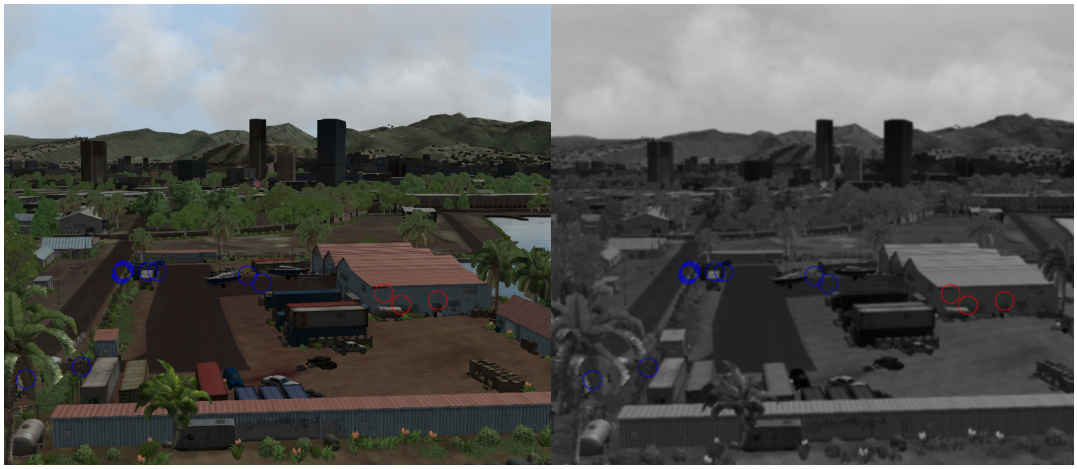
This chapter explains how to enable and configure simulated night-vision goggle (NVG) and infrared views of the scene.

Using Sensors 40-2
Enabling Sensors 40-3
Configuring Sensors 40-4
Adding New Sensors 40-5

40.1. 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. 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 (daylight) IR sensor

Figure 40-1. CameraFX



The sensors described in this chapter are strictly visual effects. They have no relationship to the sensor systems used by simulation objects.

40.2. Enabling Sensors

Sensors are observer-specific (Figure 40-2). VR-Forces includes observer modes that have sensors enabled. The names of the observer modes match the sensor modes that they enable.

- NVG (CameraFX).
- IR (CameraFX).
- IR BlackHot (CameraFX)
- Camera (CameraFX). This mode uses the Visual Camera sensor mode.

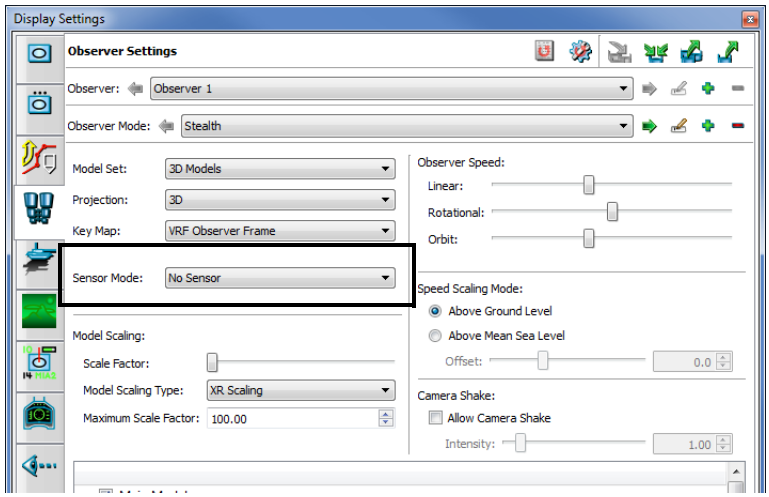


Figure 40-2. Sensor setting for observer mode

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 43-3. For details about editing channel attributes, please see "Changing a Channel's Attributes," on page 73-8.

40.3. 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)

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. (This probably sounds obvious, but the Sensor Settings page does not automatically display the configuration for the sensor that is being used. If you want to edit the sensor that you are currently using, you have to be sure that you have selected it to edit.)

To configure sensors:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Sensor Settings page (Figure 40-3).

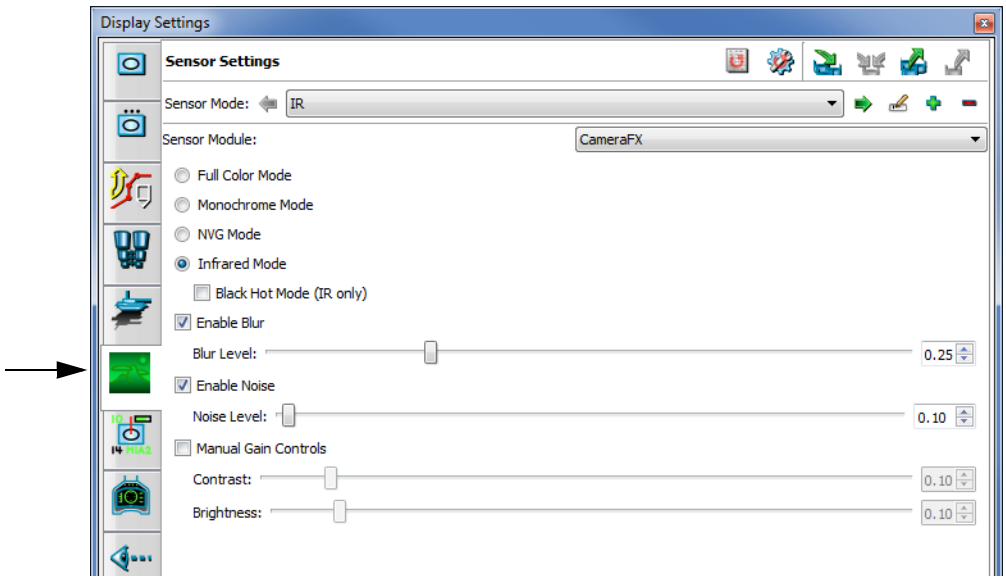


Figure 40-3. Sensor Settings page

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. Select the check boxes for the effects that you want to be enabled for this sensor configuration.
7. Adjust the values for the enabled effects.

40.4. 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 40-3).
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.



41. Intervisibility

This chapter explains how to use the intervisibility feature.

Intervisibility (Line-of-Sight) 41-2
Types of Intervisibility 41-3
Point-to-Point Intervisibility 41-3
Intervisibility Fan 41-4
Entity Intervisibility 41-4
Intervisibility Objects can be Transient or Permanent 41-5
Creating a Transient Intervisibility Object 41-6
Creating a Permanent Intervisibility Object 41-7
Editing an Intervisibility Object 41-8
Configuring Intervisibility Lines and Fans 41-9
Pinning an Intervisibility Object to a Simulation Object 41-10
Showing and Hiding Intervisibility Lines 41-10
Displaying Intervisibility Line Information 41-11
Displaying an Intervisibility Line's Terrain Profile 41-12
Deleting Intervisibility Objects 41-13

41.1. 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.

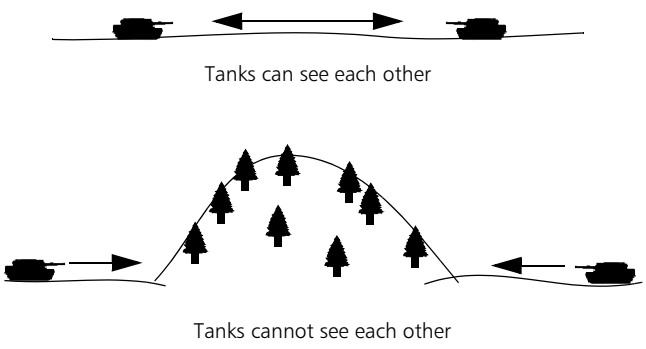


Figure 41-1. Intervisibility

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.

41.2. 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.

41.2.1. Point-to-Point Intervisibility

Point-to-point intervisibility (Figure 41-2) 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.

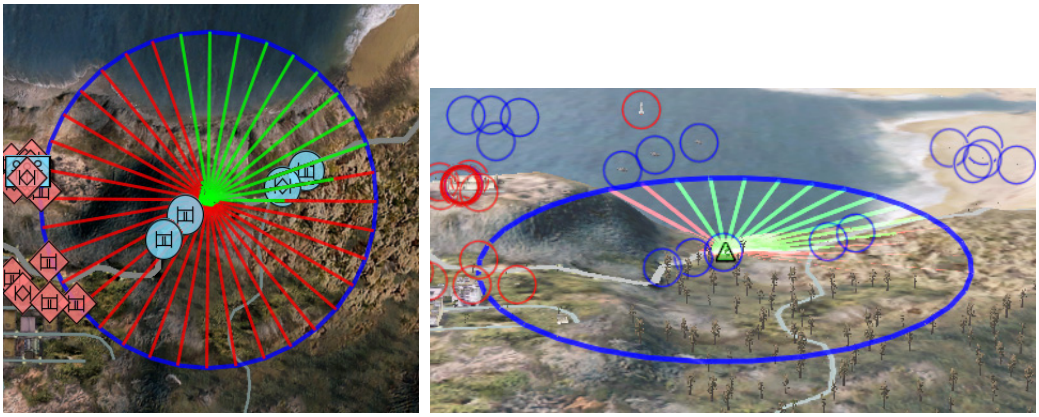


2D 3D

Figure 41-2. Point-to-point intervisibility

41.2.2. Intervisibility Fan

An intervisibility fan (Figure 41-3) 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 41-9.)



2D 3D

Figure 41-3. Intervisibility fan

41.2.3. Entity Intervisibility

Entity intervisibility (Figure 41-4) 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 41-9.)



2D 3D

Figure 41-4. Entity intervisibility

41.3. 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 41-9.) 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.

41.3.1. 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 41-5).

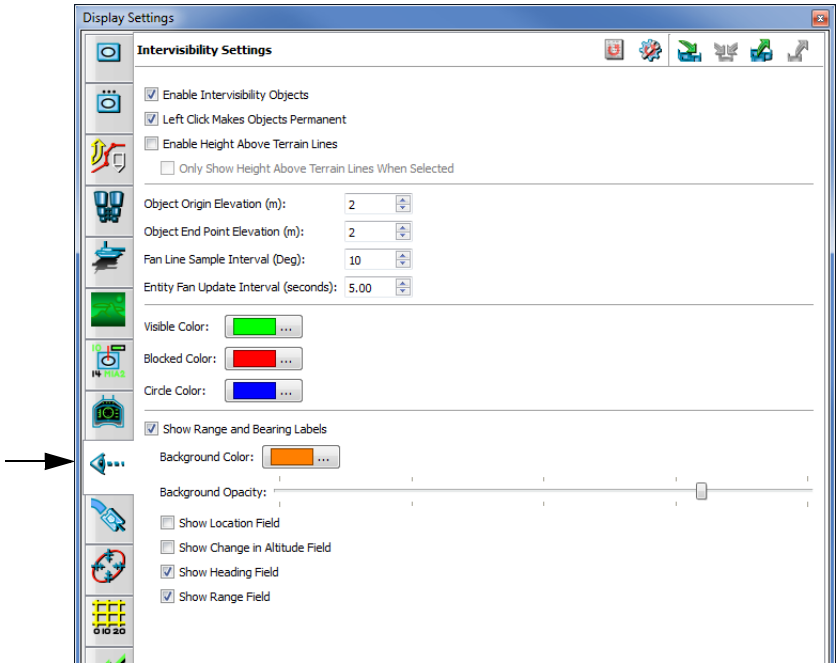


Figure 41-5. Intervisibility Settings page

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.

41.3.2. 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 41-5).
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.

41.4. Editing an Intervisibility Object

You can change the following aspects of an intervisibility object:

- Change the location of either end point of an intervisibility line, to change its length and direction.
- Change the origin of an intervisibility fan, thereby moving the entire fan.
- Change 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.

41.5. 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 41-11.

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.

To configure intervisibility objects:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 41-5).
3. Change the settings as desired. Table 41-1 describes the settings.

Table 41-1: 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.
Circle Color	The color to use for the outer circle of an intervisibility fan.
Show Range and Bearing Labels	Enable or disable display of range and bearing labels.

Table 41-1: Intervisibility settings

Option	Description
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.

41.6. 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.

41.7. 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 41-5).
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.

41.8. 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 41-6. 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 41-5).
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 41-1.
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.

41.9. 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 41-7 shows an obstructed intervisibility line and its terrain profile, whose graph illustrates the line intersecting the terrain.

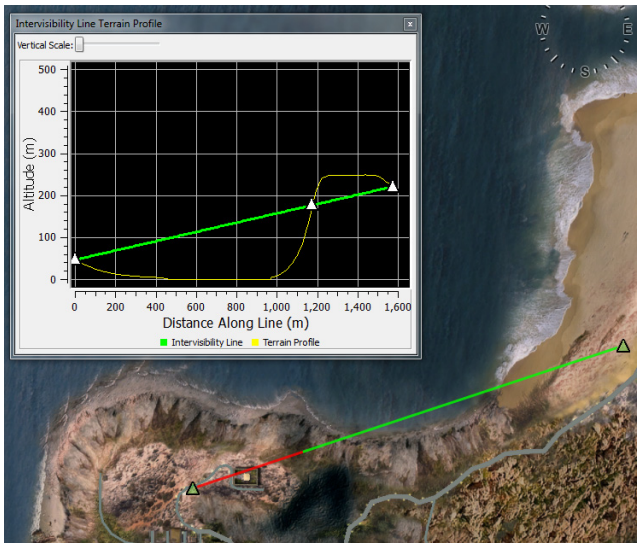


Figure 41-7. Terrain profile for an intervisibility line

You can disable this setting on the Display Settings dialog box, Terrain Profile Settings page. For details, please see "Configuring Terrain Profiles," on page 49-3.

41.10. 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**.



42. Sound Effects

VR-Forces can play sound effects for entities, fire events, and detonation events.

Introduction to Sound Effects 42-2
Enabling Sound Effects 42-2
Setting the Sound Volume 42-3
Specifying the Range in which Sounds are Heard 42-3
Associating Sound Files with Entities and Events 42-4
Mapping a Sound File to an Entity 42-5
Mapping Sound Files to Fire and Detonation Events 42-8

42.1. 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 that sounds will be heard only when the observer is within a certain distance of the entity or event that generates them.

Sound effects are not supported on Linux.

42.2. Enabling Sound Effects

To enable sound effects:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Audio Settings page (Figure 42-1).

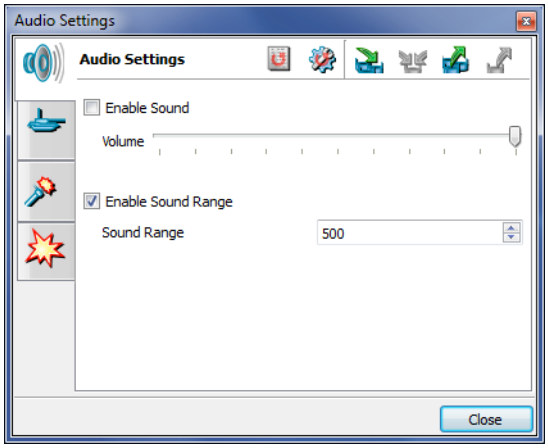


Figure 42-1. Audio Settings page

3. Select the Enable Sound check box.

You can also enable and disable sound effects by clicking the Audio button (▾) on the Display Settings toolbar.

42.2.1. Setting the Sound Volume

You can adjust the volume of sound effects.

To set the sound volume:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Audio Settings page (Figure 42-1).
3. Adjust the Volume slider.



You can also change the volume by clicking the arrow on the Audio button ( ) (on the Display Settings toolbar) and adjusting the Audio Volume slider.

42.2.2. Specifying the Range in which Sounds are Heard

You can specify that sound effects be played only if the observer is within a certain distance of the entities that can emit sounds. If sound range is enabled, as soon as the observer moves beyond the range from an entity, sound effects are shut off. If this option is disabled, sound effects are played for all entities in the scenario regardless of their distance from the observer. However, even if sound range is not enabled, as the distance between the observer and entities increases, sound levels drop off.

To specify the range in which sounds can be heard:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Audio Settings page (Figure 42-1).
3. Select the Enable Sound Range check box.
4. Specify a value in the Sound Range text box.

42.3. 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*. Sound files are associated with entity type enumerations. You can change the mappings and add new mappings.

You add and edit mappings between entity types, fire events, or detonation events and sound files on pages in the Audio Settings dialog box. You can add or edit mappings using mapping dialog boxes or directly in the mappings windows. The mapping dialog boxes behave differently depending on the state of the mapping window when you open the dialog box, as follows:

- If you click the Add Mapping button and have not selected a mapping in the Entity Mapping Settings window, you are presented with a default entity type (all wildcards).
- If you click the Add Mapping button and have selected an existing mapping, the dialog box displays the mapping you selected. From here, you can edit the existing mapping or create a new one.

When you edit a mapping, the result depends on whether you edit the entity or event enumeration or the sound file. The differences are based on the requirements for uniqueness in a mapping. The uniqueness of an entity type or fire event mapping is based on the entity type enumeration. The uniqueness of a detonation event mapping is based on the entity type enumeration and the detonation result value. Table 42-1 lists the results of different editing actions.

Table 42-1: Editing sound mapping files

Action	Result
Edit enumeration in dialog box.	New mapping is added. The previous mapping is not changed.
Edit enumeration in place.	New mapping replaces previous mapping.
Edit sound file in dialog box or in window.	Mapping to current entity or event is updated.
Edit channel in dialog box or in window.	Mapping to current entity or event is updated.
Edit detonation result in dialog box.	New mapping is added. The previous mapping is not changed.
Edit detonation result in window.	New mapping replaces previous mapping.

42.3.1. Mapping a Sound File to an Entity

An entity type enumeration can only have one sound file mapped to it. A sound file can be mapped to as many entity types as you want.

Adding a New Entity Sound Mapping

To add a new entity sound mapping:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Entity Sound Editor page (Figure 42-2).

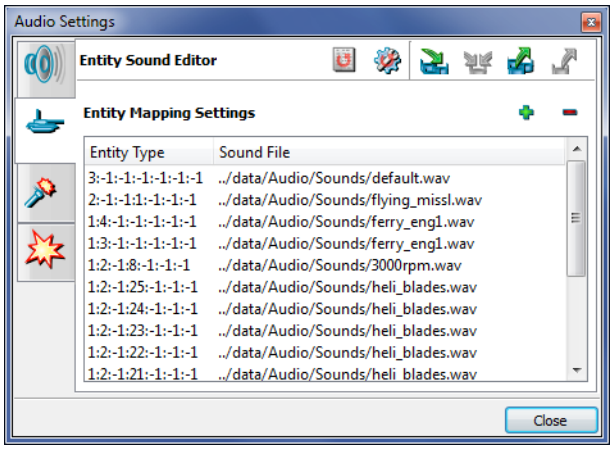


Figure 42-2. Entity Sound Editor page

3. Click the Add button (+). The Create New Entity Type Sound Mapping dialog box opens (Figure 42-3).

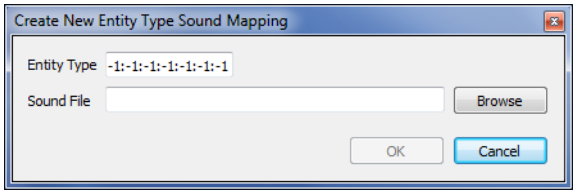


Figure 42-3. Create New Entity Type Sound Mapping dialog box



You cannot open the Create New Entity Type Sound Mapping dialog box with default options if one of the existing mappings is selected. If you have selected a mapping and want to return to the unselected state, click on white space in the mappings window or select one of the other tabs and then select the Entity Sound Editor tab again.

4. In the Entity Type box, type the enumeration for this mapping.

5. In the Sound File text box, type the pathname of a sound file or click Browse and select a file in the file chooser dialog box.
6. Click OK.

> The OK button does not become enabled until you specify a valid entity type and sound file name.

Creating an Entity Sound Mapping from an Existing Mapping

When you edit a mapping, you can either change the sound file associated with the current entity type or create a new mapping with a different entity type.

To create a sound mapping from an existing mapping:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Entity Sound Editor page (Figure 42-2).
3. Select the mapping that you want to use as the starting point.
4. Click the Add button (+). The Create Entity Type Sound Mapping from Existing Mapping dialog box opens (Figure 42-4).

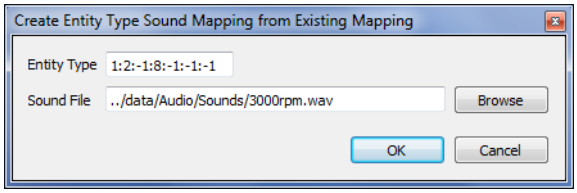


Figure 42-4. Create Entity Type Sound Mapping from Existing Mapping dialog box

5. To change the sound file associated with the entity type, type a new pathname or click Browse and select a sound file.
6. To create a new mapping, change the Entity Type enumeration.
7. Click OK. If you just changed the sound file, the mapping is updated. If you changed the entity type, a new mapping is added to the list.

Editing the Entity Type in the Mappings Window

If you edit the entity type in the mappings window (that is, without using a dialog box), it replaces the previous mapping.

To edit the entity type in the window:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Entity Sound Editor page (Figure 42-2).
3. Double-click the entity type you want to edit. It becomes editable (Figure 42-5).

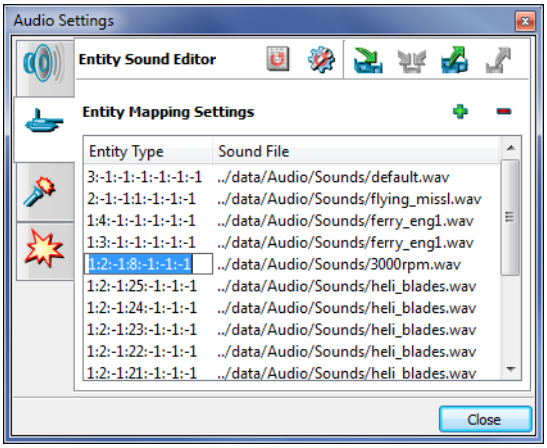


Figure 42-5. Editable entity type

4. Change the entity type enumeration.
5. Press **`Enter`**.

Editing the Sound Mapping in the Mappings Window

When you edit the sound mapping in the mappings window, the change does not affect any other mapping.

To edit the sound mapping in the window:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Entity Sound Editor page (Figure 42-2).
3. Double-click the sound mapping that you want to change. The Sound File dialog box opens.
4. Select the new sound file.
5. Click Open.

42.3.2. Mapping Sound Files to Fire and Detonation Events

When you map a sound file to a weapon fire or a detonation event, you can specify a channel for the sound. When you map a detonation sound, you can specify the detonation result for the mapping.

Specifying a channel for a sound event lets you control the number of sounds that can play concurrently. Sounds that are mapped to channel 0 can all play at the same time. Sounds mapped to any other channel can play only if no other event is using the channel. For example, suppose you map two weapon fire events to channel 1. One of the weapon type fires and the sound is generated. Then the second type of weapon event occurs. Since the first event is using channel 1, the second sound is not generated.

The SISO Enumerations document specifies a list of detonation results, such as Entity Impact, Ground Impact, and Air Hit. The detonation result is part of the detonation message sent over the network. You can specify that a detonation sound be played only for a certain type of detonation result. If you set the result to "any" it plays for any type of detonation. If, for a particular detonation type, you map one sound to "any" and another sound to a specific detonation result, the specific detonation result takes precedence.



You can change fire and detonation mapping settings directly, rather than using the sound mapping dialog box, just like you can change entity type sound mappings. For details, see the sub-procedures under "Mapping a Sound File to an Entity," on page 42-5.

To map a sound file to a weapon fire or detonation:

1. Choose **Settings** → **Audio Settings**. The Audio Settings dialog box opens.
2. Select the Fire Sound Editor page or Detonation Sound Editor page (Figure 42-6), as appropriate.

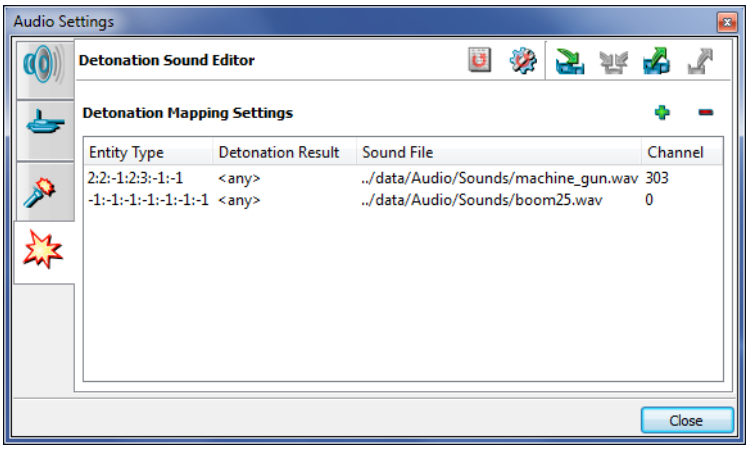


Figure 42-6. Detonation Sound Editor page

3. Click the Add button (). The Create New Detonation (or Weapon Fire) Sound Mapping dialog box opens (Figure 42-7).

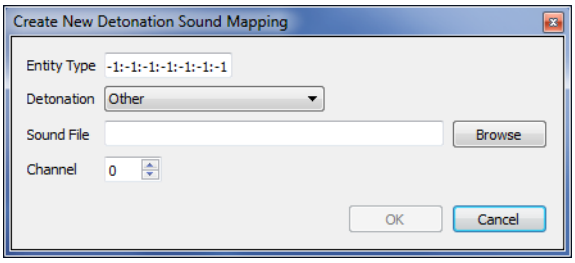


Figure 42-7. Create New Detonation Sound Mapping dialog box

When you click the Add button, if you have selected a mapping in the list, the Create New Sound Mapping from Existing dialog box displays the values for the selected mapping. If you change the entity type or detonation result, a new mapping is added with the new values.

4. In the Entity Type box, type the enumeration for this mapping.
5. Optionally, for detonation sounds, select an option on the Detonation list.
6. In the Sound File text box, type a pathname for the sound file you want to use or click Browse and select a file in the file choose dialog box.
7. Optionally, specify a channel in the Channel box.
8. Click OK.



IX. 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 43, *The Observer and Observer Modes*.
- Chapter 44, *Moving the Observer*.
- Chapter 45, *Attaching the Observer to Objects*.



43. 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 44, *Moving the Observer*. For details about attaching the observer to simulation objects and props, please see Chapter 45, *Attaching the Observer to Objects*.

Switching Between the 2D and 3D Projections 43-2
Observer Modes 43-2
Changing the Observer Mode 43-3
Creating and Editing Observer Modes 43-3
Creating an Observer Mode 43-4
Deleting an Observer Mode 43-5
Editing an Observer Mode 43-5
Renaming an Observer Mode 43-5
Adding an Observer 43-6
Adding an Observer on the Observer Settings Page 43-6
Adding an Observer in the Display Engine Configuration Editor 43-7
Removing an Observer 43-9
Introduction to XR Mode 43-10
Model Sets 43-11
Changing the Model Set 43-11
Communicating Information About the Observer 43-11
Making the Observer Visible to Other Applications 43-11
Viewing and Attaching to Other Observers 43-12
Displaying Models for Remote Observers 43-12

43.1. 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 43-3.) 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).

43.2. 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 43-3.

43.3. 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 43-2.

➤ To change the observer mode, choose **Observer** → **Set Observer Mode** → *mode*, or on the Observer Settings Toolbar (Figure 43-1), choose an option from the Observer Mode list.



Figure 43-1. Observer Settings Toolbar

43.4. 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.

43.4.1. 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 43-2).

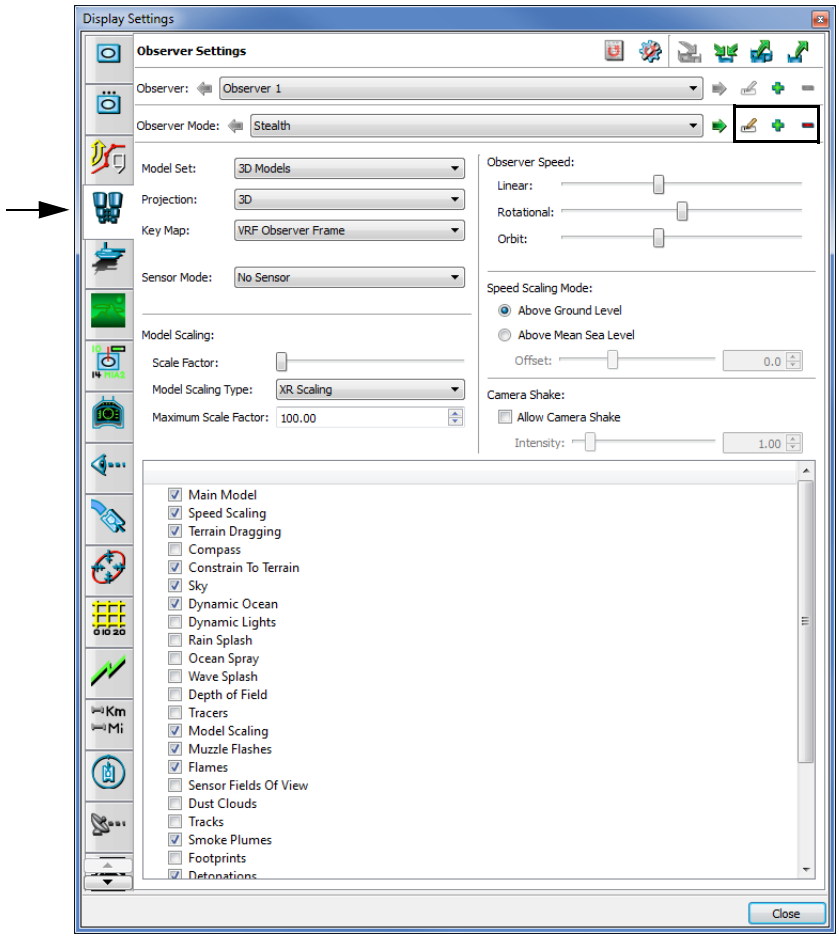


Figure 43-2. Observer Settings page

3. Click the Add button (+) to the right of the Observer Mode list. The Create New Observer Mode dialog box opens (Figure 43-3).

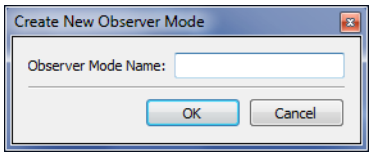


Figure 43-3. Create New Observer Mode dialog box

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.

43.4.2. 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 43-2).
3. In the Observer Mode list, select the observer mode that you want to delete.
4. Click the Delete button ( ).

43.4.3. 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 43-2).
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.

43.4.4. 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 43-2).
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.

43.5. 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 73, *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 73, *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.)

43.5.1. 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 43-2).
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 73-6.)

43.5.2. 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 43-4).

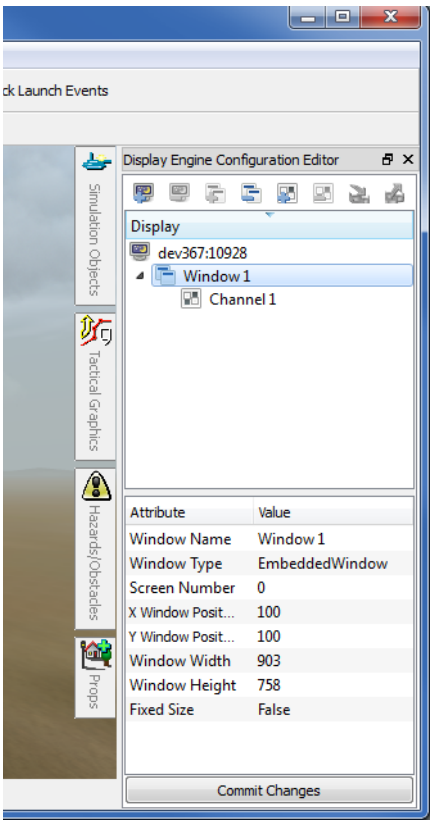


Figure 43-4. Display Engine Configuration Editor Panel

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 43-5).

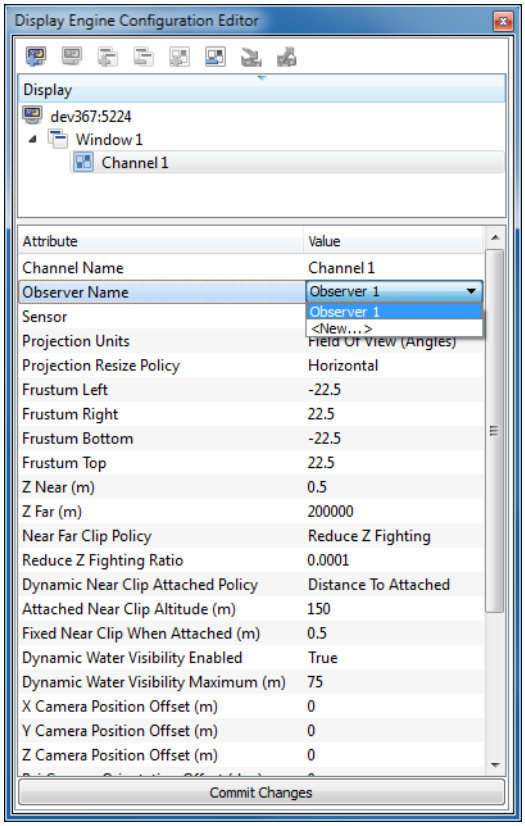


Figure 43-5. Observer Name attribute

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 73-6.)

43.6. 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 43-2).
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.

43.7. 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 43-6 illustrates model scaling with 3D colorized models.

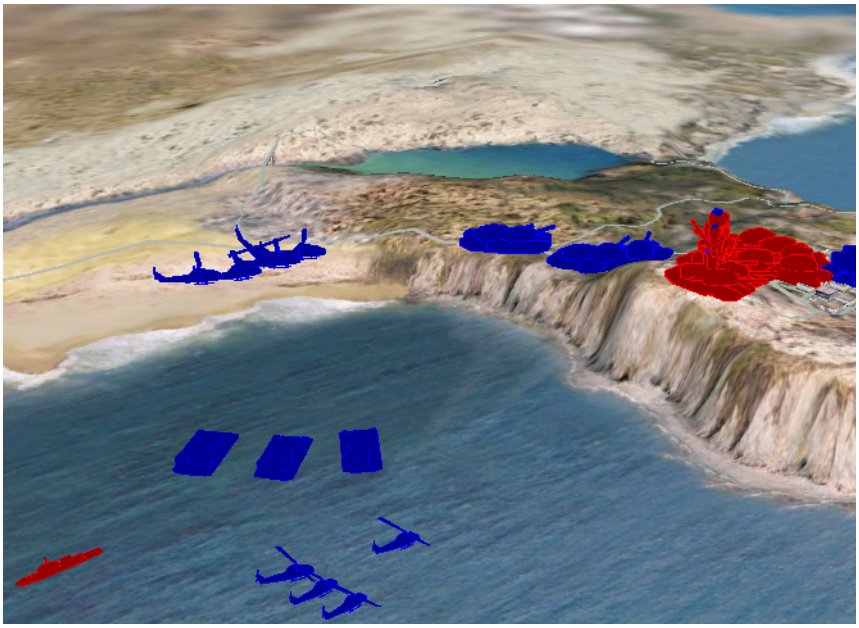


Figure 43-6. Model scaling and colorized models

For details about scaling models, please see "Exaggerating the Scale of 3D Entity Models," on page 19-24.

43.8. 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.

43.8.1. 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 43-5), 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.

43.9. 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.

43.9.1. 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.

43.9.2. 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 43-12.

43.10. 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 43-7).

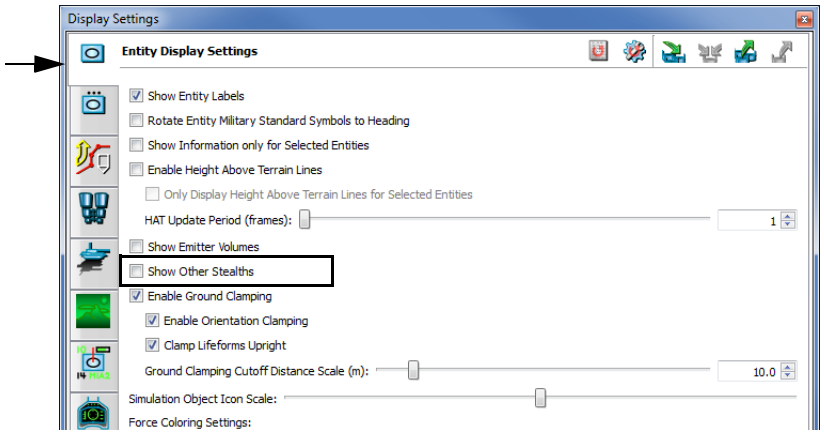


Figure 43-7. Display Settings, Entity Display Settings page

3. Select the Show Other Stealths check box.



44. Moving the Observer

This chapter explains how to move the observer (eyepoint) in the VR-Forces window.

Moving the Observer 44-3
Using the Compass to View the Observer's Heading 44-4
Editing the Compass's Attributes 44-5
Observer Movement Functions and Keyboard Mappings 44-6
3D Observer Movement Coordinate Systems 44-7
2D Observer Coordinate System 44-7
The Attached Context and Observer Movement (3D Only) 44-8
Moving the Observer from the Keyboard in the 3D View 44-10
Navigation Options 44-12
Orbiting 44-13
Moving the Camera 44-13
Moving the Observer Using the Mouse (3D View) 44-14
Dragging the Terrain 44-14
Changing the Observer's Orientation with the Mouse 44-15
Dragging the View 44-15
Orbiting the Terrain with the Mouse 44-15
Teleporting the Observer 44-16
Moving the Observer in the 2D View 44-16
Moving the Observer from the Keyboard in the 2D View 44-17
Moving the Observer Using the Mouse in the 2D Projection 44-18
Zooming the Observer 44-18
Zooming the Observer in the 3D Projection 44-19
Zooming the Observer in the 2D Projection 44-20
Setting the Scale Factor for Paged LODs 44-20

Zooming to Extents 44-20
Using the Depth of Field Effect 44-22
Configuring Depth of Field 44-24
Centering the Observer on a Simulation Object 44-25
Simulating Camera Jitter 44-25
Forcing Orientation North 44-26
Enabling and Disabling View Constraints (3D Only) 44-27
Controlling Navigation Speed 44-27
Enabling or Disabling Speed Scaling 44-27
Changing the Observer's Movement Speed 44-28
Saving and Recalling Views 44-29
Recalling a View 44-30
Deleting Views 44-31
Specifying the Startup View 44-31
Saving Views to a File 44-31
Loading Views 44-32
Controlling the Observer from Other Applications 44-32
Enabling the Processing of View Control Messages 44-32

44.1. 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 45, *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 44-1).

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.

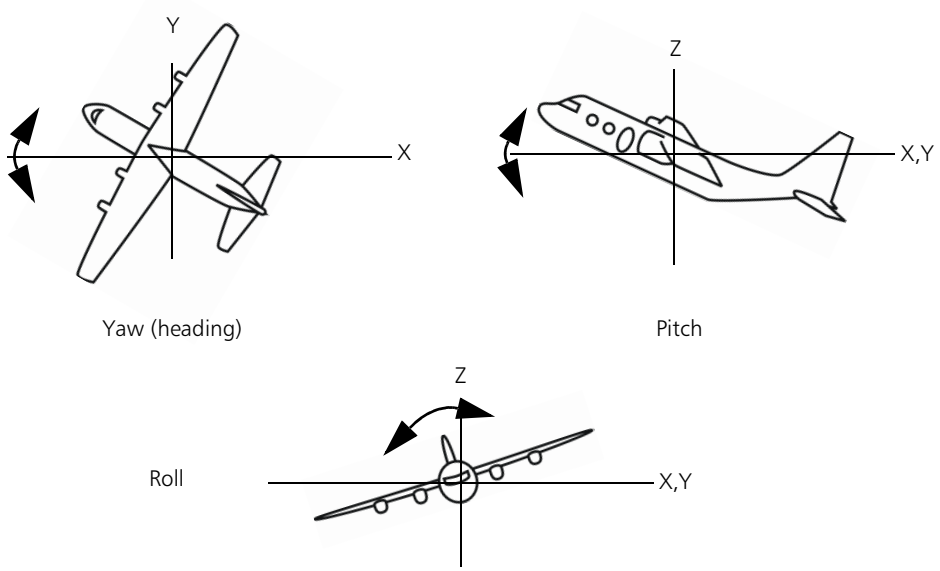


Figure 44-1. Orientation

The Observer Information Panel (Figure 44-4) displays the location and orientation of the observer in whichever coordinate system you select.

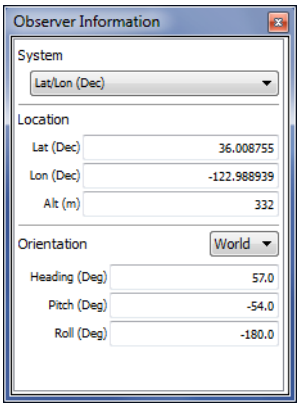


Figure 44-2. Observer Information Panel

44.1.1. 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 44-3, the observer's heading is approximately Northwest.

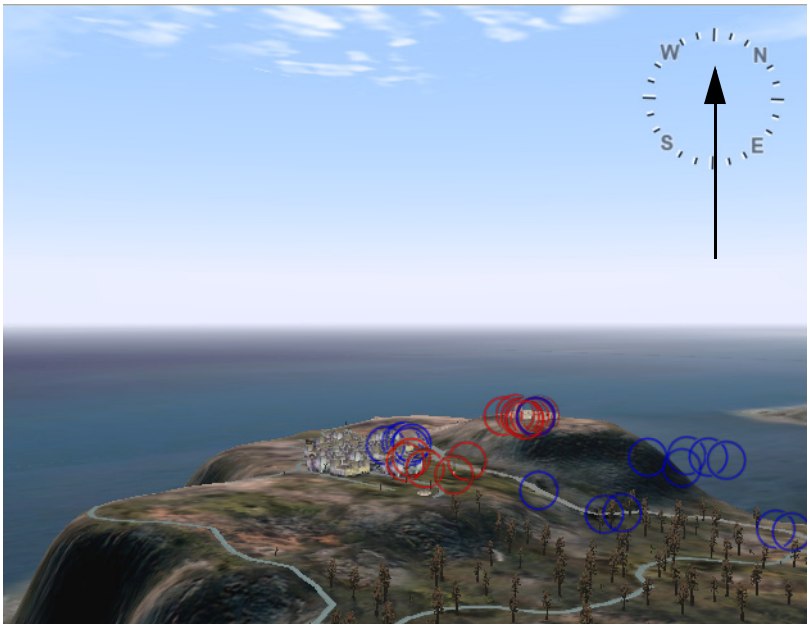


Figure 44-3. Compass

To enable or disable the compass:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
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**.

44.1.2. 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 57-3.)

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 73-8.

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 lightTick-Color 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.

44.2. 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 44-7.)

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 76, *Creating and Editing Key Mappings*.) This section describes how observer movement can vary based on the observer coordinate system and the attach mode.

Table 44-1 lists the observer modes and key map settings provided with VR-Forces.

Table 44-1: 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.		

44.2.1. 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.

44.2.2. 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.

44.2.3. 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 MÄK'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 44-4 and Figure 44-5 illustrate the differences between moving the observer in an attached context and without an attached context.

Figure 44-4 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.

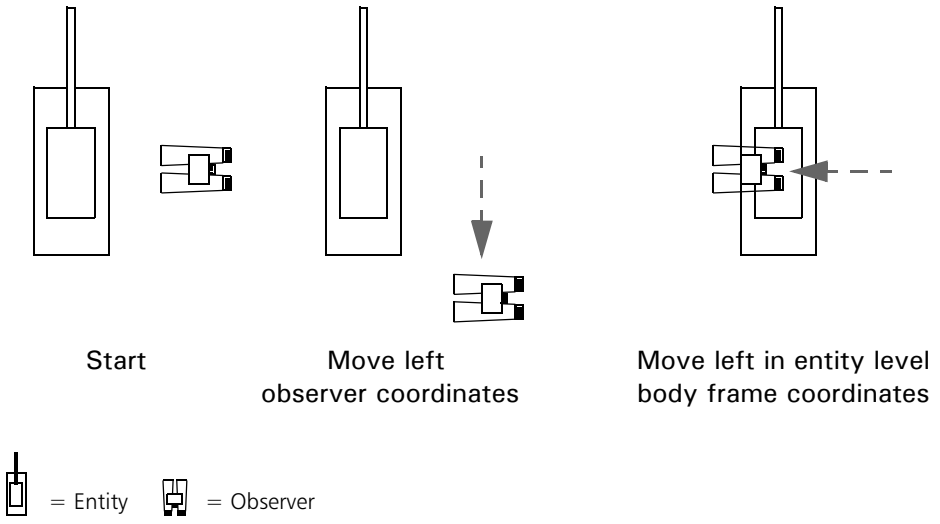


Figure 44-4. Observer movement in entity level body frame coordinates

Figure 44-5 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.

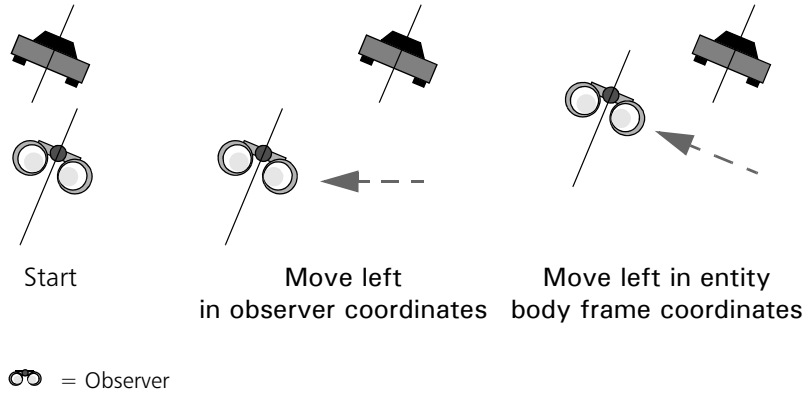


Figure 44-5. Movement in entity body frame coordinates

44.3. 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 44-3, 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 44-6 illustrates the positional movement keys. Figure 44-7 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 44-6. Primary movement keys, observer coordinates (3D)

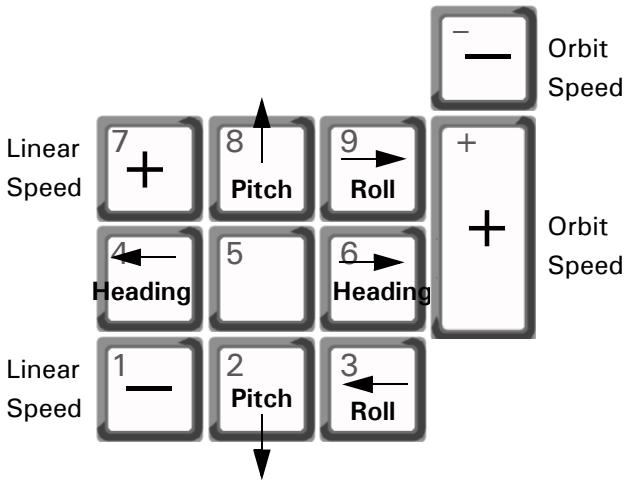


Figure 44-7. Keypad layout, heading, pitch, and roll (3D)

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.

44.3.1. 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 44-13. For details about moving the camera, please see "Moving the Camera," on page 44-13.



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.

44.3.2. 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.

44.3.3. 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.

44.4. 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 Mappings 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).

44.4.1. 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 disable terrain dragging, on the Observer Control Panel, clear the Allow Terrain Dragging check box (Figure 44-8).

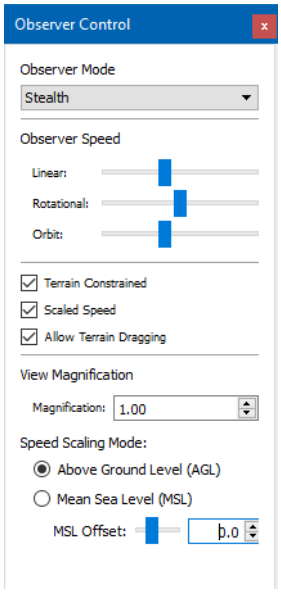


Figure 44-8. Observer Control Panel

44.4.2. 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.

44.4.3. 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.

44.4.4. Orbiting the Terrain with the Mouse

➤ To orbit the terrain with the mouse, press **`Ctrl`** and drag the mouse.

44.4.5. 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.
- **`Ctrl`**+**`Shift`**+click on the simulation object that you want to jump to.

44.5. 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 44-26.

44.5.1. Moving the Observer from the Keyboard in the 2D View

Figure 44-9 illustrates the positional movement keys for the 2D view. Figure 44-10 illustrates the keypad layout for changing heading and movement speed. For a complete list of movement functions and their keyboard and mouse mappings, please see *VR-Forces Quick Reference Card*.

! To use the keypad for navigation, NumLock must be on.

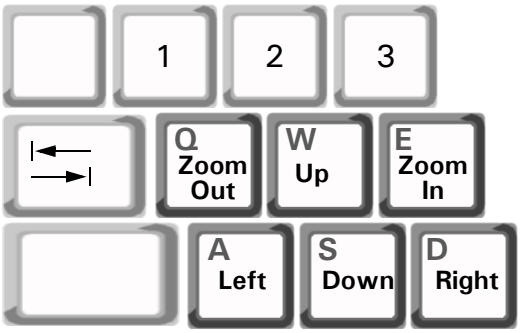


Figure 44-9. Primary movement keys, observer coordinates (2D)

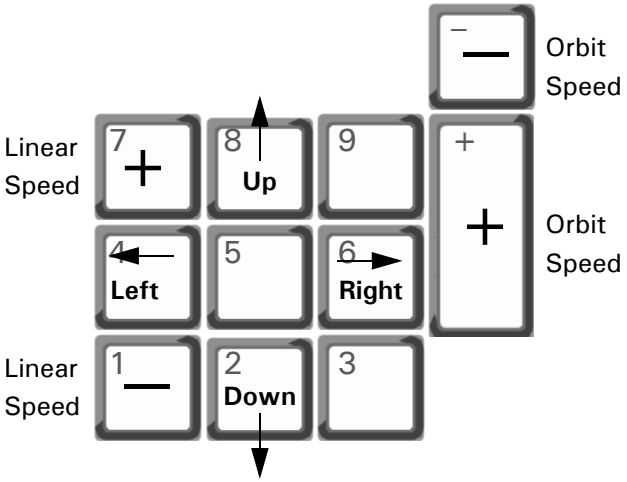


Figure 44-10. Keypad layout, heading, pitch, and roll (2D)

44.5.2. 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 44-14. You can change heading by orbiting, as described in the next section. (To see the full list of mouse mappings, view the Mouse Mappings 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 44-26.

➤ To orbit the terrain with the mouse, press the right mouse button and drag the mouse.

44.6. 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.

44.6.1. 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.

---

i 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.

44.6.2. 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 **Space bar**.

44.6.3. 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.
2. Adjust the Paged LOD Magnification Scale Factor slider.

44.6.4. 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 **Space bar**.
- If the observer is attached to a simulation object, the observer resets to the default attached view. This is the same behavior as pressing **Space bar**.
- 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 Session Options 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 will zoom to the entity extents.

44.7. 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 44-11, 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 44-12, 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 44-13 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 44-11. Depth of field (close)



Figure 44-12. Depth of field (far)



Figure 44-13. 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 37-2).
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**.

44.7.1. 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 44-14).

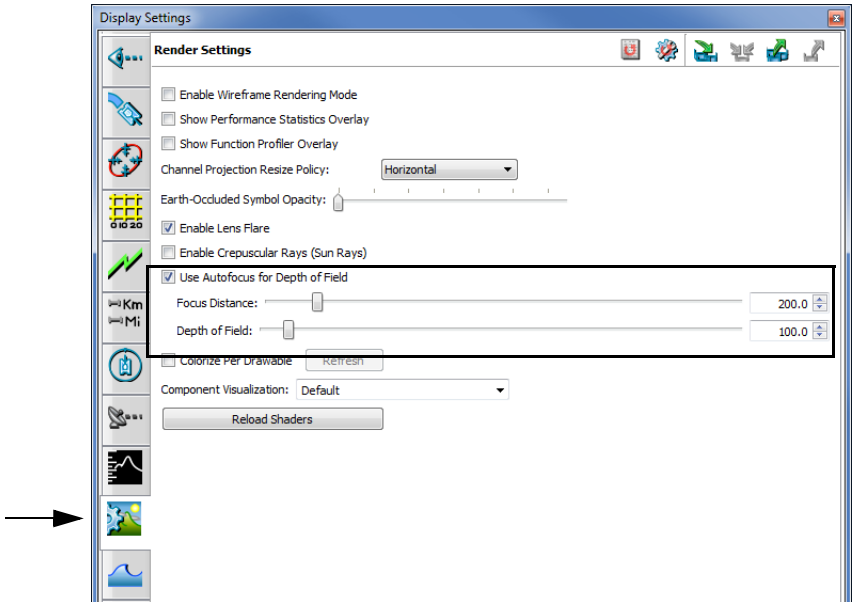


Figure 44-14. Depth of Field settings

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.

44.8. Centering the Observer on a Simulation Object

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 45-11.

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**.

44.9. Simulating Camera Jitter

You can simulate the jitter that cameras attached to aircraft or unsteady human hands may experience.

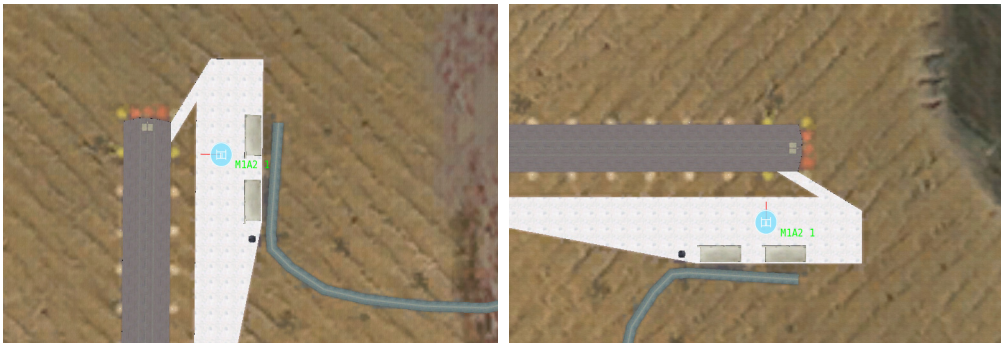
To simulate camera jitter:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 43-2).
3. In the Camera Shake section, select Allow Camera Shake.
4. Optionally, adjust the Intensity slider or change the value in the box.

44.10. 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 44-15 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 44-15. 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 37-2).
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**.

44.11. 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 44-8), select or clear the Terrain Constrained check box.

44.12. 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.

44.12.1. 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 44-8), 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 44-8), 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)

44.12.2. 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 44-8) 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.

44.13. 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. The procedures in this section describe that process.

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 44-16).

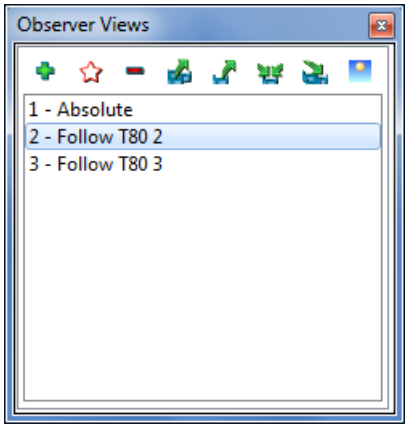


Figure 44-16. Observer Views Panel

Its buttons have the following meanings:

- Save the current view.
- Mark view to load on scenario startup.
- Delete the selected view.
- Save all views to a file.
- Save the selected view to a file.
- Load a saved views file and append it to the list.
- Load a saved views file and replace the current list.
- Clear the list.



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 44-16).
2. Click the Add button (). The current view is added to the list.

44.13.1. 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.

44.13.2. Deleting Views

To delete a view:

1. In the Observer Views Panel, select the view you want to delete.
2. Click the Delete button ( ).

Clearing the View List

➤ To remove all views from the list, in the Observer Views Panel, click the Clear List button ( ).

44.13.3. Specifying the 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.)

To specify a view as the startup view:

1. Select the view.
2. Click the star button ( ) on the Observer Views Panel.

44.13.4. Saving Views to a File

You can save views to a file independently of the scenario and load them at a later time. You can save an individual view or all of the views in the list.

The default location for saved views is *./appData/settings/vrfGui*.

To save the selected view to a file:

1. In the Observer Views Panel, click the Save Individual View button ( ). The Save Selected Observer dialog box opens.
2. Select the file to which you want to save the view, or type the name of a new file.
3. Click OK.

To save all views to a file:

1. In the Observer Views Panel, click the Save All 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.

44.13.5. 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, or you can replace the list in the panel with the saved views.

To load views and append them to the list of views:

1. In the Observer Views Panel (Figure 44-16), click the Load Views and Merge button ( ). The Open Observer Views dialog box opens.
2. Select the saved views file that you want to load.
3. Click OK.

To load views and replace the list of views:

1. In the Observer Views Panel, click the Load Views and Replace button ( ). The Open Observer Views dialog box opens.
2. Select the saved views file that you want to load.
3. Click OK.

44.14. Controlling the Observer from Other Applications

The VR-Forces front-end can respond to view control messages sent by applications such as the MÄK 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.

44.14.1. 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.



45. 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 45-2
Attaching to an Articulated Part 45-4
Attaching the Observer in Mimic Track or Tether Track Mode 45-4
Attaching the Observer in Mimic Location Track or Tether Location Track Mode 45-4
Filtering the Object Attachment List 45-5
Detaching from a Simulation Object 45-5
Hiding the Attached Model (3D Projection Only) 45-5
Attaching to Props 45-5
Improving Performance when Attaching to Ground Objects 45-6
Attach Modes in the 3D View 45-6
Absolute Mode 45-7
Follow Mode 45-7
Mimic Mode 45-8
Mimic Track Mode 45-9
Mimic Location Track Mode 45-9
Space Follow Mode 45-10
Tether Mode 45-11
Tether Track Mode 45-12
Tether Location Track Mode 45-12
Track Mode 45-13
Attach Modes in the 2D View 45-14
Attaching the Observer at Startup 45-14

45.1. 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.



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 45-4. 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 45-4.

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 45-1) 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 45-6 and "Attach Modes in the 2D View," on page 45-14.)

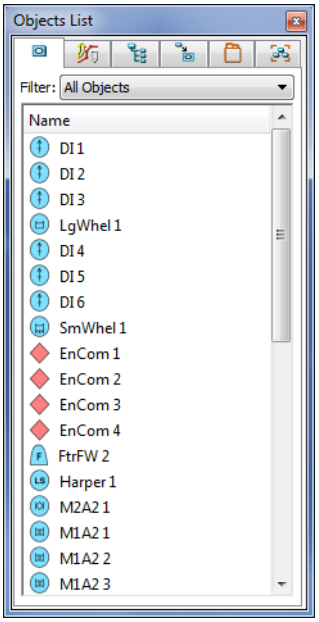


Figure 45-1. Objects List Panel

- 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 17-4.) On the Attachments Panel (Figure 45-2), or on the Attach Object Toolbar (Figure 45-3), click the Attach button () to attach with the current attachment mode.

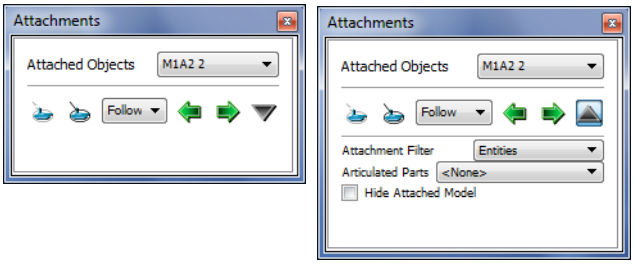


Figure 45-2. Attachments Panel



Figure 45-3. 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.
- 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 **`Space bar`**.

45.1.1. 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 45-2.
2. On the Attachments Panel (Figure 45-2), 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.

45.1.2. 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 popup 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.

45.1.3. 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 popup 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.

45.1.4. 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 45-2), 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.

45.1.5. Detaching from a Simulation Object

➤ To detach the observer from the simulation object to which it is attached, on the Attachments Panel (Figure 45-2), click the Detach button (), or press **z**.

45.1.6. 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 45-2), click the down arrow. The Attachments Panel expands.
2. Select the Hide Attached Model check box.

45.1.7. 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.

45.1.8. 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.

45.2. 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 45-2.

45.2.1. 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.

45.2.2. 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 45-4 illustrates the characteristics of Follow mode.

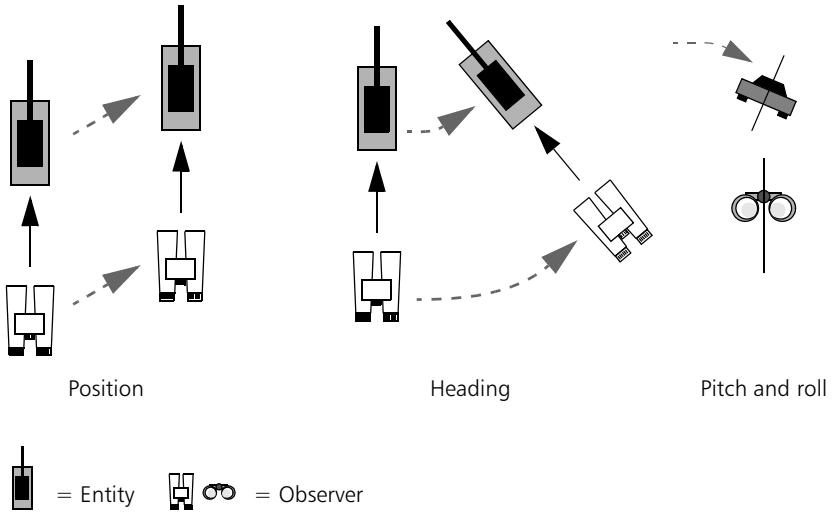


Figure 45-4. Follow mode

45.2.3. 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 45-5 illustrates the characteristics of Mimic mode.

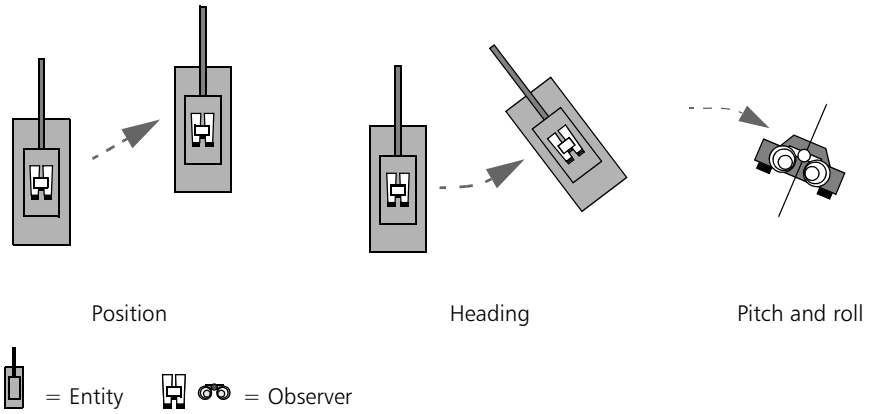


Figure 45-5. Mimic mode

45.2.4. 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 45-6 illustrates the characteristics of Mimic Track mode.

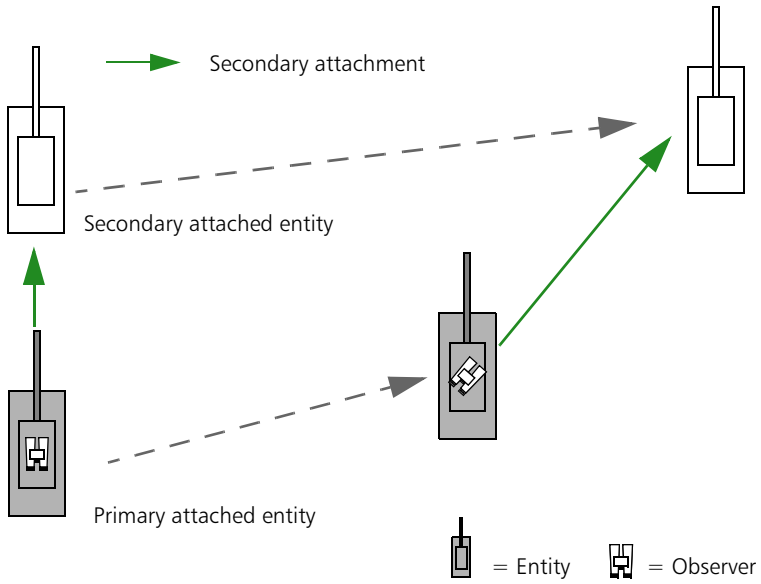


Figure 45-6. Mimic Track mode

45.2.5. 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.

45.2.6. 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 45-7 illustrates the characteristics of Space Follow mode.

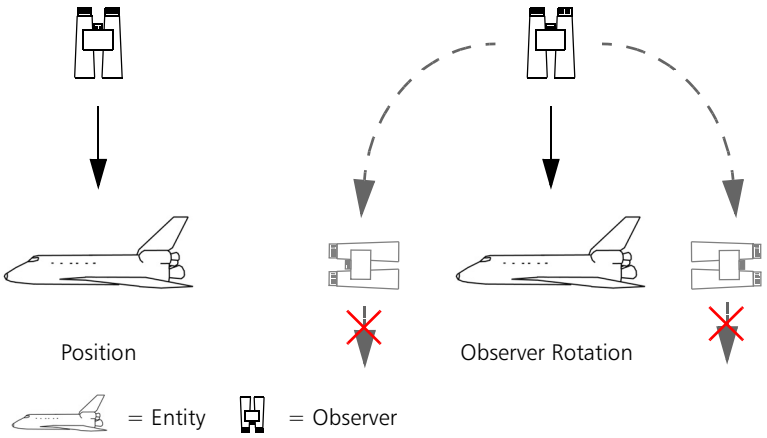


Figure 45-7. Space Follow mode

45.2.7. 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 45-8 illustrates the characteristics of Tether mode.

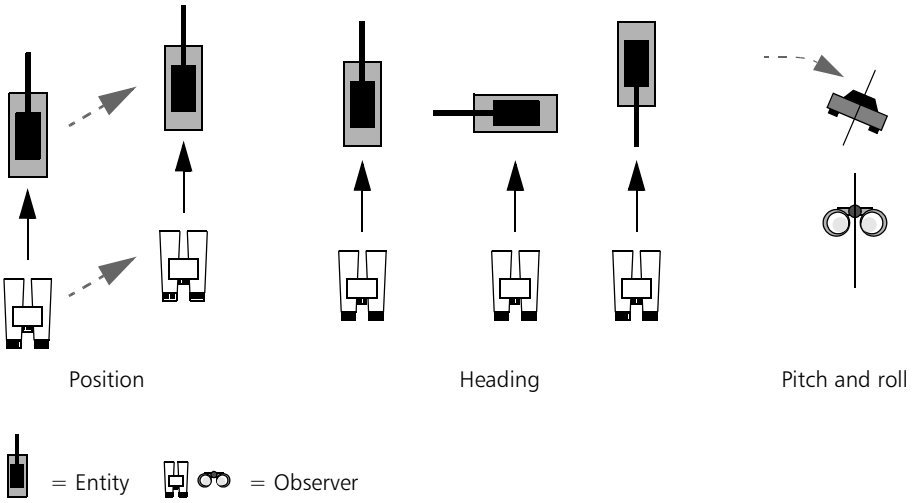


Figure 45-8. Tether mode

45.2.8. 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 45-9 illustrates the characteristics of Tether Track mode.

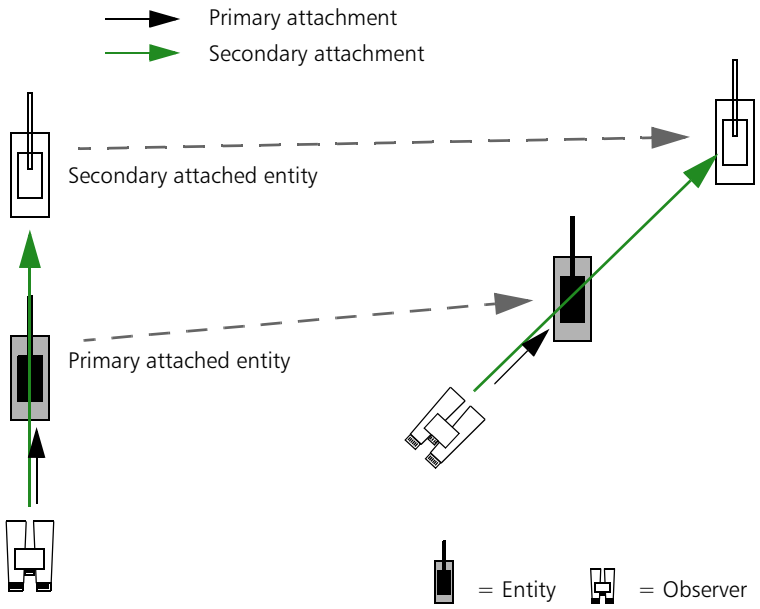


Figure 45-9. Tether Track mode

45.2.9. 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.

45.2.10. 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 45-10 illustrates the characteristics of Track mode.

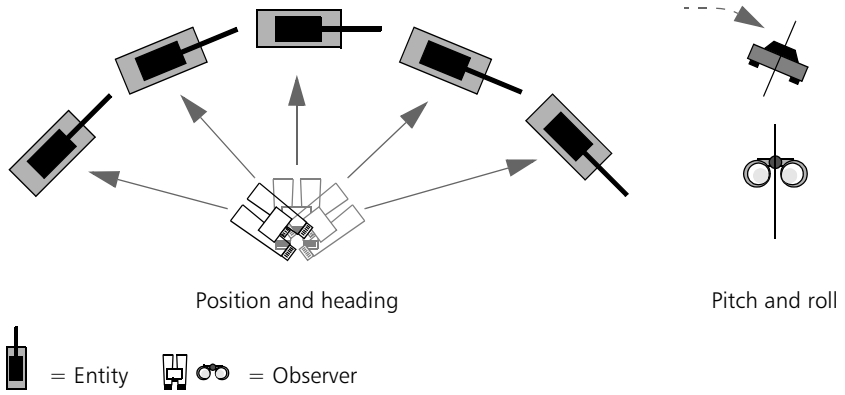


Figure 45-10. Track mode

45.3. 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 45-1. If you then switch back to the 3D view the original attach mode is restored.

Table 45-1: 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

45.4. 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 Startup View," on page 44-31.



46. Inset Views, Sensor FOVs, and HUDs

This chapter describes several kinds of special secondary windows plus head up displays.

Inset Views 46-2
Displaying Cockpit Displays (3D Only) 46-4
Displaying Instrument Panels 46-6
Configuring Entities to Use Instrument Panels 46-7
Displaying Sensor Views (3D Only) 46-8
Displaying a Sensor Field of View Cone 46-9
Controlling the Sensor View 46-10

46.1. Inset Views

You can display inset views of individual simulation objects. Figure 46-1 shows a helicopter hovering over the Ala Moana area in the VR-TheWorld Online - MAK Earth terrain with an inset view of a car driving through the town.



Figure 46-1. 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 73, *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. Right-click a simulation object (or its entry in the Objects List Panel). A menu opens.
2. On the menu, choose **Create Inset View**.

46.2. Displaying Cockpit Displays (3D Only)

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 59-10.

VR-Forces supports 2D cockpits (Figure 46-2) and 3D cockpits (Figure 46-3). 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 46-2. 2D cockpit



Figure 46-3. 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 46-4).

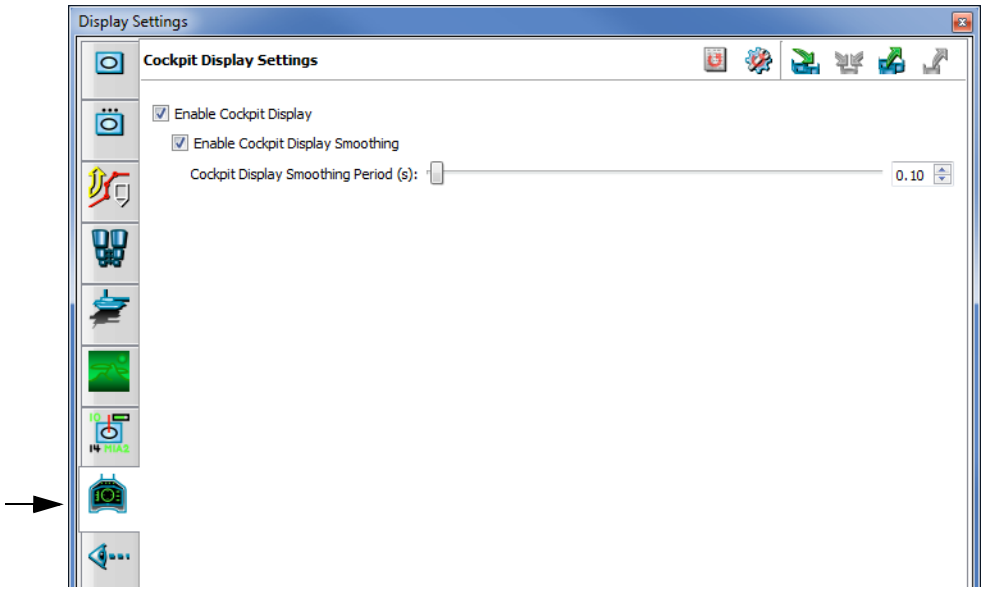


Figure 46-4. Cockpit display settings

3. To enable or disable cockpit displays, select or clear the Enable Cockpit Display check box.
4. 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.

46.3. 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 46-5). It does not affect the observer mode. You can organize multiple instrument panels to create your own virtual cockpit.



Figure 46-5. 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. On its right-click menu, choose **Create Instrument Panel**. A submenu is displayed.
3. Choose the instrument panel you want to open.

46.3.1. 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.

46.4. Displaying Sensor Views (3D Only)

VR-Forces can display the view from gimbaled visual sensors, such as a camera on a UAV. The view is displayed in an inset window that has information about the observer mode and area being viewed (Figure 46-6). The window has its own observer and you can change the observer mode in the view.

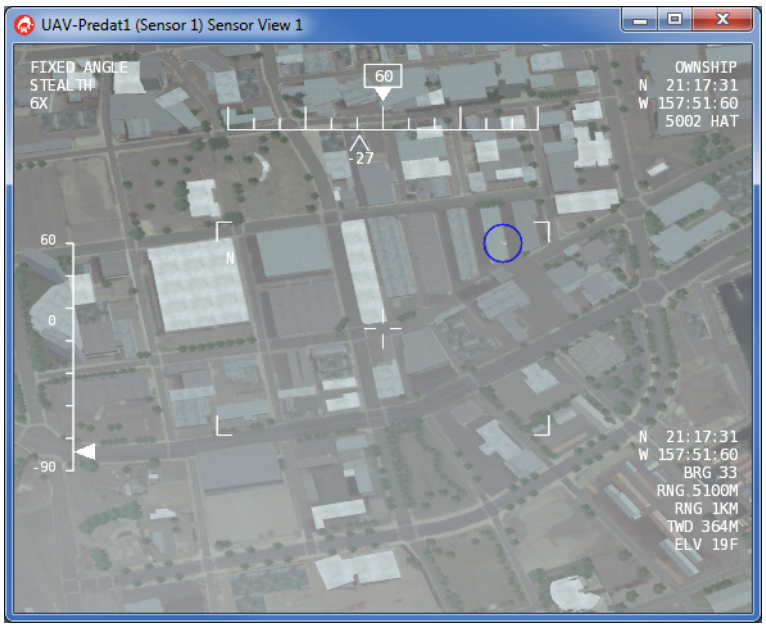


Figure 46-6. Sensor View

Each sensor view has a control panel (Figure 46-7) that you can use to move the sensor, change its aim, and zoom in and out.

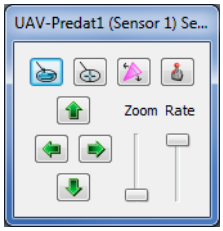


Figure 46-7. Sensor Control Panel

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. Right-click the entity and on the right-click menu choose **Create Sensor View** → *sensor*, where *sensor* is one of the sensors on the entity.

46.4.1. 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 46-8 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 46-8. 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 43-2).
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.

46.4.2. Controlling the Sensor View

When a sensor view opens, a control panel opens with it (Figure 46-7). 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 30-38.)
- 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 72-2.
- Zoom the sensor. (You can also zoom a sensor with the Sensor Zoom set data request. For details, please see "Sensor Zoom," on page 30-38.)
- Move the sensor field of view. You can change the sensitivity of the movement buttons.

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.



X. 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 or MTD[1] 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*.

1. An MTD file is a descriptive file created by the MÄK Terrain Database Tool (TDB Tool). Some legacy terrains may use MTD files, but the TDB Tool is no longer a recommended tool for creating terrains for VR-Forces.



47. 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) 47-2
Supported File Formats 47-2
- Terrain Data 47-3
- Elevation Data 47-3
- Supported Image Formats 47-4
- Supported Feature Data Formats in the Front-end 47-4

Feature Layers and Props 47-4
- Geometric Representation of Point Feature Data 47-5
- Feature Data Types Supported by the Back-end 47-5

Coordinate Systems 47-6
- The UTM Coordinate System 47-6
- Geocentric Terrain Databases 47-6

Loading Supported Databases 47-7
Using Large Terrain Databases 47-7
- Terrain Paging 47-8
- Terrain Streaming 47-9

47.1. 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 50-3.
- Specify a set of files to load using an 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 51-2 and Chapter 52, *Editing Earth Files*.

When you compose a terrain, you can save it as a MÄK 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 50-31.

47.2. 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.

47.2.1. Terrain Data

VR-Forces imports terrain data directly as terrain patches. It supports the following formats:

- MÄK terrain database (*.gdb*).
- MÄK encrypted data format (*.medf*).
- MÄK Stealth (6.2 and earlier) format (*.mtl*).
- MÄK terrain project or terrain descriptor file (*.mtp*, *.mtd*).
- OpenFlight (*.flt*).
- MetaFlight (*.mft*).
- CTDB 4, 7, 7L (*.c4b*, *.c7b*, *.c7l*).
- Shape terrain data (*.shp*).
- TerraPage (*.txp*).
- Collada (*.dae*).
- 3DS.
- OBJ.
- OSG.
- LWO.

47.2.2. 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.

47.2.3. Supported Image Formats

You can import images in the following formats:

- GeoTiff (*.tif*, *.tiff*).
- JPEG (*.jpg*, *.jpeg*).
- PNG.
- RGB.
- Windows bitmap (*.bmp*).
- GIF.
- ECW.
- 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. For more information about earth files, please see "Connecting to Terrain Servers," on page 51-2 and Chapter 52, *Editing Earth Files*.

47.2.4. 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.

47.3. 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 a terrain patch or from a feature layer.

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 MÄK Terrain Format (GDB) files, shapefiles (SHP), or DFAD data (DFD).

For more information, please see "Adding Props to a Terrain," on page 50-24.

47.3.1. 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 building or trees based on the data provided in the feature layer. As mentioned in the previous section, you can extract these objects 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 VR-TheWorld Online - MAK Earth, 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.

47.3.2. Feature Data Types Supported by the Back-end

The back-end supports the following types of feature data files:

- MÄK 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 55.7, "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 55.8, "Converting S-57 Feature Data to Shapefiles".

47.4. 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.

47.4.1. 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.

47.4.2. 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.

47.5. 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 non-MTF terrains 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.

47.6. 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 51-5. For details about connecting to terrain servers, please see "Connecting to Terrain Servers," on page 51-2 and "Creating a Terrain with an Earth File," on page 56-14.



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, particularly for 32-bit versions of VR-Forces.

47.6.1. 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 51-6.

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 A-1, in "The vrfSim.mtl Configuration File," on page A-3.)
- 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.

VR-Forces can display information about paged terrains that can help you understand how your terrain is performing and why objects may be interacting with it as they do. For details, please see "Displaying Information About Paged and Streaming Terrains," on page 51-7.

47.6.2. 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 MÄK's VR-TheWorld Server.



48. Displaying Terrain Information

This chapter describes how to change the display of terrain to provide different types of useful terrain information.

Introduction 48-2
Configuring Visible Surfaces (GDB Soil Types) 48-2
Coloring Terrain Based on Elevation 48-4
    Configuring Elevation Color Settings 48-5
Displaying Grid Lines 48-7
    Configuring Grid Lines 48-8
Displaying Contour Lines 48-9
    Configuring Contour Lines 48-10
Displaying Models and Terrain in Wireframe Mode 48-11
Filtering Earth File Layers 48-13
Displaying Feature Data (2D) 48-14
    Displaying a List of Vector Features 48-15
    Changing the Colors Used for Feature Data 48-16
Displaying Coordinates and Soil Type for a Location 48-17
Viewing the Map Scale 48-18
Terrain Databases Provided with VR-Forces 48-18

48.1. 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 50, *Composing Terrains*.

48.2. 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 48-1 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.

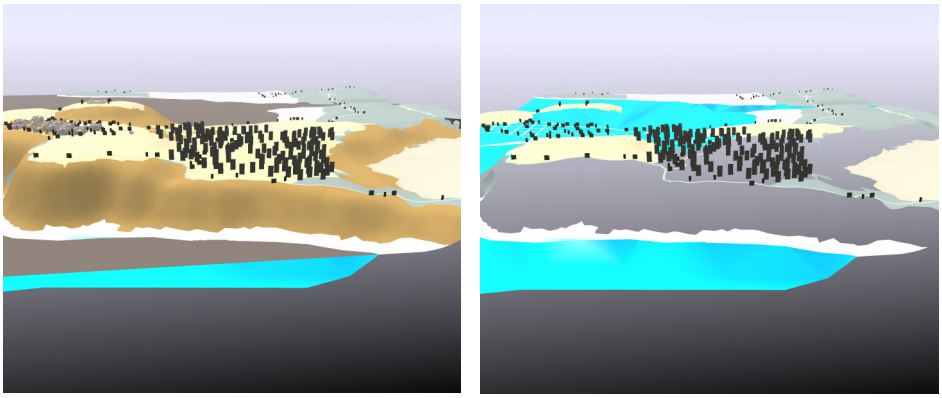


Figure 48-1. Visible surfaces toggled off

To display or hide GDB soil types:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Visible Surfaces page (Figure 48-2).

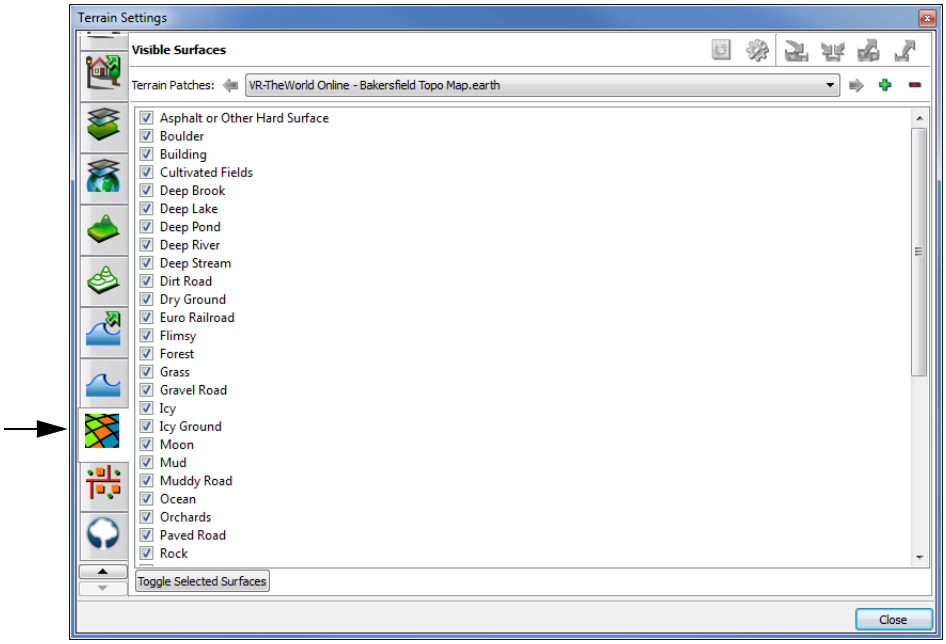


Figure 48-2. Visible Surfaces page

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.

48.3. 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 48-3).

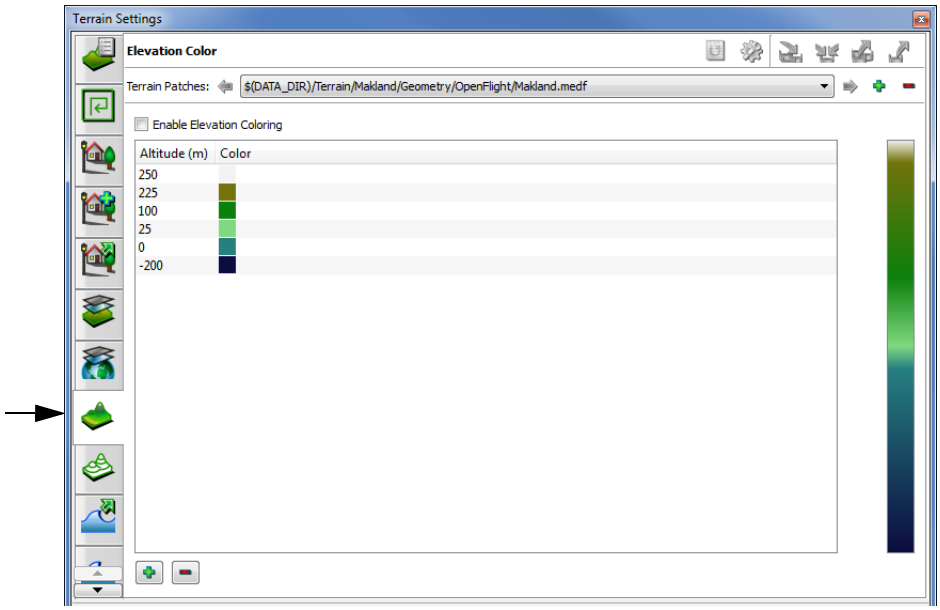


Figure 48-3. Elevation Color page

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.

48.3.1. 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 48-3).
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 48-3).
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 48-3).
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.

48.4. 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 48-4).

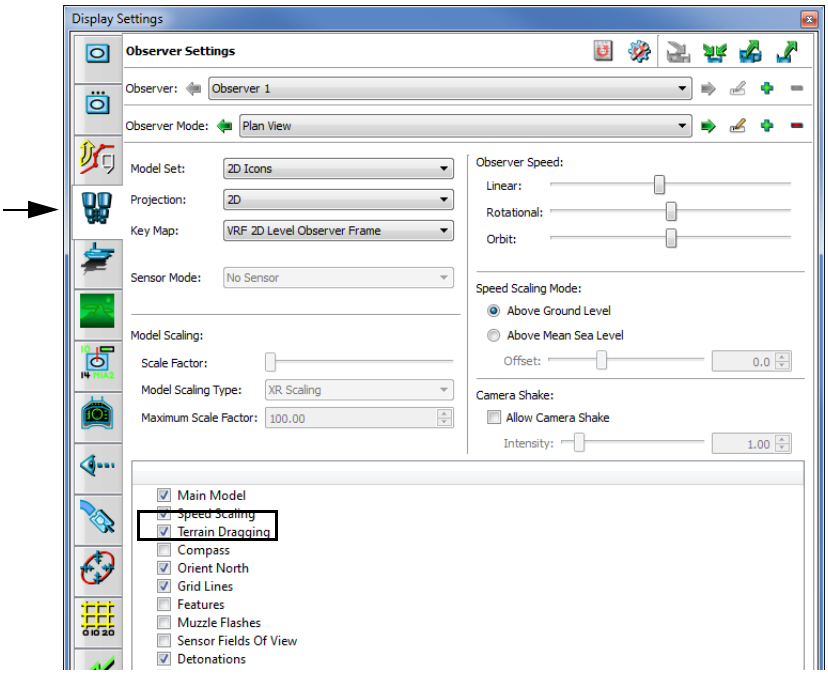


Figure 48-4. Observer Settings page

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**.

48.4.1. 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 48-5).

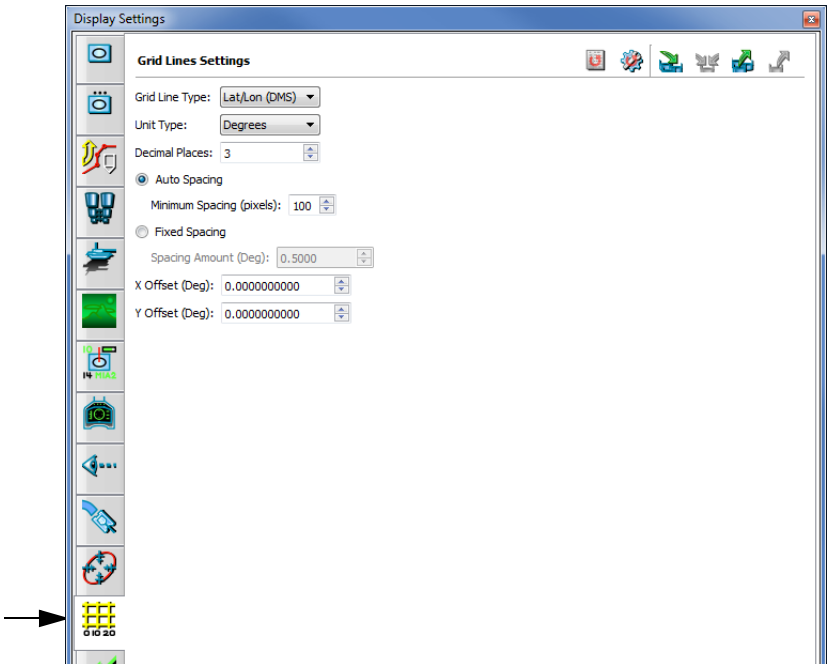


Figure 48-5. Grid Lines Settings page

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.

48.5. Displaying Contour Lines

You can display contour lines and intermediate contour lines on the terrain. Intermediate contour lines divide the contour line interval into fifths (Figure 48-6). You can configure the thickness of contour lines and the interval between them.



Figure 48-6. Contour lines and intermediate contour lines

To display contour lines:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Contour Lines page (Figure 48-7).

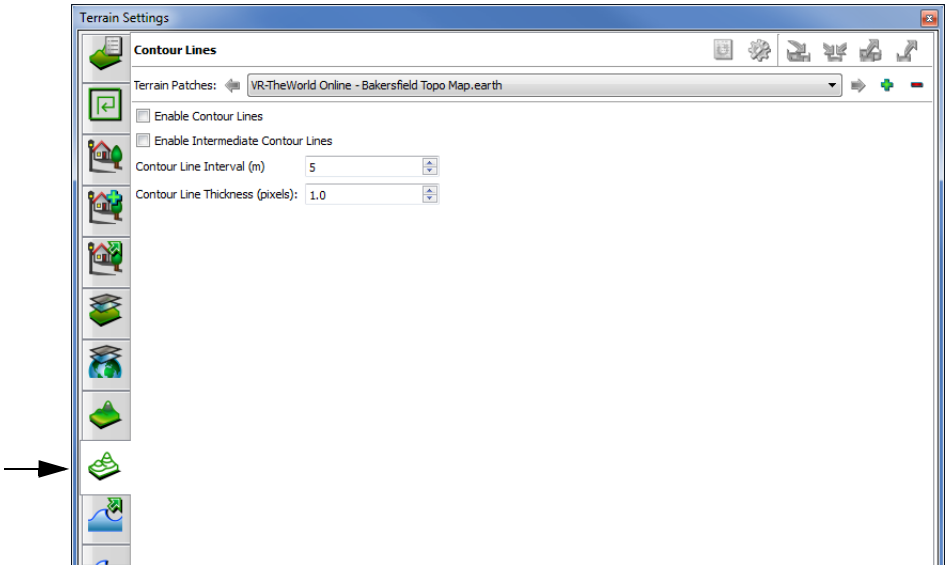


Figure 48-7. Contour Lines page

3. In the Terrain Patches list, select the terrain patch for which you want to enable or disable the display of contour lines.
4. Select Enable Contour Lines.
5. To enable intermediate contour lines, select Enable Intermediate Contour Lines.

48.5.1. Configuring Contour Lines

You can configure the thickness of contour lines and the interval between them.

To configure contour lines:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Contour Lines page (Figure 48-7).
3. In the Terrain Patches list, select the terrain patch for which you want to configure the thickness and interval of contour lines.
4. In the Contour Line Interval text box, specify the distance, in meters, between contour lines.
5. In the Contour Line Thickness text box, specify the thickness, in pixels, for contour lines.

48.6. Displaying Models and Terrain in Wireframe Mode

In wireframe mode, VR-Forces displays the polygons that make up the terrain and objects (Figure 48-8).



Figure 48-8. 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 48-9).

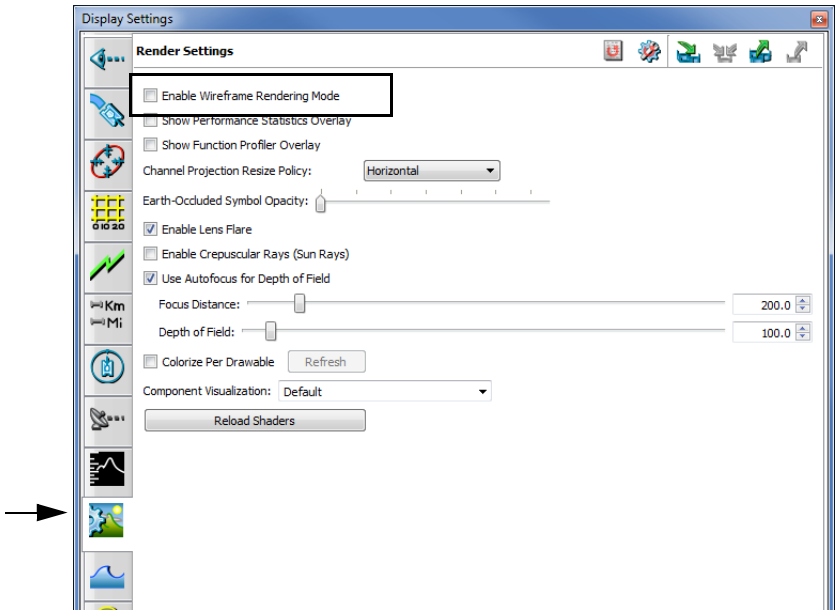


Figure 48-9. Render Settings page

3. Select the Enable Wireframe Rendering Mode check box.



You can also enable and disable wireframe mode by choosing **Settings** → **Wireframe Rendering**.

48.7. Filtering Earth File Layers

When you load 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.

- 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 48-10).

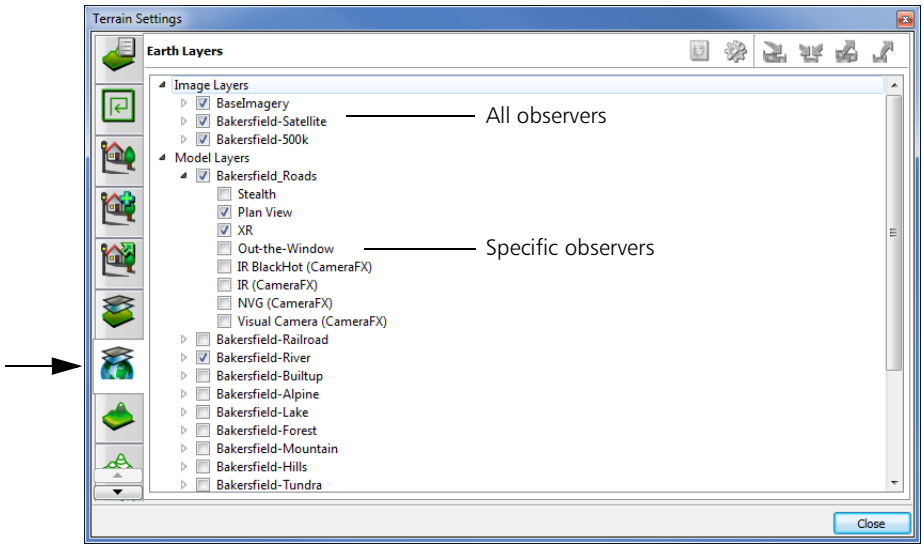


Figure 48-10. Earth Layers page

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.

48.8. Displaying Feature Data (2D)

If a terrain contains feature data, you can display it in the 2D view. Figure 48-11 shows a section of the Little Pond terrain with features disabled and enabled.



Figure 48-11. 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, choose **Observer** → **Features**.

48.8.1. 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 vice versa (Figure 48-12).

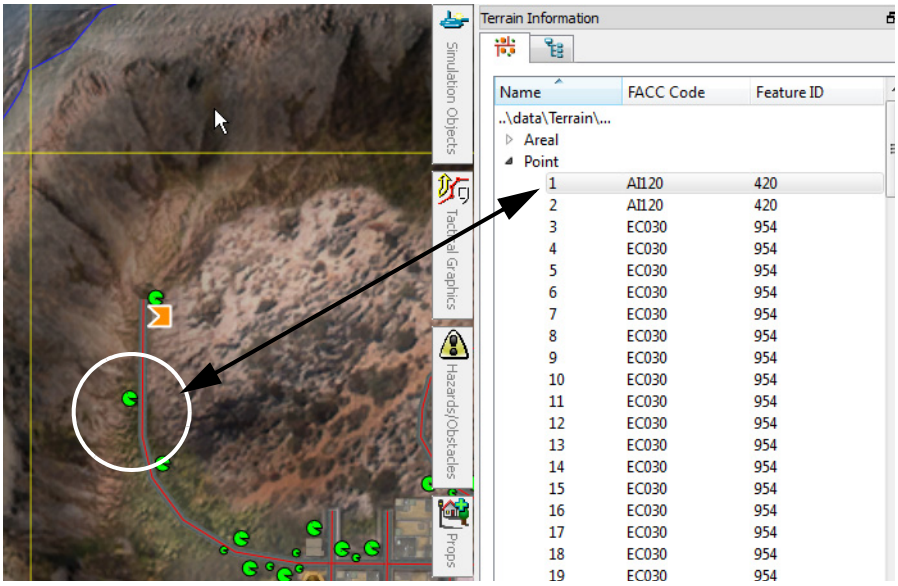


Figure 48-12. Vector feature selected

To display the feature list:

1. Choose **View** → **Terrain Information Panel**. The Terrain Information Panel opens.
2. Select the Vector Network tab (Figure 48-12).
3. Expand the top-level categories to see the lists of individual features.

48.8.2. 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 48-13, the Woodland feature in the Little Pond terrain was changed to pink. Compare it to the default color (Figure 48-11).



Figure 48-13. 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 48-14).

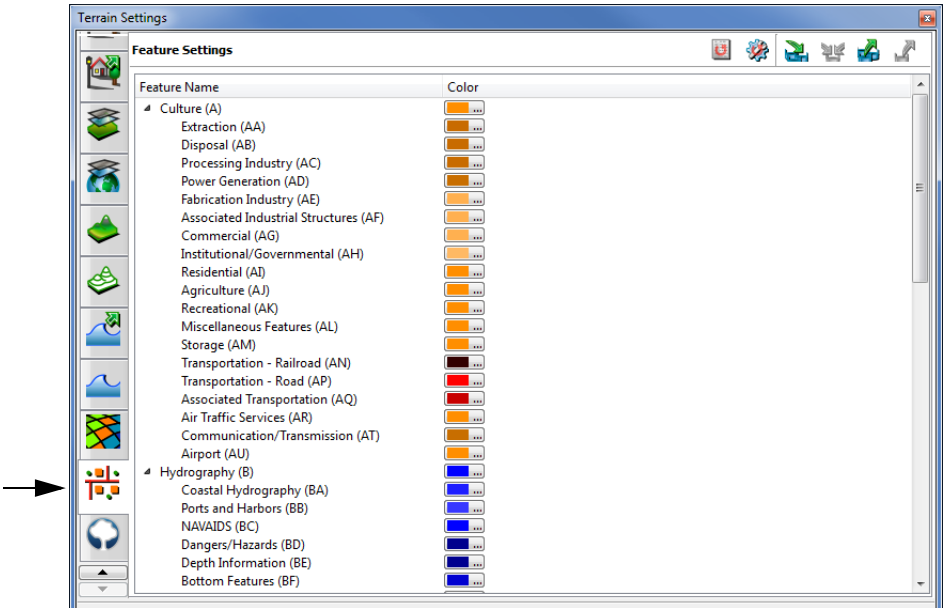


Figure 48-14. Feature Settings page

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.

48.9. 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**.

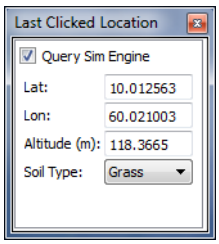


Figure 48-15. Last Clicked Location Panel

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.

48.10. 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 48-16). 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.

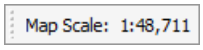


Figure 48-16. Map Scale Toolbar

➤ To enable or disable display of the Map Scale Toolbar, choose **View** → **Map Scale Toolbar**.

48.11. Terrain Databases Provided with VR-Forces

Table 48-1 describes the terrain databases provided with VR-Forces. Some have variants, which indicate the coordinate system or other variations. VR-Forces also has a supplemental data package that includes additional terrains. Please contact your MÄK salesperson for download details.

Table 48-1: Terrain databases shipped with VR-Forces

Database	Variation	Description
Ala Moana	Ala Moana.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.
Ground	Ground-db.mtf	UTM. OpenFlight. A generic relatively flat ground terrain.
Little Pond	LittlePond.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. This terrain was created in VR-Vantage using source data as described in "Composing a Terrain through the GUI," on page 56-2.
Local World Procedural Terrain	Local World Procedural Terrain.mtf	A database that generates terrain procedurally based on soil types. It does not use a satellite image.
Makland	Makland.mtf	UTM. OpenFlight. A fictional terrain created by MÄK. It includes a variety of terrain and feature data, including underwater features. It is placed in the Indian Ocean.

Table 48-1: Terrain databases shipped with VR-Forces

Database	Variation	Description
SanLuisO-bispo	SanLuisObispo.mtf	Flat earth. DTED. Imagery. A terrain that covers a portion of California, U.S.A.
	SanLuisObispo-TopoGdal.mtf	Geocentric. San Luis Obispo in topographic coordinates as local streaming terrain using an earth file.
VR-TheWorld-Online	VR-TheWorld Online.mtf	Lets you quickly connect to the publicly available VR-TheWorld Server for streaming terrain.
	VR-TheWorld Online - Little Pond.mtf	VR-TheWorldOnline with the Little Pond terrain inset.
	VR-TheWorld Online - MAK Earth - Base.mtf	VR-TheWorldOnline with high resolution data for Hawaii, Boston, and other areas, but does not support streaming features and some other capabilities.
	VR-TheWorldOnline - MAK Earth.mtf	VR-TheWorldOnline with support for high resolution areas and streaming features. This is the most performant version of VR-TheWorldOnline. This terrain is not supported on 32-bit versions of VR-Forces.
	VR-TheWorldOnline - Massachusetts S-57.mtf	VR-TheWorldOnline with S-57 data (as shapefiles) for Boston harbor. It also demonstrates streaming buoys and beacons.
	VR-TheWorldOnline - VR-Village Afghanistan.mtf	VR-TheWorldOnline with the VR-Village terrain inset in Afghanistan.
	VR-TheWorldOnline - Bakersfield Topo Map.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.
VR-Village	VR-Village.mtf	GDB, feature data, OpenFlight terrain. *VR-Village* is a fictional database created by VT MÄK. 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.
World	WorldFlatEarthNoElevation.mtf	Flat earth. Low resolution projected image of the world.

Table 48-1: Terrain databases shipped with VR-Forces

Database	Variation	Description
	WorldGeocentric.mtf	Geocentric. Low resolution projection of the world.
	WorldGlobalWater.mtf	A terrain with no elevation data for land masses.
	WorldProce-dural<type>.mtf	Procedurally generated terrains with various types of procedurally gener-ated terrain.



49. Using Terrain Profiles

This chapter explains what terrain profiles are and how to display and configure them.

Terrain Profiles 49-2
Displaying a Line's Terrain Profile 49-3
Configuring Terrain Profiles 49-3
Analyzing Terrain Using a Terrain Profile Line 49-4

49.1. 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 49-1, 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).

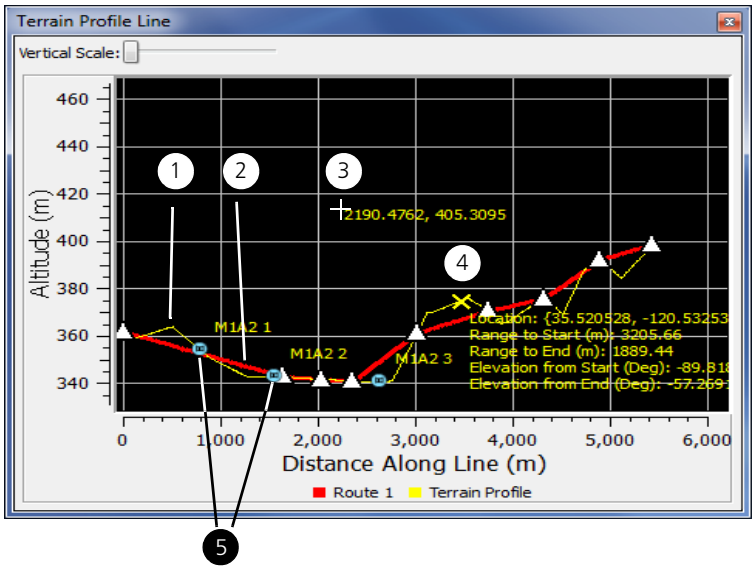


Figure 49-1. Terrain profile

49.1.1. 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 popup 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 37-8.

49.1.2. 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 49-2).

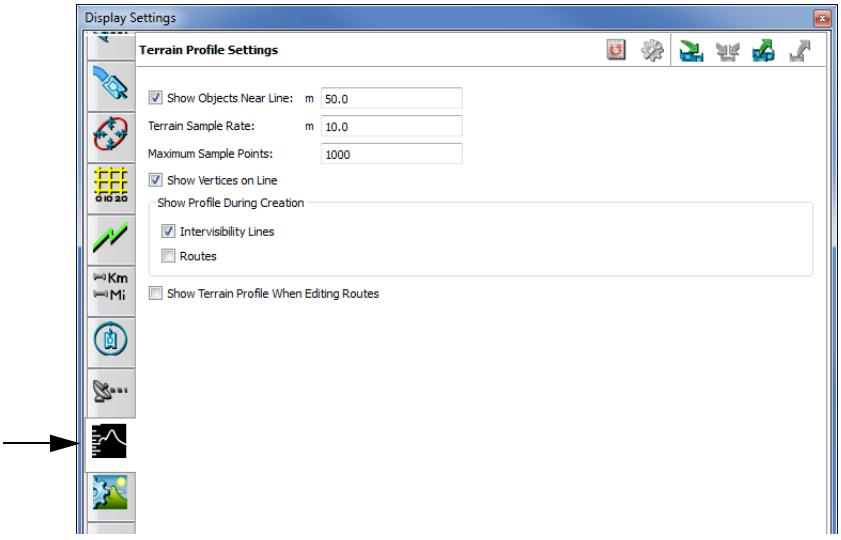


Figure 49-2. Terrain Profile Settings page

3. Change settings as desired. Table 49-1 describes the configuration options:

Table 49-1: Terrain profile parameters

Option	Description
Show Objects Near Line	If enabled, displays icons for simulation objects near the line. The specify 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.

49.2. 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 49-3).

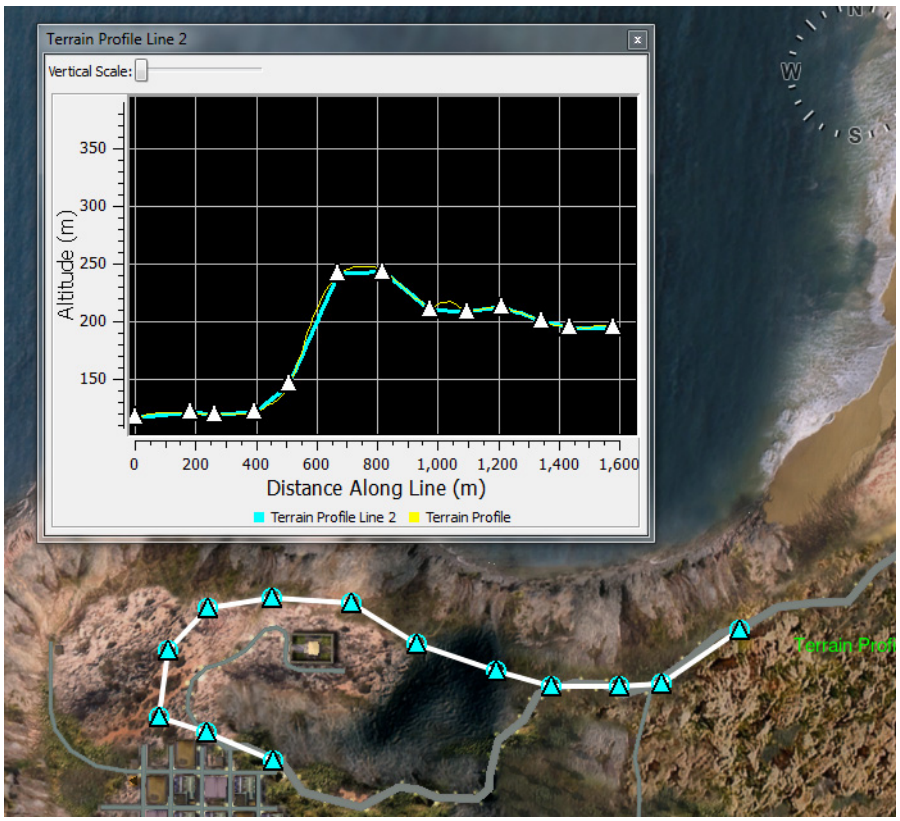


Figure 49-3. Terrain profile line

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 35-13.



50. 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 X, "Terrain,". "Composing a Terrain through the GUI," on page 56-2 is a tutorial that steps you through the entire process of creating a terrain. For information about paged and streaming terrains, please see Chapter 51, *Paged and Streaming Terrains*.

- Creating a Composed Terrain 50-3
  - Considerations and Limitations for Building Terrains 50-4
  - Saving a Terrain 50-5
- Adding Elevation Data (Terrain Patches) to a Terrain 50-5
- Adding Images to a Terrain 50-7
  - Changing the Display Order of Raster Maps 50-10
- Adding a Feature Layer 50-11
- Adding a Dynamic Ocean Layer 50-13
  - Extracting Water Textures and Adding an Ocean Layer 50-15
  - Adding a Dynamic Ocean Layer Directly 50-16
  - Setting the Ocean LOD Elevation 50-16
  - Configuring the Ocean Height Map 50-17
- Specifying Simulation Ocean Settings 50-19
- Using Shader-based Effect Maps 50-20
  - Debugging Shaders 50-22
  - Reloading Shaders 50-23
- Adding Props to a Terrain 50-24
  - Extracting Props from a Terrain Patch 50-24
  - Adding Props from a Feature Layer 50-26
  - Viewing a List of Props 50-28
  - Selecting Props 50-28

Setting the Opacity of Props 50-29
Changing a Prop's Type 50-30
Changing a Prop's Position or Orientation 50-30
Changing a Prop's Model Definition 50-30
Configuring File Caching 50-31
Caching Terrain Server Data 50-32
Caching osgEarth Terrain Data Offline 50-32
Enabling Texture Compression 50-33
Clearing the File Cache 50-33
Displaying DDS Textures Correctly 50-34
Flipping DDS Textures Globally 50-35
Converting OpenFlight File Texture Data 50-36

50.1. 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 51, *Paged and Streaming Terrains* and Chapter 52, *Editing Earth Files*.) The general procedure for composing a terrain this way is as follows:

1. Choose **File** → **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 50-5.
3. Optionally, add imagery, as described in "Adding Images to a Terrain," on page 50-7.
4. Optionally, add feature data, as described in "Adding a Feature Layer," on page 50-11.
5. Optionally, add a dynamic ocean layer, as described in "Adding a Dynamic Ocean Layer," on page 50-13.
6. Optionally, add props or extract props, as described in "Adding Props from a Feature Layer," on page 50-26 and "Extracting Props from a Terrain Patch," on page 50-24.
7. Save the terrain, as described in "Saving a Terrain," on page 50-5.

"Composing a Terrain through the GUI," on page 56-2, demonstrates how to create a composed terrain.

50.1.1. Considerations and Limitations for Building Terrains

As you compose a terrain, consider the following issues:

- VR-Forces supports DTED level 0, 1, and 2. DTED is loaded using a Flat Earth projection. The default post spacing for DTED level 0 is 1. For other levels it is 5. You can load a 1 degree cell.
- If you load more than one DTED cell, they will be placed on top of each other. If you need to load multiple DTED cells, you can do so using an earth file. For details, please see "Loading Multiple DTED Files," on page 52-6.
- 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.



This section does not apply to terrains loaded using an earth file.

For information about EGM96 and geoid files, please see the following URL:

http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm96/egm96.html

50.1.2. 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 50-31.) 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** → **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.

50.2. 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 50-4.

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 50-1).

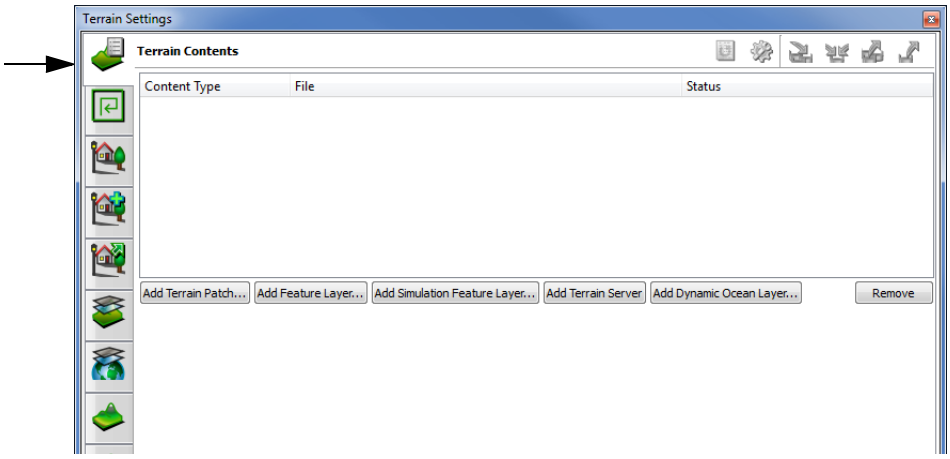


Figure 50-1. Terrain Contents page

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 50-2).

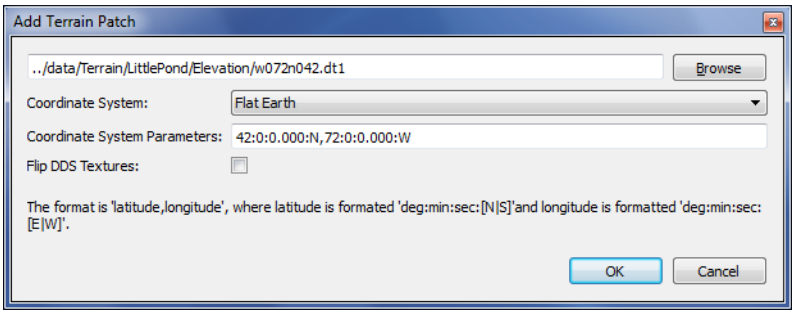


Figure 50-2. Add Terrain Patch dialog box

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 50-34.)
7. Click OK. The terrain patch is added.
8. Optionally, add additional terrain patches.
9. Save the terrain.

50.3. 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 50-3).

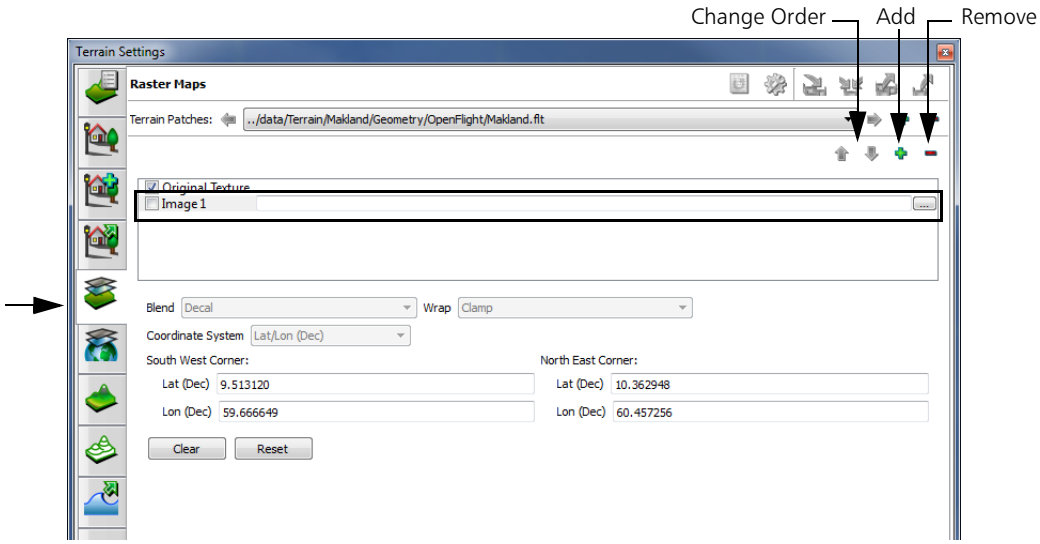


Figure 50-3. Raster Maps page

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 50-4).

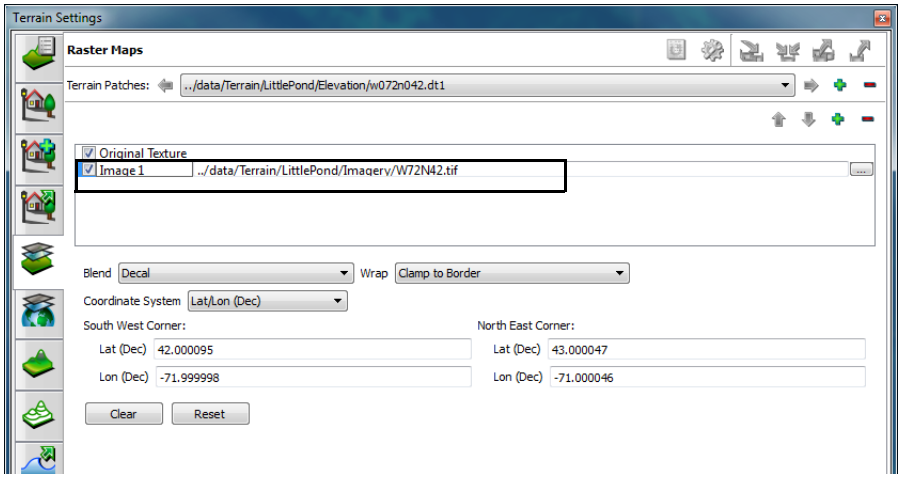
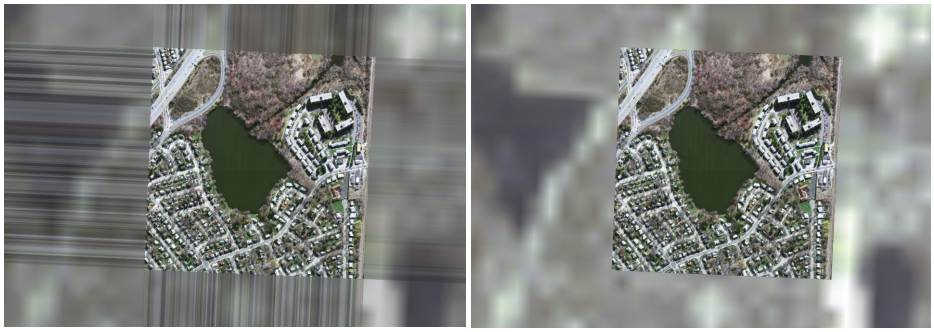


Figure 50-4. Making an image visible

7. Optionally, specify the attributes of the image, as follows:
   - Blend specifies how the image interacts with underlying images, as follows:
     - 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 follows, and as illustrated in Figure 50-5 and Figure 50-6:
     - 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.

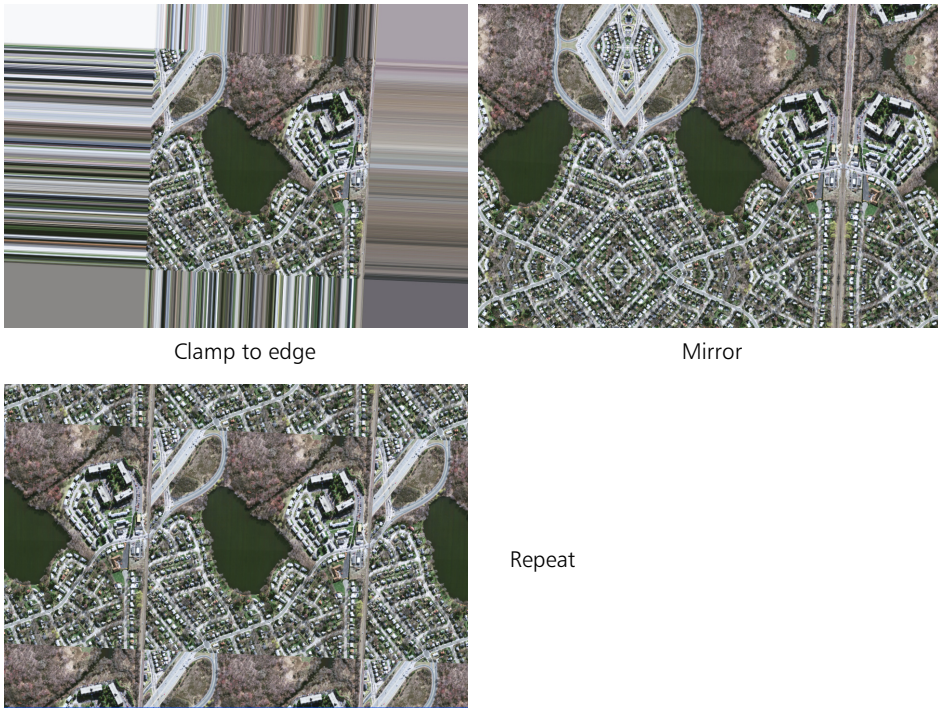
– Coordinate System specifies the coordinate system to use to place the image. Most options are self-explanatory. The Texture coordinate system specifies UV (X,Y) coordinates for placing textures on the terrain.

– 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 50-5. Raster image wrap effects (Clamp and Clamp to Border)



Clamp to edge | Mirror

Repeat

Figure 50-6. Raster image wrap effects (Clamp to Edge, Mirror, Repeat)

50.3.1. 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 in VR-Forces.

Figure 50-7 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 50-7. 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 50-3).
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.

50.4. 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 MÄK 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 55, *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 55-7.

- 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 50-24.
- 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, *VR-TheWorld Online - Massachusetts S-57.mtf*.

To add a feature layer:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Terrain Contents page (Figure 50-8).

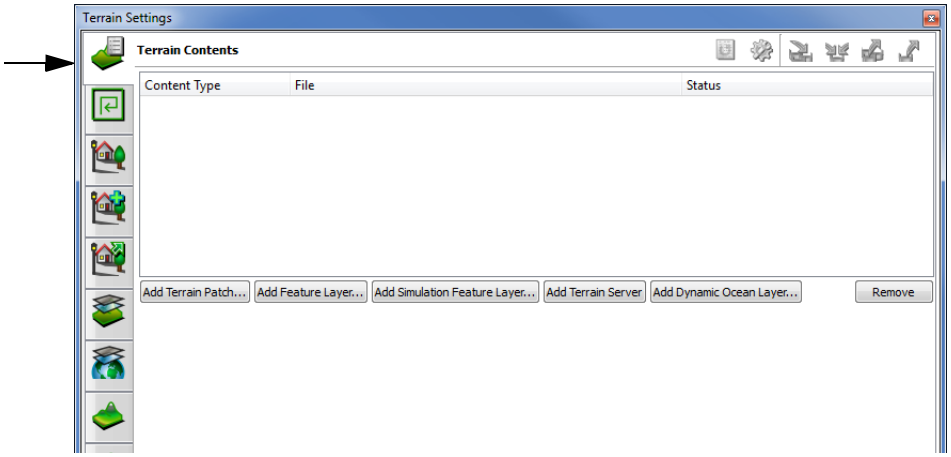


Figure 50-8. Terrain Contents page

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 50-9).

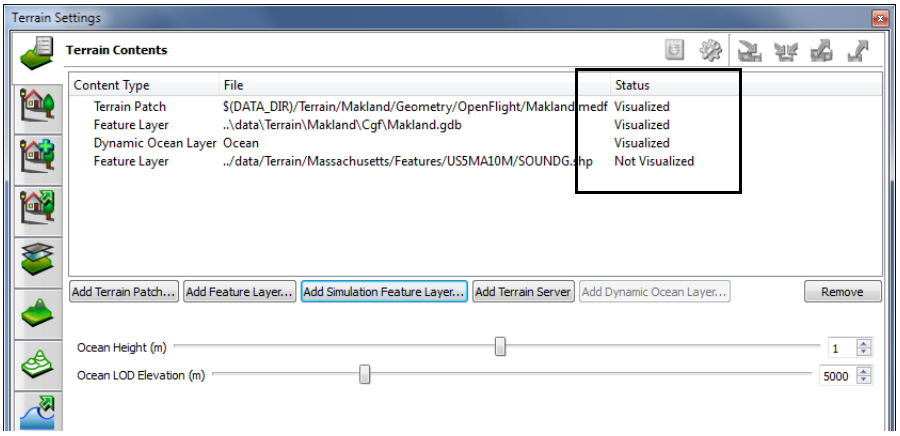


Figure 50-9. Terrain contents status

50.5. 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 50-10). This problem occurs because the ocean layer and the texture are both at the same height (0, or sea level).

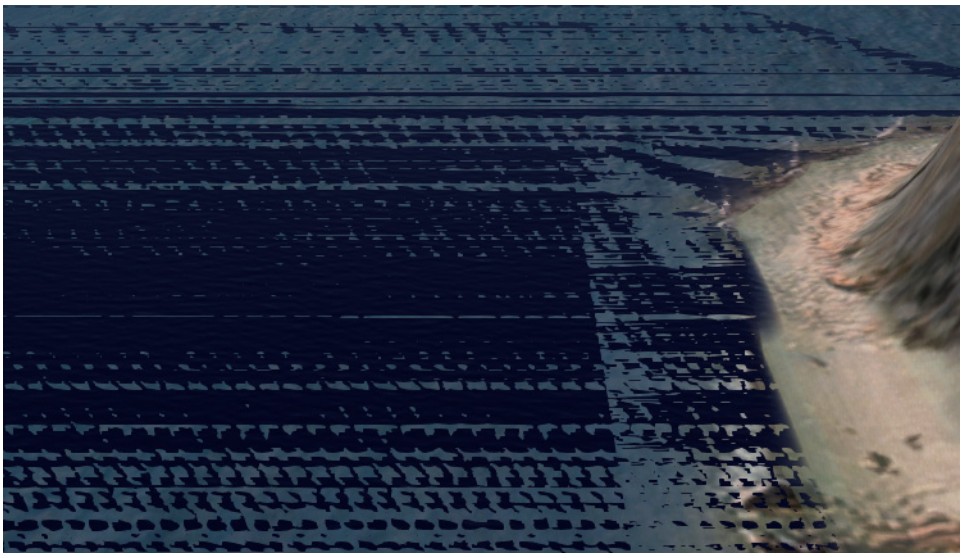


Figure 50-10. Z-fighting

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 50-11 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.

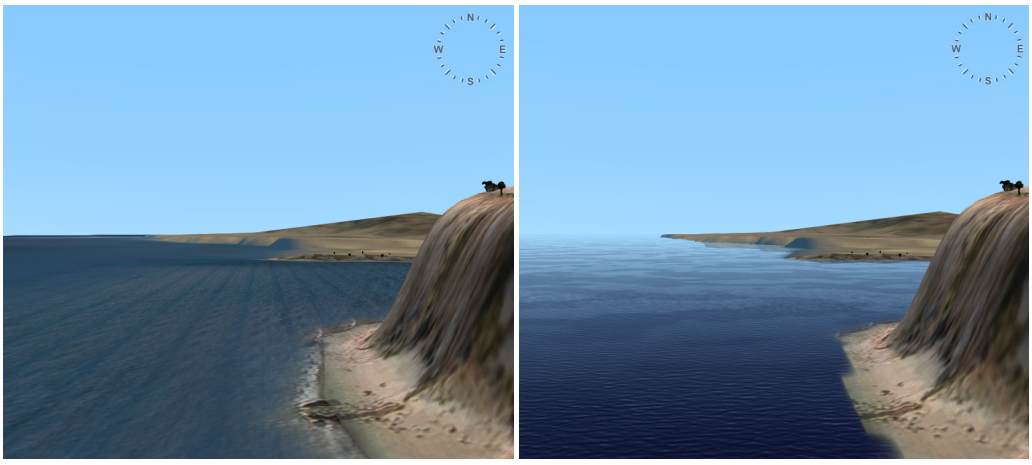


Figure 50-11. Ocean height

50.5.1. Extracting Water Textures and Adding an Ocean Layer

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 50-12).

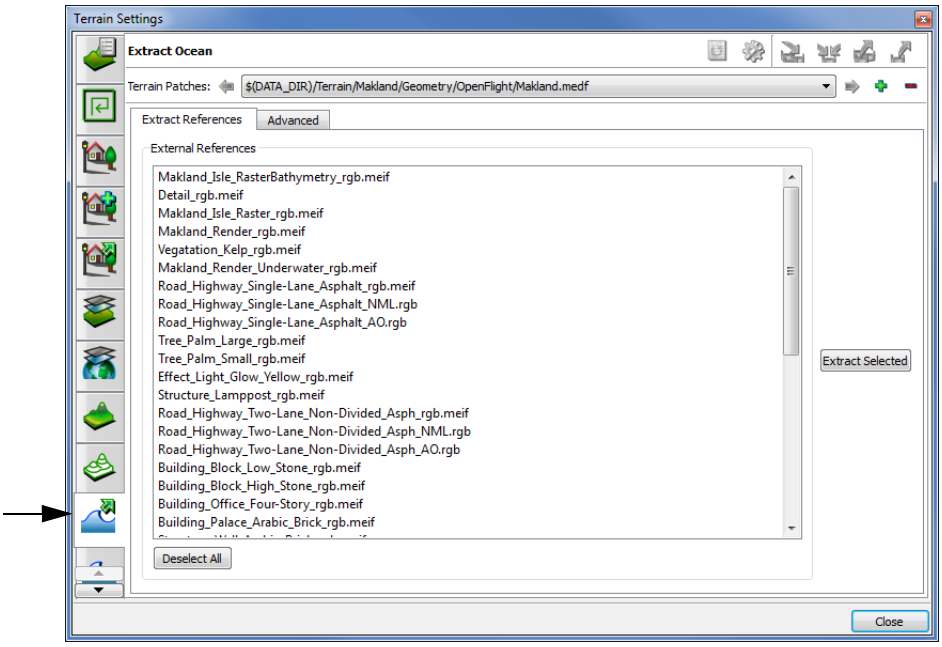


Figure 50-12. Terrain Settings dialog box, Extract Ocean page

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 50-13).
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 50-16.

50.5.2. 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 "Terrain Contents with dynamic ocean layer," on page 50-16.
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 50-13).

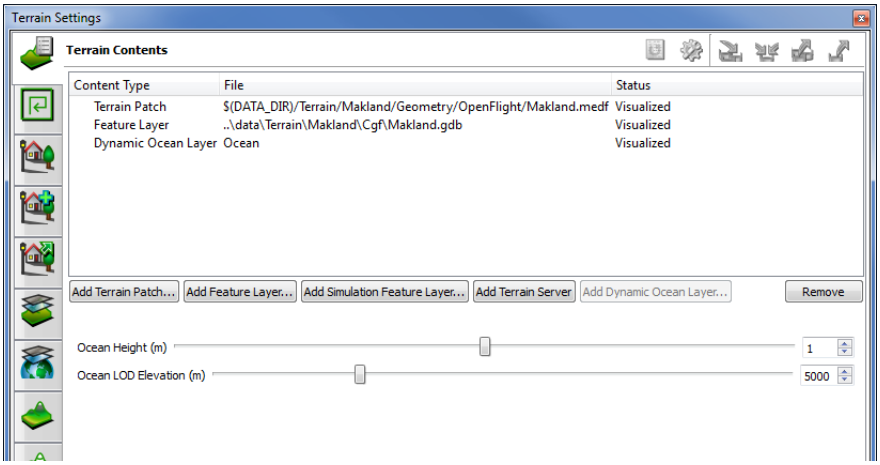


Figure 50-13. Terrain Contents with dynamic ocean layer

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 50-16.

50.5.3. 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 50-13).
3. Adjust the Ocean LOD Elevation slider or change the value in the box.

50.5.4. 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 50-13 shows a coastline at different levels of resolution.

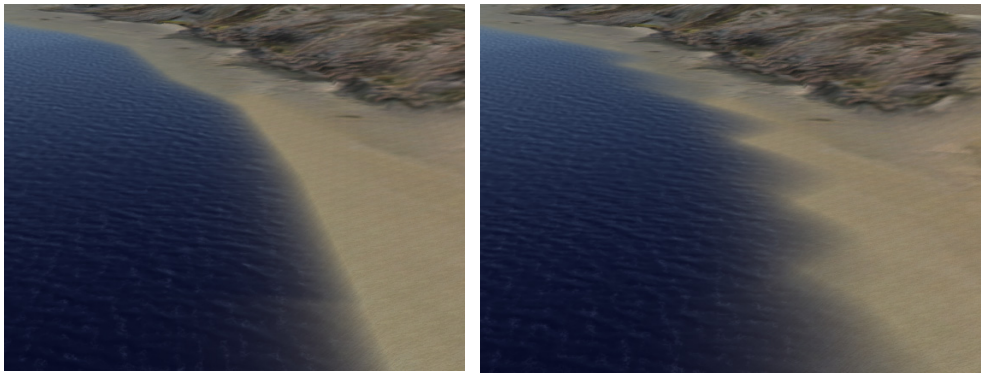


Figure 50-14. Ocean height map resolution



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 11-16.

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 Render Settings page (Figure 50-15).

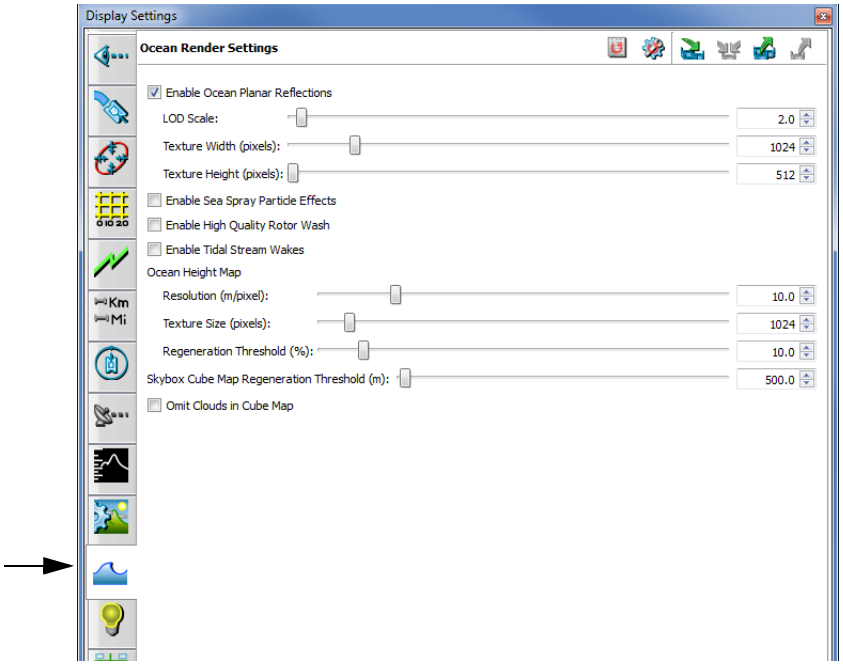


Figure 50-15. Ocean Render Settings

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 Render Settings page (Figure 50-15).
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 Render Settings page (Figure 50-15).
3. Adjust the Ocean Height Map Regeneration Threshold slider, or enter a value in the box.

50.6. 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 50-16).

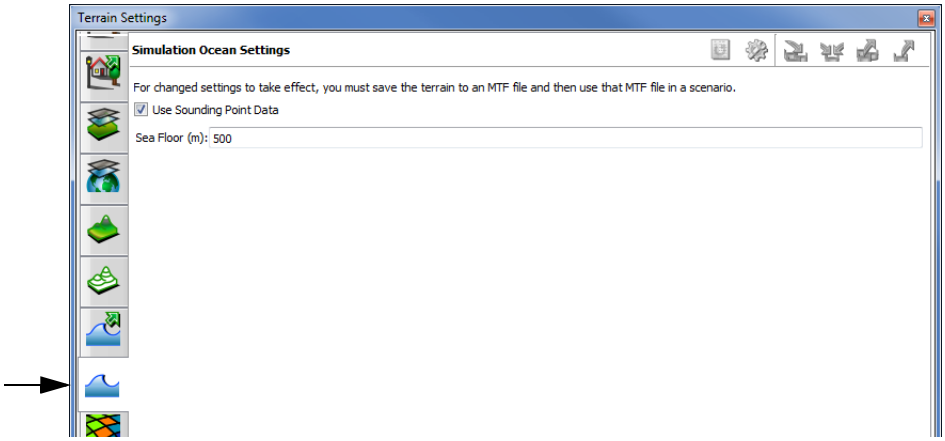


Figure 50-16. Simulation Ocean Settings page

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.

50.7. 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. 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. 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.

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*.

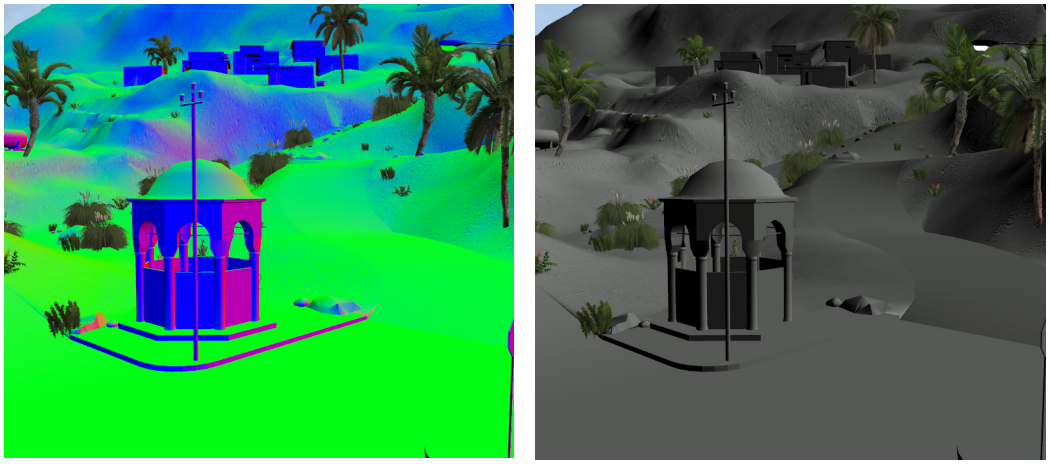


The map code must be uppercase on Linux.

To see the effect of effect maps, you must enable lighting. For details, please see Chapter 38, *Lighting Effects*.

50.7.1. 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 50-17 shows a view of VR-Village with two different shader debug settings.



Bumpmap Normals | Diffuse Light

Figure 50-17. Shader component visualization

To debug shaders:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 50-18).

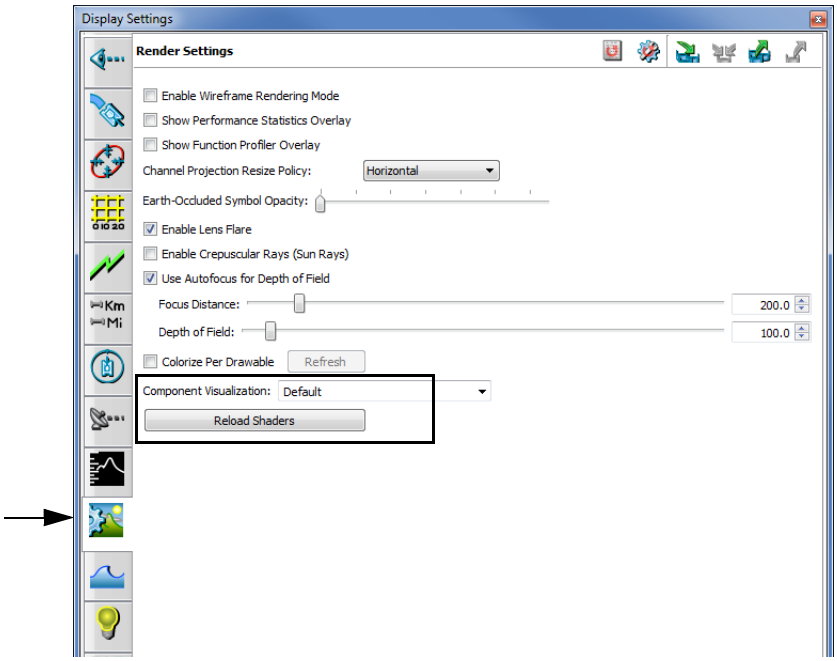


Figure 50-18. Render Settings page

3. Select an option from the Component Visualization list.

50.7.2. 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.

To dynamically reload shaders:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 50-18).
3. Click Reload Shaders.

50.8. 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 saved as part of the order of battle. They are a mechanism for placing objects into a scenario for visualization or simulation purposes.

50.8.1. 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 56-20.

To extract props from a terrain:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Extract Props page (Figure 50-19).

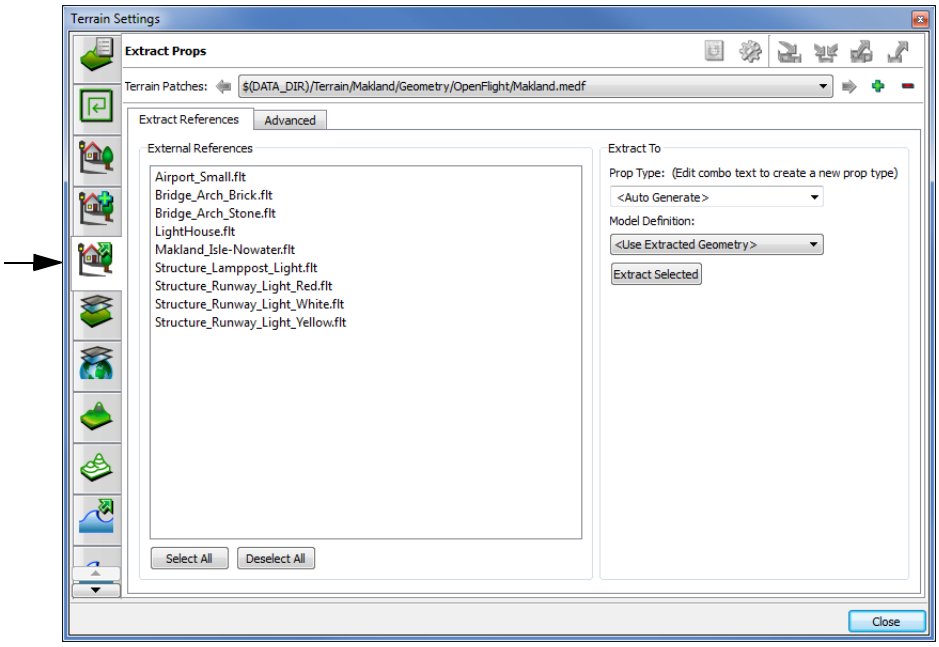


Figure 50-19. Extract Props page

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.

50.8.2. Adding Props from a Feature Layer

A feature layer source file contains information about point, line, 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.

Please see "Add Props," on page 56-10 for an example of adding props from a feature layer.

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 50-20).

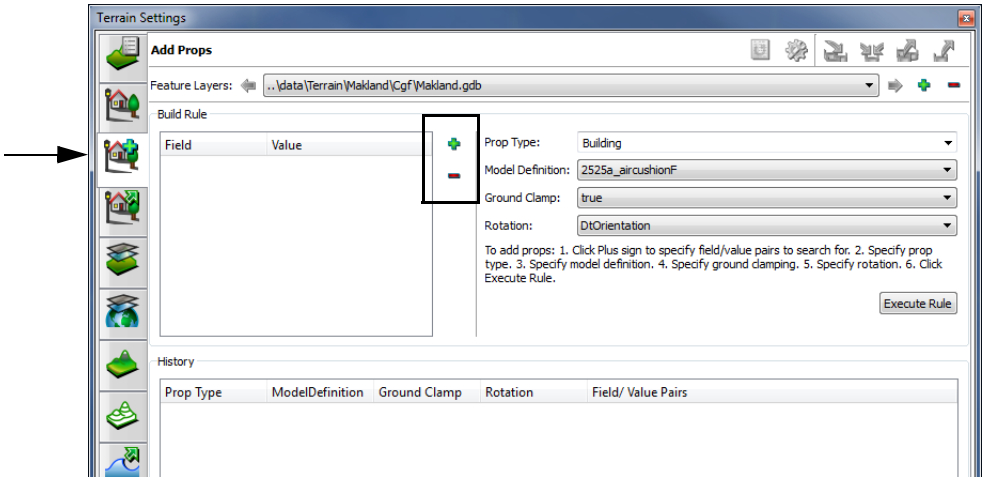


Figure 50-20. Terrain Settings, Add Props page

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 50-20). The Build Field/Value Pair dialog box opens (Figure 50-21).

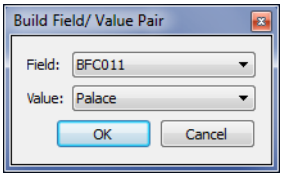


Figure 50-21. Build Field/Value Pair dialog box

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 50-22).

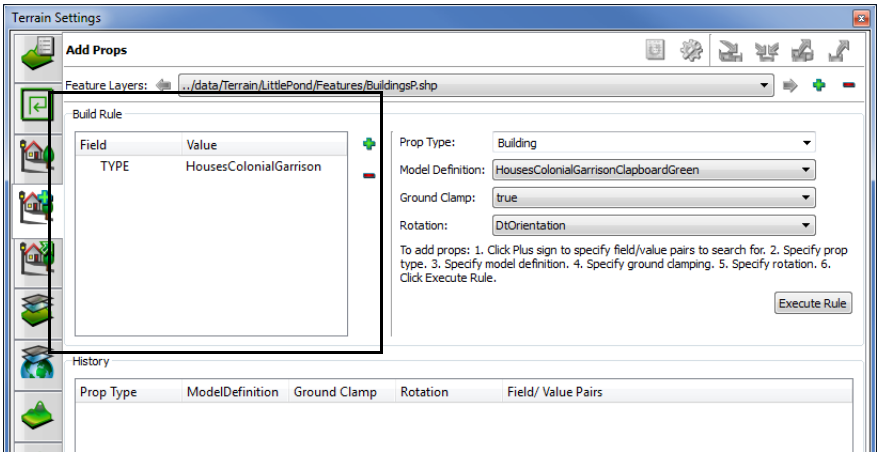


Figure 50-22. New Field/Value pair

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.

50.8.3. 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 50-23).

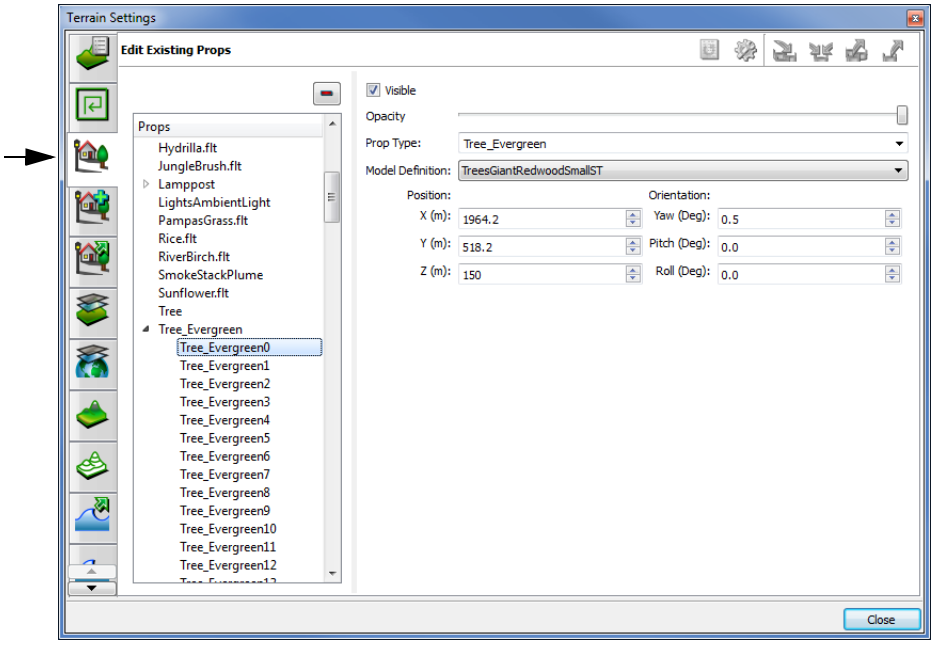


Figure 50-23. Edit Existing Props page

3. Expand the list to view the entries under a particular type of prop.

50.8.4. 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.

50.8.5. 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 50-24 illustrates a building at two levels of opacity.



100% 40%

Figure 50-24. 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 50-23).
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 popup menu, choose **Make Transparent**.

➤ To make a transparent prop fully opaque, right-click the prop and on the popup menu, choose **Make Opaque**.

50.8.6. 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 50-23).
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.

50.8.7. 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 50-23).
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.

50.8.8. 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 57, *Model and Element Definitions*. For an example of changing a prop's model definition, please see "Test the Model Mapping," on page 62-10.

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 50-23).
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.

50.9. 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 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 50-25).

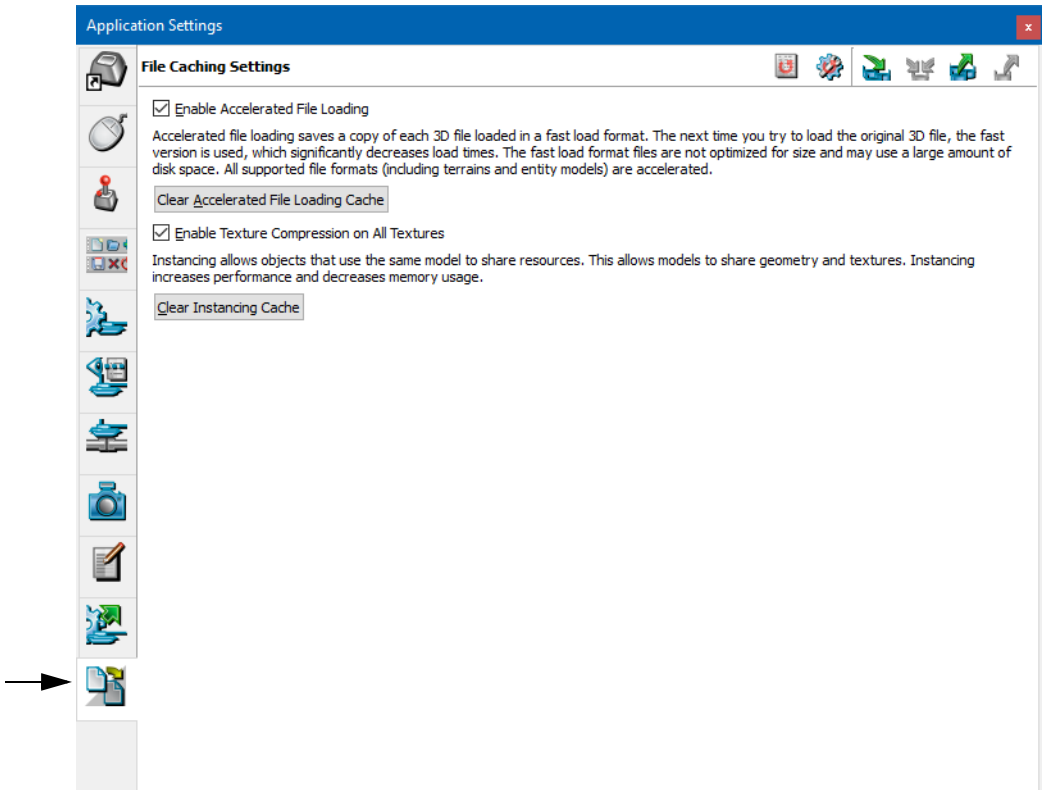


Figure 50-25. File Caching Settings page

3. Select or clear the Enable Accelerated File Loading check box.

50.9.1. Caching 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 an alternative caching location in the OSGEARTH_CACHE_PATH environment variable. (Do not use the `cache` element in the earth file.)

50.9.2. Caching osgEarth 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` lists information about the cache in an earth file.
- `--seed` seeds the cache for an earth file.
- `--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. For CDB terrains, this value must match the max_level attribute in the earth file you use to load the terrain. If the values do not match, the GUI and back-end may not correlate.

! 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` 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\HawaiiLocal-
  Server-mak.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*.

50.9.3. 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.

To enable or disable texture compression:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 50-25).
3. Select or clear the Enable Texture Compression On All Textures check box.

50.9.4. 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 50-25).
3. Click Clear Accelerated File Loading Cache.

50.10. 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 59-17.)
- When you add a terrain patch. (For details, please see "Adding Elevation Data (Terrain Patches) to a Terrain," on page 50-5.)
- As a global setting. (For details, please see "Flipping DDS Textures Globally," on page 50-35.)

50.10.1. 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 50-26).

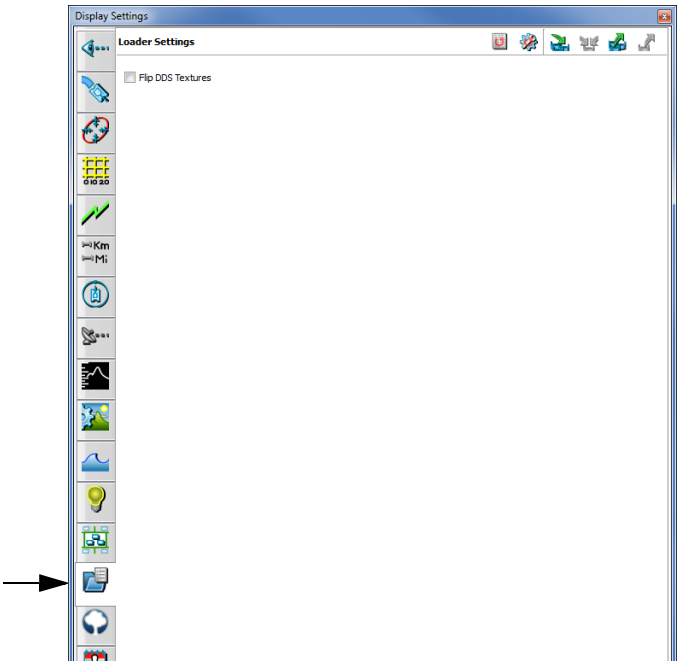


Figure 50-26. Loader Settings page

3. Select the Flip DDS Textures check box.
4. Click Close.

50.11. 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 the 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, the 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 50-1 lists the MÄK soil types to which you can map the OpenFlight texture files.

Table 50-1: MÄK 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 MÄK soil type.



The list of supported soil types is fixed. Therefore, you cannot add new soil types.



51. 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 51-2
Adding Terrain Server Connections 51-4
Connecting to Terrain Servers through a Proxy 51-5
Loading MetaFlight Terrains 51-5
Preprocessing Paged Terrains 51-5
Manually Paging-In Terrain 51-6
Displaying Information About Paged and Streaming Terrains 51-7
Displaying Information about Feature Paging 51-9
Displaying Additional Terrain Paging Information 51-10
Configuring Terrain Paging for the Back-End 51-12
Building Efficient MetaFlight Terrains for VR-Forces 51-12

51.1. Connecting to Terrain Servers

VR-Forces supports the osgEarth libraries for creating terrains from remote or local data sources (Figure 51-1). 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 51-1. 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 MÄK. 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.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 a Terrain with an Earth File," on page 56-14, 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 **New** → **Scene**. The current terrain closes.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Terrain Contents page (Figure 51-2).

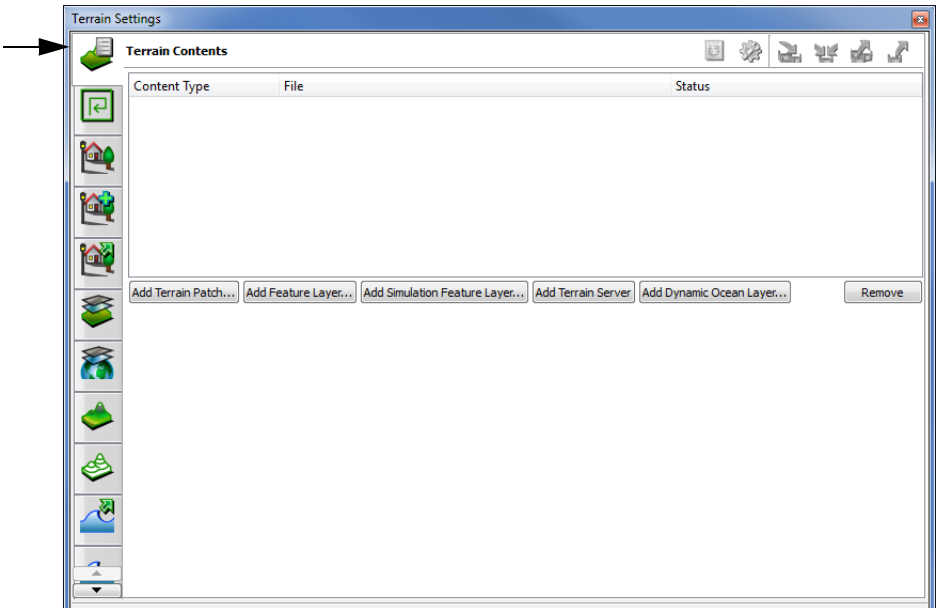


Figure 51-2. Terrain Contents

4. Click Add Terrain Server. The Add Terrain Server dialog box opens (Figure 51-3).

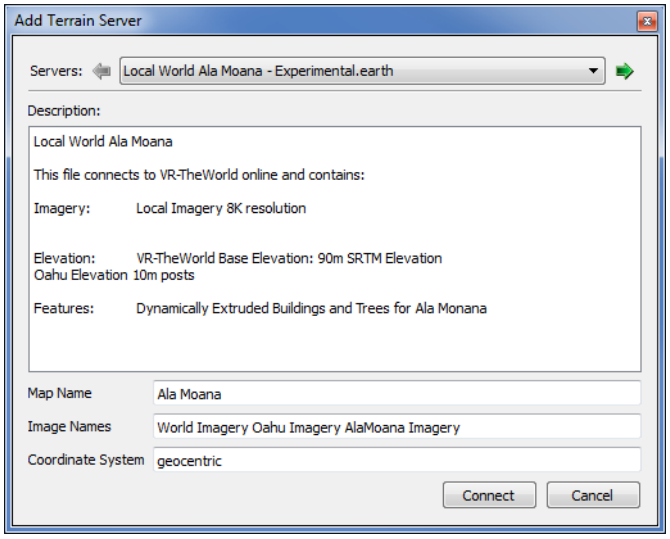


Figure 51-3. Add Terrain Server dialog box

5. Select a server from the Servers list. The dialog box displays information about the selected terrain server. It is not editable.
6. 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 50-5.

51.1.1. 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. It is beyond the scope of this manual to fully describe the configuration process. However, 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. (Chapter 52, *Editing Earth Files*, provides a brief introduction to earth file syntax.)

"Creating a Terrain with an Earth File," on page 56-14, is a tutorial that shows how to create a terrain server that serves local data.

51.1.2. 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.

51.2. 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 54, *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 50-31.



Geometry files referenced by a MetaFlight file must be in OpenFlight format, and not in Vega Prime's binary format (*.vsb*). MÄK's terrain libraries do not support *.vsb* files.

51.3. 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 59-23.

51.4. 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 Categories 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 16-3.

51.5. Displaying Information About Paged and Streaming Terrains

By default, when you enable the display of terrain paging information, VR-Forces displays the portions of the terrain that are paged into the back-end. In Figure 51-5, green spheres represent the area that is paged in. Red spheres represent area that is not paged in. In Figure 51-5, green squares show the tiles that have been paged in as a simulation object moves over the terrain. If you are debugging a terrain and need help using this feature, please contact support@mak.com.

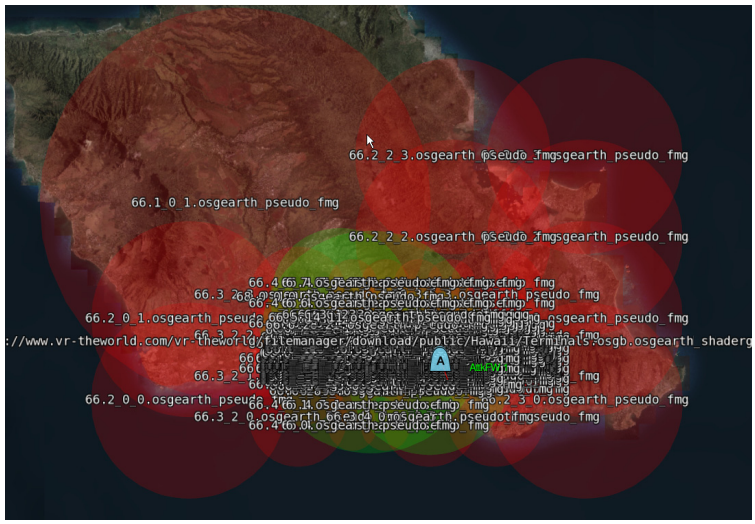


Figure 51-4. Paged-in terrain



Figure 51-5. Paged-in tiles

You can display different information by entering commands on the back-end console.

When you display terrain paging information, you can also display the features that are loaded in the back-end. You specify which features to draw using console commands, as described in "Displaying Additional Terrain Paging Information," on page 51-10. We recommend that you display features this way only in Plan View mode.

By default, the ability to display terrain paging information is disabled. To enable it, you must add the environment variable MAK_VRF_ENABLE_DEBUG_DRAWER and set it to 1. If you set this environment variable from the command line before you run VR-Forces (as opposed to setting it for the computer as a whole), you must set if for both the front-end and back-end.

! Enabling display of terrain paging information can introduce stability problems. It should only be enabled for debugging terrain.

➤ To enable or disable display of terrain paging information, choose **Settings** → **Terrain Paging Information** → **Terrain Paging Areas**.

51.5.1. Displaying Information about Feature Paging

Similar to the information you can display about paged terrain, you can display information about features as they are paged in. Figure 51-6 shows feature paging information from the inset area of the S57-NavalPathPlanning example scenario.

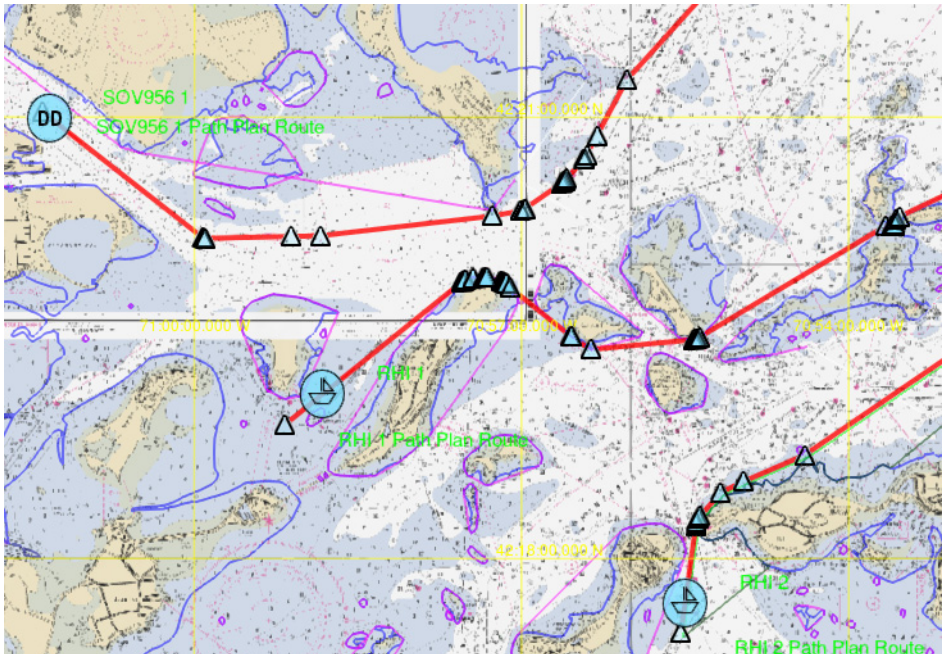


Figure 51-6. Feature information

By default, the ability to display feature paging information is disabled. To enable it, you must add the environment variable MAK_VRF_ENABLE_DEBUG_DRAWER and set it to 1. If you set this environment variable from the command line before you run VR-Forces (as opposed to setting it for the computer as a whole), you must set if for both the front-end and back-end.

! Enabling display of feature paging information can introduce stability problems. It should only be enabled for debugging terrain.

➤ To display feature paging information, choose **Settings** → **Features Paging Information** → *feature*. where *feature* is an option on the pull-right menu.

---

i The list of features to be viewed changes as features are paged in.

51.5.2. Displaying Additional Terrain Paging Information

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 51-1 lists the commands and their results.

Table 51-1: Terrain paging information commands

Command	Result
`features config`	Print queries and transforms from *featureconfig.txt*. For details about feature configuration, please see Chapter 55, *Configuring Feature Data*.
`features draw` *feature_set_index ...*	Draws the feature sets specified by the list of feature set indexes. Separate the indexes by spaces.
`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 drawmode` *mode*	Specifies a bitmask that controls what information to display, as follows: ◆ 0 — Turn off information display. ◆ 1 — Display polygons (Figure 51-7). ◆ 2 — Display collision object bounding boxes (paged terrains) (Figure 51-8). ◆ 4 — Display spheres (paged terrains). ◆ 8 — Tile extents (streaming terrains). ◆ 12 — Show spheres for paged terrain and tiles for streamed terrain (the default). ◆ 15 — Display all terrain paging information.
`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*.

Table 51-1: Terrain paging information commands

Command	Result
`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.

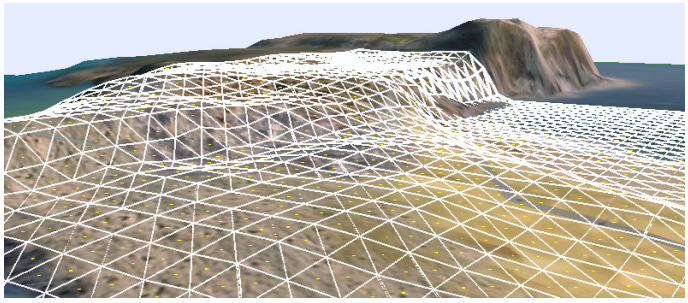


Figure 51-7. Polygon display

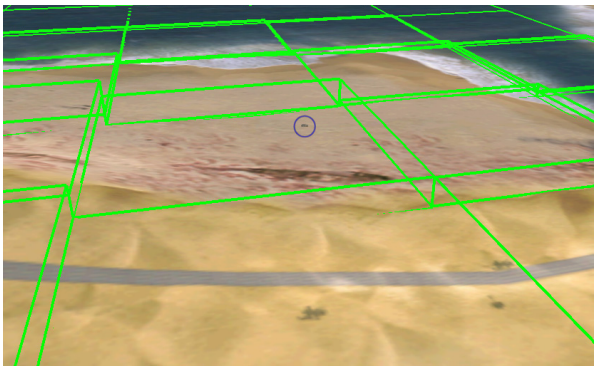


Figure 51-8. Collision objects display

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`**.

51.5.3. Configuring Terrain Paging for the Back-End

You can configure terrain paging for the back-end in *./appData/settings/vrfSim/terrainOptions.mtl*. Of particular note, you can configure the total number of triangles to be paged in at any one time. If the number of triangles exceeds this threshold, VR-Forces begins to page out triangles based on the priorities described in "Using Large Terrain Databases," on page 47-7. The parameters that you can configure are described in comments in the file. (To configure terrain paging in the front-end, use the procedures described in "Loading MetaFlight Terrains," on page 51-5.)

51.6. 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 following coordinate systems: Geodetic, Geocentric, UTM, or Flat Earth.
- 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 will reduce cull times. (For example, instead of having 10 individual buildings, externally reference one city block with all 10 buildings.)



52. Editing Earth Files

This chapter provides an introduction to the syntax of Earth files.

Earth Files 52-3
A Simple Earth File 52-3
Including External Files into Earth Files 52-5
Loading Multiple DTED Files 52-6
Processing Streamed Data 52-6
Adding Feature Layers to an Earth File 52-7
Configuring Point Features 52-9
Configuring Linear Features 52-10
Configuring Areal Features 52-11
Extruded Buildings 52-12
Using Cut-in Sites for High Resolution Insets 52-13
Using the Boundary Generation Tool 52-14
Loading CDB Databases 52-16
Preprocessing CDB Databases 52-16
Earth File Options 52-17
Differential Processing of Elements and Attributes 52-17
Differential Processing of model Elements 52-18
Streaming Buoys and Beacons 52-19
Specifying Model Definitions for Streamed Buoys 52-20
Modeling Buoys Using Texture Atlases 52-23
Configuring Streaming Beacons 52-24
Configuring Streamed Navigation Lights 52-25
Configuring Seasonal Buoys and Beacons 52-26
Streaming Daymarks 52-27

Streaming Topmarks 52-27
Generating Signal Sequences 52-28

52.1. Earth Files

An earth file is an XML format configuration file used to display 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.

52.2. A Simple Earth File

The *VR-TheWorldOnline.earth* file is an example of a simple earth file. It has one image layer and one elevation layer:

```
<!--
VR-TheWorld Online
This file connects to VR-TheWorld online and contains:
Imagery:VR-TheWorld Base Imagery: 15m SRTM 90 Imagery for the World
     Boston Inset Imagery
     Washington DC Imagery
     Austin, TX Imagery
     Afghanistan Imagery

Elevation:VR-TheWorld Base Elevation: 90m SRTM Elevation
-->
<map name="VR-TheWorld" type="geocentric" version="2">

<image name="VR-TheWorld Base Imagery" driver="tms"
 lod_blending="true" cacheid="VR-TheWorldBaseImagery">
    <url>http://vr-theworld.com/vr-theworld/tiles/1.0.0/22/</url>
</image>

<elevation name="VR-TheWorld Base Elevation" driver="tms"
 max_data_level=13 max_level=13 cacheid="VR-TheWorldBaseElevation">
    <url>http://vr-theworld.com/vr-theworld/tiles/1.0.0/34/</url>
</elevation>

<options>
  <lighting>true</lighting>

  <elevation_interpolation>triangulate</elevation_interpolation>
  <terrain>
    <min_tile_range_factor>3</min_tile_range_factor>
  </terrain>
  <cache type="filesystem">
    <path>../../appData/cache</path>
  </cache>
</options>
</map>
```

Table 52-1 describes the XML elements in the file and their attributes.

Table 52-1: 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.
options		Specifies options that affect the entire map. For more information, please see "Earth File Options," on page 52-17.

52.2.1. 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 an alternative include format. Use the syntax `{% include filename %}`. This syntax is used in some of the earth files shipped with VR-Forces. The following lines are from *VR-TheWorld Online - MAK Earth.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. Files included with the `xi:include` syntax do not have this restriction.

52.3. Loading Multiple DTED Files

You can load multiple DTED files from an earth file. To do so, in the `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.

52.4. Processing Streamed Data

VR-Forces has several "processors" that search through streamed-in data looking for things like SpeedTrees, 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 SpeedTrees, speedTreeManager, is mapped to Vegetation. In *VR-TheWorld Online - MAK Earth.earth*, SpeedTrees 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 speedTree-Manager.

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">
```

registrant_feature_list.csv file is fully commented and provides additional information about this scheme.

52.5. Adding Feature Layers to an Earth File

You can add the following types of features to a terrain:

- Point features, such as 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. Table 52-2 describes the `model` element, its attributes, and sub-elements.

Table 52-2: `model` element attributes

Element	Attribute	Description
model		Base element for specifying a feature layer.
	name	Name of the layer. This can be any text.
	driver	Driver to use for creating models.
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.
	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 (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.

Table 52-2: `model` element attributes

Element	Attribute	Description
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.
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 Elements and Attributes," on page 52-17.



When you load a file using the OGR driver, OGR decides what the layer names will be. For shapefiles it is always 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.

52.5.1. Configuring Point Features

Point features, such as trees, use the `model` element. The following code, from *VR-TheWorld Online - MAK Earth.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>
```

52.5.2. Configuring Linear Features

Linear features, such as roads, rivers, or railways, use the `model` element. The following code, from *VR-TheWorld Online - MAK Earth.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>
```

52.5.3. Configuring Areal Features

Areal features, such as forests, use the `model` element. The following code, from *VR-TheWorld Online - MAK Earth.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>
```

52.6. 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 *VR-TheWorld Online - MAK Earth.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 52-2.

52.7. 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 *VR-TheWorld Online - MAK Earth.mtf* has two terrain patches, *VR-TheWorld Online - MAK Earth.earth*, which provides the base terrain from VR-TheWorld Server, and *./data/Terrain/Hawaii/Geometry/OpenFlightAlaMoana/AlaMoana-ModifiedGeocentricProxy.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 52-3 describes the `mask` element and its sub-elements.

Table 52-3: `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.
	driver	The driver to use. OGR reads raw vector feature data from shapefiles and other formats.

Table 52-3: `mask` element

Element	Attribute	Description
geometry		Specifies the 3D boundaries of the site. You must know this information.
	geometry_url	The location of the file with the feature data.

52.7.1. 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 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 52-13.)
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 52-4 describes the options.

Table 52-4: osgEarth Boundary Generation Tool command-line options

Option	Description
`--convex_hull`	Calculate 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`	Show the result in a 3D window.

52.8. Loading CDB Databases

VR-Forces supports image and elevation data in CDB databases. CDB databases define a file structure. MÄK products load the data using an earth file. The earth file entries point to the files and specify the CDB driver.

Because MÄK 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.

The following code shows how to configure elevation and image data in an earth file:

```
<image name="cdb_imagery" driver="CDB">
  <cdb_root>C:\CDB\CampPendelton\CDB\CDB_CMPP_1_0</cdb_root>
</image>

<elevation name="cdb_elevation" driver="CDB">
  <cdb_root>C:\CDB\CampPendelton\CDB\CDB_CMPP_1_0</cdb_root>
</elevation>
```

Be sure to set <cdb_root> to the top level of your CDB database.

! In almost all cases, initial loading of a CDB terrain will result in unacceptable performance. Therefore, you should always preprocess the terrain, as described in "Preprocessing CDB Databases," on page 52-16.

52.8.1. Preprocessing CDB Databases

VR-Forces treats CDB databases as a streaming terrain source. When it loads the data, it organizes and tiles the imagery in a way that promotes long term optimal performance. However, this initial processing means CDB loading can be very slow and halting. Often, this delay is not acceptable in a run-time environment. To optimize the process and eliminate run-time delays you should preprocess the terrain.

VR-Forces includes a tool, *./bin64/osgearth_cache*, that you can use to cache terrain data offline. For details about how to use osgearth_cache, please see "Caching osgEarth Terrain Data Offline," on page 50-32.

When you preprocess the terrain, the cached data is written to *./appData/settings/cache*. There will be a directory for each CDB element (one for each elevation layer and one for each imagery layer). You must copy these directories to*./appData/settings/cache/vrfsim*.

52.9. Earth File Options

An earth file can have `option` elements that affect the entire map. Table 52-5 describes the option elements.

Table 52-5: `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.

52.10. Differential Processing of Elements and Attributes



VR-Vantage shares earth files with VR-Forces. The differential processing described in this section would not be used by VR-Vantage. It ignores vrfsim: syntax. This section is included to help you understand why the earth files provided with VR-Vantage are written this way.

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>
```

52.10.1. 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 *VR-TheWorld Online - MAK Earth.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.

52.11. 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, CardinalBuoys, and SpecialPurpose-Buoys.

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 currently contains 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>
```

52.11.1. 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.

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,Stripe
  d,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.

52.11.2. 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 52-20 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 52-27 and "Streaming Topmarks," on page 52-27.) 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 alphanumeric 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*.

52.11.3. 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
```

52.11.4. 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. Again, style names can be applied and *light_layer_list.csv* controls which styles will be read. *VR-TheWorld Online - Massachusetts S-57.earth* uses a single style name for all lights, **NavigationLights**. An example light model block is shown below:

```
<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 will be 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()
```

52.11.5. 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] ;
```

52.11.6. Streaming Daymarks

VR-Forces supports streamed daymarks for beacons. A limited number of daymark models and textures are provided as examples. *VR-TheWorld Online - Massachusetts S-57.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").

52.11.7. 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.

52.11.8. 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 straight-forward 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.



53. Generating Navigation Data

This chapter explains how to generate navigation data for use with VR-Forces.

Introduction 53-2
Creating Navigation Areas 53-3
Editing a Navigation Area 53-4
Reverting Edits to Navigation Area 53-5
Displaying Navigation Areas 53-5
Removing a Navigation Area 53-5
Generating Navigation Data 53-5
Navigation Data Profiles 53-6
Generating Navigation Data for a Navigation Area 53-6
Using Newly Generated Navigation Data 53-8
Generating Navigation Data for Entities 53-8
Importing Navigation Areas and Navigation Data 53-10
Displaying Navigation Data 53-11
Opening Navigation Lab from an Information Dialog Box 53-13
Showing Simulation Object Movement in Navigation Lab 53-14

53.1. 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 generate navigation data 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.

Navigation data is generated 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.

53.2. Creating 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 *./vrforces4.4.1/appData/settings/vrfSim/navigationProfiles.mtl*.)

You can create multiple navigation areas on a terrain.

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 53-1).

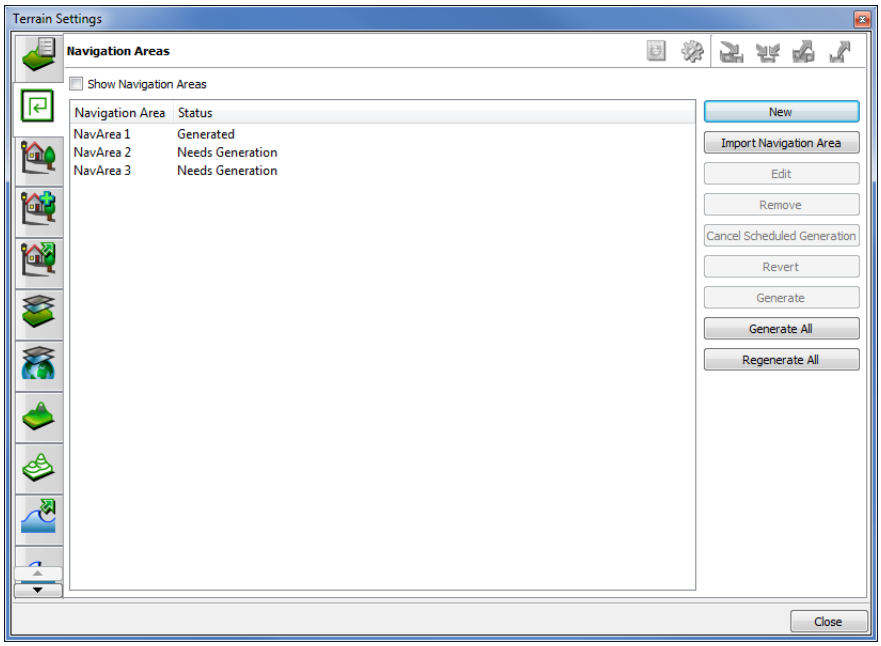


Figure 53-1. Navigation Areas page

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 53-2).

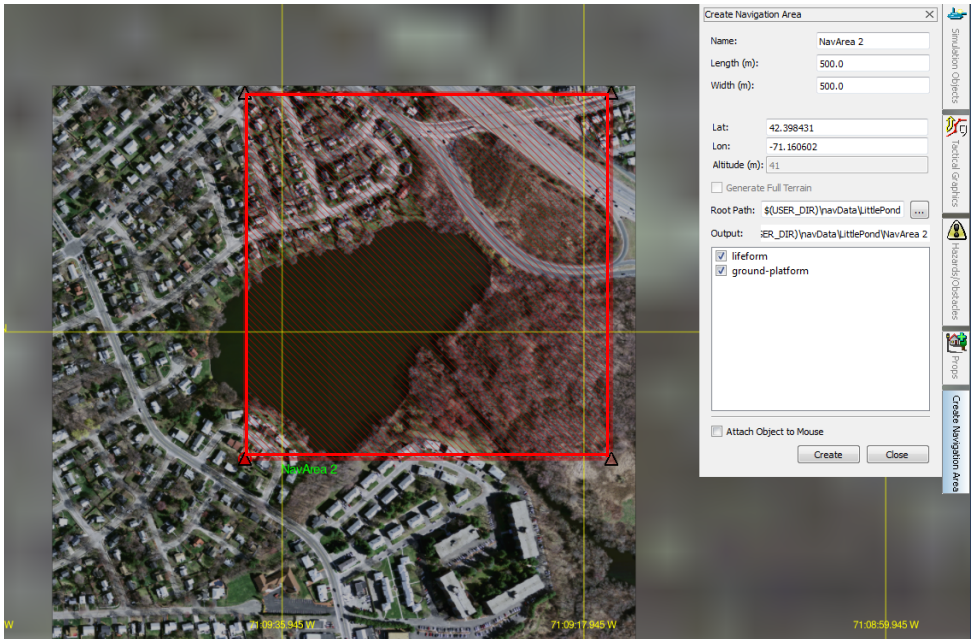


Figure 53-2. New navigation area

6. In the Create Navigation Area dialog box, edit any of the properties of the navigation area, including:
   - Name. (You cannot change the name of a navigation area after you create it.)
   - Length and Width. Edit these values to change the size 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.
   - The type of navigation data to generate – for lifeform, ground platform, or both.
7. Click Create.

53.2.1. 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 53-1).
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.

53.2.2. Reverting Edits to 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 53-1).
3. Select the navigation area you want to revert.
4. Click Revert.

53.2.3. Displaying Navigation Areas

➤ To display navigation areas, choose Settings → Navigation Areas, or click the Navigation Area button ( ) on the Display Settings toolbar.

53.2.4. 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 53-1).
3. Select the navigation area you want to remove.
4. Click Remove.

53.3. 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.

53.3.1. Navigation Data Profiles

VR-Forces generates navigation data based on parameters in a profile. VR-Forces includes profiles for lifeforms and ground entities. They provides default values for generating navigation data that should work for most terrains. If you want to, 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.

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")
)
```

53.3.2. Generating Navigation Data for a Navigation Area

The Navigation Areas page (Figure 53-1) 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 should 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 53-1).
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 53-3). 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.

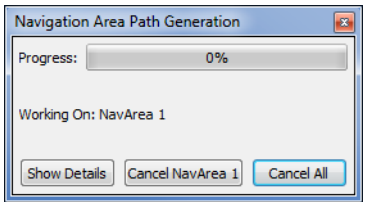


Figure 53-3. Navigation Area Path Generation dialog box

8. Click OK.
9. Click Finish.
10. Save the terrain. (**File** → **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.

53.3.3. 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.

53.4. 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 63-3.
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. In the attribute panel, select Use Detailed Geometry in Simulation Engine (Figure 53-4).

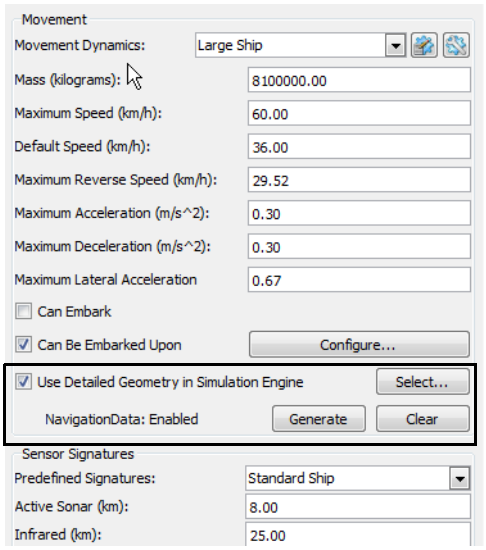


Figure 53-4. Object geometry

5. 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 an Entity," on page 64-13.)
6. Click Generate. The Generating Navigation Data dialog box opens.
7. Specify the platform types for which you want to generate navigation data.
8. Click Generate. VR-Forces generates navigation data (Figure 53-5). When the process finishes, the Cancel button changes to read Complete.

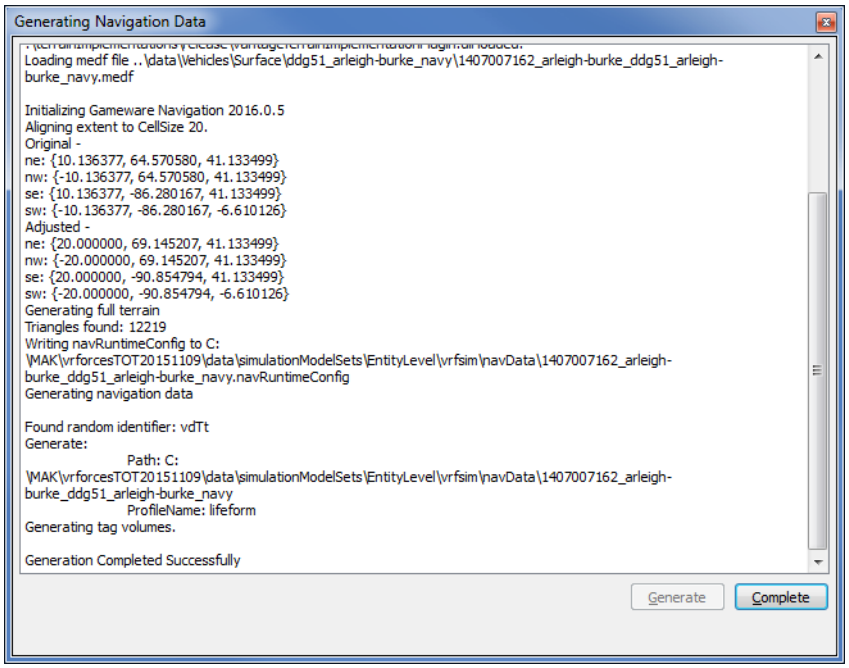


Figure 53-5. Generating Navigation Data dialog box

9. Click Complete.

53.5. 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 file for the Ala Moana navigation area provided with VR-Forces is *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 53-1).
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.

53.6. 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 53-6 displays the navigation data for VR-Village.

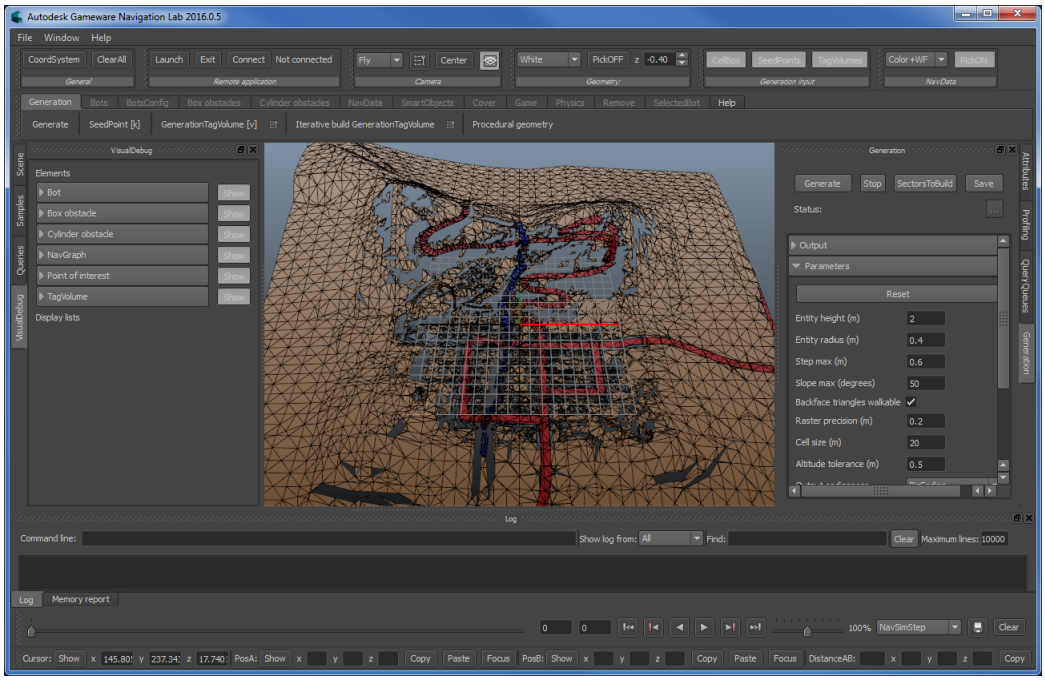


Figure 53-6. Navigation data display



Autodesk® Gameware Navigation Lab is not supported on Linux.

To display navigation data by opening it in Navigation Lab:

1. On the Windows 7 Start menu, choose **Programs** → **MAK Technologies** → **VR-Forces 4.4.1** → **Tools** → **Navigation Lab**. Navigation Lab opens.

   On Windows 10, Start menu, choose **All apps** → **MAK Technologies** → **Navigation Lab**.

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.

53.6.1. 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 53-7).

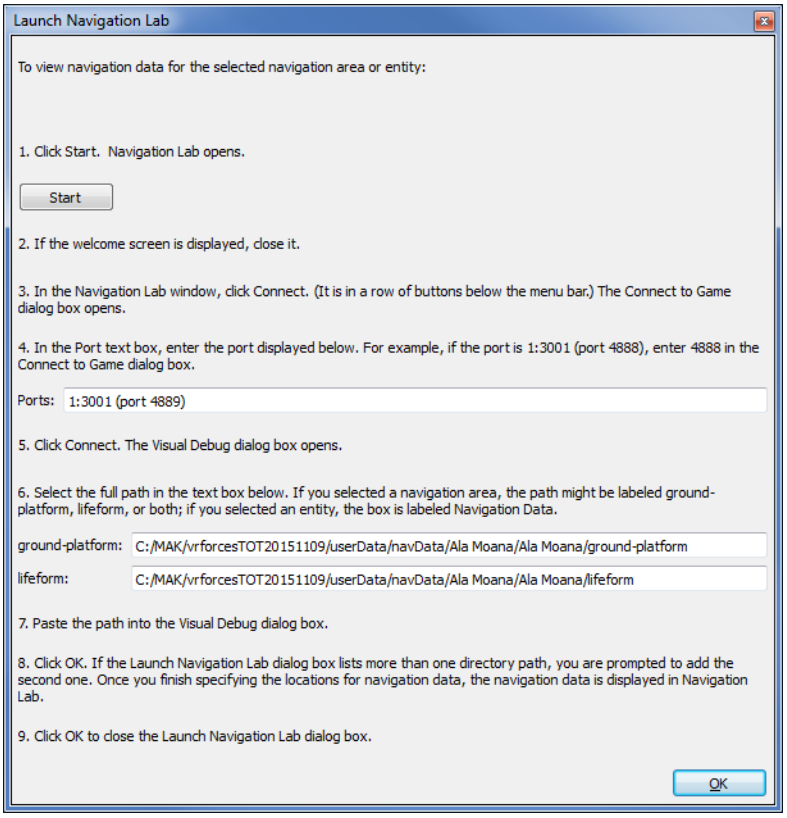


Figure 53-7. Launch Navigation Lab dialog box

3. Follow the directions on the dialog box.

53.6.2. 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 53-13.
2. Run the scenario.
3. In Navigation Lab, look for volumetric objects that represent the simulation objects (Figure 53-8).

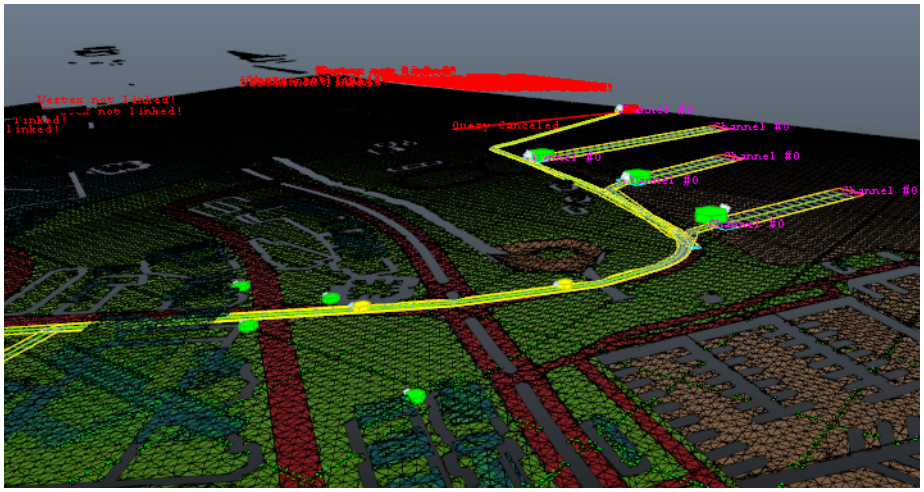


Figure 53-8. Simulation objects represented in Navigation Lab



54. Processing MetaFlight Files

To display MetaFlight files in VR-Forces, you must first process them. This chapter explains how to do so.

Processing MetaFlight Files for Use in VR-Forces 54-2
Splitting Datasets into Individual MetaFlight Files 54-2
Converting Virtual Texture Datasets to Tiled Textures 54-3
The mftTool 54-3
Using the mftTool to Process MetaFlight Files 54-5
Convert Geometry Grid Datasets with Tiled Textures 54-6
Convert Source Data into MEDF Format 54-6
Create an MTF File for the MetaFlight Terrain 54-7

54.1. 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 MÄK 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*). MÄK's terrain libraries do not support VSB files.

54.2. Splitting Datasets into Individual MetaFlight Files

MÄK 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).

54.3. 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.

54.3.1. 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 54-1 describes the command-line options.

Table 54-1: 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 54-1: 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 54-1: 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.

54.3.2. 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-tex-
  ture\ds200-vt.mft -r C:\yourTerrain\textures -r C:\your-
  Terrain\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.

54.3.3. 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 59-23.

To only publish a new MetaFlight for the cultural dataset, do the following:

```
mftTool C:\yourTerrain\flt\
   ds7_culture_db5_building_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.

54.4. 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
```

54.5. 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.



55. Configuring Feature Data

This chapter explains how to update the feature data configuration file so that the simulation engine can understand feature data.

Introduction 55-2
Specifying Different Feature Names for the Front-End and Back-end 55-2
Named Queries 55-3
Feature Configuration Attributes 55-4
Attribute Transformations 55-4
- Transformations as Aliases 55-4
- Transformations as Derived Attributes 55-5
- Transformations as New Attributes 55-5
- Writing Multiple Transforms to Account for Possible Failures 55-5

Source-Specific Configuration 55-5
- Loading Source-Specific Configuration Files 55-6
- Redefining Queries 55-6

Extracting Feature Data from GDB to Shapefiles 55-7
Converting S-57 Feature Data to Shapefiles 55-7
Configuring Aggregate-Level Movement Restrictions 55-9
Dynamic Features 55-10

55.1. 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.

55.2. Specifying Different Feature Names for the Front-End and Back-end

As described in "Adding Feature Layers to an Earth File," on page 52-7, 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 52-17.

55.3. 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 simulation model set configuration, or in the VR-Forces code. Do not change the names of these queries.

55.4. 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 55-3, 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.

55.5. 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.

55.5.1. 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. Attribute transformations can be used 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.

55.5.2. Transformations as Derived Attributes

Transformations can be used 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`.

55.5.3. Transformations as New Attributes

Transformations can be used 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.

55.5.4. 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.

55.6. 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.

55.6.1. 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.

55.6.2. 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.

55.7. 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 50-11.) If you have GDB terrains that contain feature data, you can extract that data to shapefiles with the vectorNetworkToShape-Files 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.

55.8. 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).

55.9. 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)
  )
)
```

55.10. 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"}
```



56. Terrain Tutorials

Although the individual procedures for managing terrain are fairly straightforward, understanding how they fit together to help you achieve your visualization goals is not always obvious. This chapter contains tutorials that demonstrate how to combine various procedures to create terrains and how to create a local terrain server.

Composing a Terrain through the GUI 56-2
- Add Elevation Data (Add a Terrain Patch) 56-3
- Add Imagery 56-5
- Add a Feature Layer 56-9
- Add Props 56-10

Creating a Terrain with an Earth File 56-14
- Add a Map Element 56-15
- Add Elevation Data to the Earth File 56-15
- Add Imagery to the Earth File 56-15
- Add Feature Data to the Earth File 56-16
- Add Options to the Earth File 56-18
- Save the Earth File 56-19
- Load the Earth File and Save is as an MTF File 56-19

Extracting Props from a Terrain 56-20
- Create a New Terrain and Add a Terrain Patch 56-21
- Extract the Props 56-21
- Change the Model Definition for a Prop 56-24

56.1. Composing a Terrain through the GUI

This tutorial shows how to compose a terrain by combining elevation data, imagery, and feature data in the VR-Forces GUI. It recreates a portion of the LittlePond terrain that is included with VR-Forces. Little Pond is a small detailed area in a suburb of Boston, Massachusetts, U.S.A.



This tutorial shows you how to add props to the LittlePond elevation and image data. However, the tutorial does not ask you to add every type of prop that the complete LittlePond terrain contains. Therefore, at the completion of the tutorial, your terrain will not look exactly like the LittlePond terrain provided with VR-Forces. If you want to completely recreate the LittlePond terrain, you will have to continue to add props on your own.

The basic steps for composing a terrain are:

1. Add elevation data (a terrain patch) ("Add Elevation Data (Add a Terrain Patch)," on page 56-3).
2. Add imagery ("Add Imagery," on page 56-5).
3. Add a feature layer ("Add a Feature Layer," on page 56-9).
4. Add props ("Add Props," on page 56-10).

56.1.1. Add Elevation Data (Add a Terrain Patch)

The first step in building a terrain is to add elevation data. Elevation data is added using terrain patches. The terrain patch in this tutorial is DTED data that covers about 1° east of Boston, Massachusetts, U.S.A.

To add the terrain patch:



1. If you are joined to a session, choose **File** → **Resign from Session**.
2. Choose **Observer** → **Set Observer Mode** → **Stealth**.
3. Choose **File** → **New Terrain**. This results in an empty window (Figure 56-1).



Figure 56-1. No terrain loaded

4. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
5. Select the Terrain Contents page.
6. Click Add Terrain Patch. The Add Terrain Patch dialog box opens and an Open dialog box opens on top of it.
7. In the Open dialog box, navigate to *./data/Terrain/LittlePond/elevation*.
8. Select *w072n042.dt1*.
9. Click Open. Information about this terrain patch is displayed in the Add Terrain Patch dialog box (Figure 56-2).

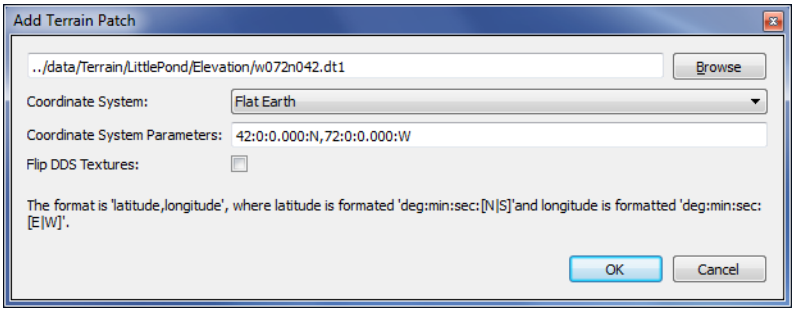


Figure 56-2. Add Terrain Patch

10. Click OK. The terrain is added to the scene (Figure 56-3).

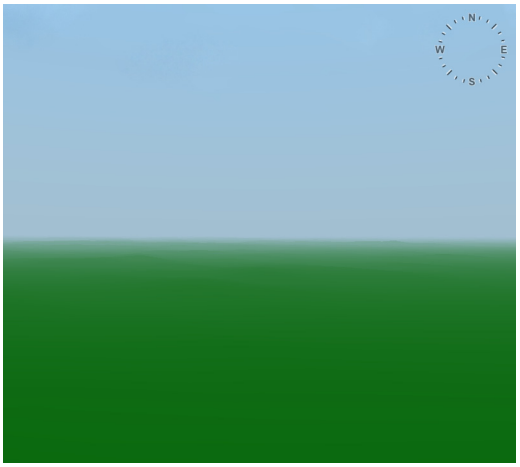


Figure 56-3. Terrain patch added

56.1.2. Add Imagery

The terrain patch you added in the first step of this tutorial is just elevation data. It has no texture and no features. You can get some sense of its contours by enabling elevation coloring (Figure 56-4). However, this still does not provide the rich 3D environment you expect to see in a product like VR-Forces. Now we will add image data.

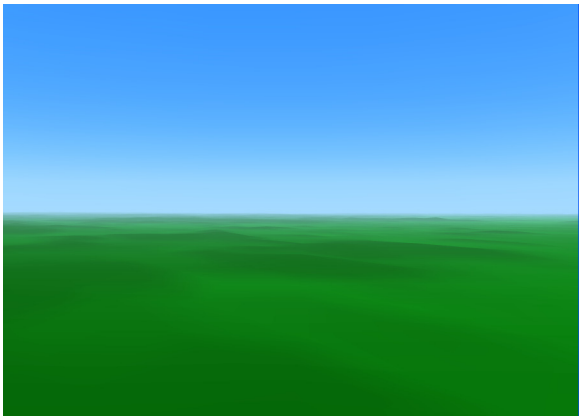


Figure 56-4. Elevation coloring

To add image data:



1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Raster Maps page (Figure 56-5).

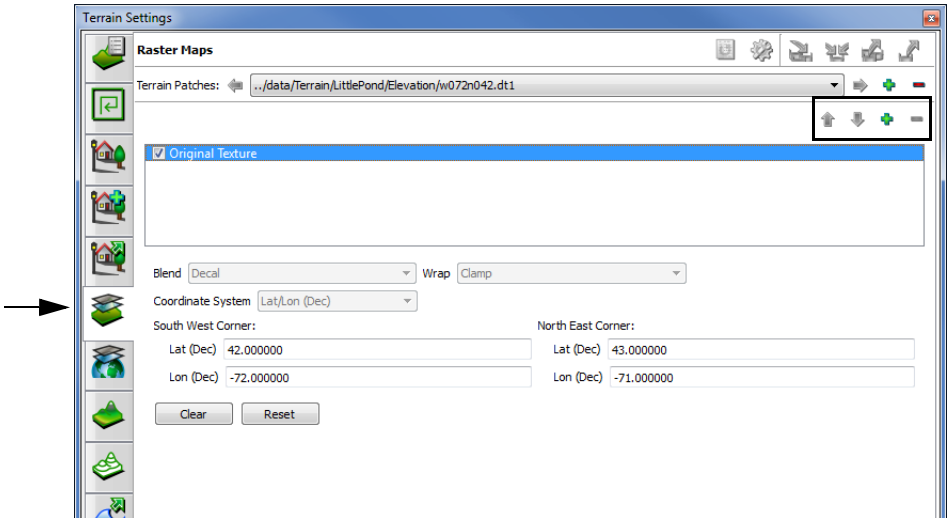


Figure 56-5. Raster Maps page

3. Clear the Original Texture check box. (If you leave this selected, the terrain will be green.)

4. Click the Add button (+) that is at the upper right of the list window (Figure 56-5). A line labeled Image 1 is added to the list in the window (Figure 56-6).

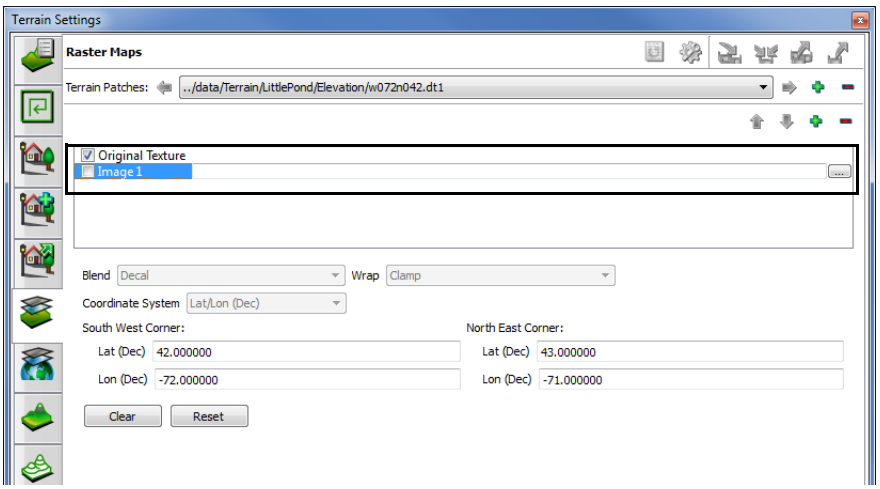


Figure 56-6. New image

5. Click the Browse button at the right end of the image entry. The Image File dialog box opens.
6. In the Image File dialog box, navigate to *./data/Terrain/LittlePond/imagery*.
7. Select *W72N42.tif*.
8. Click Open. The image is listed in the Raster Maps window. To view it, select the check box at the left end of the entry. This image adds low resolution imagery over the entire terrain (Figure 56-7).

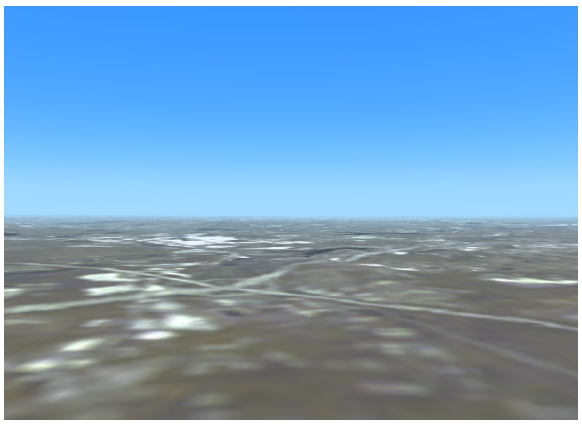


Figure 56-7. Low resolution image

9. Add another image. This time, select *LittlePond.tif*.

10. Select the check box for the *LittlePond.tif* image (Figure 56-8). A high resolution image is added to the Little Pond Area of the terrain (Figure 56-9). (The high resolution image might be hard to find. The next section explains how to find it quickly.)

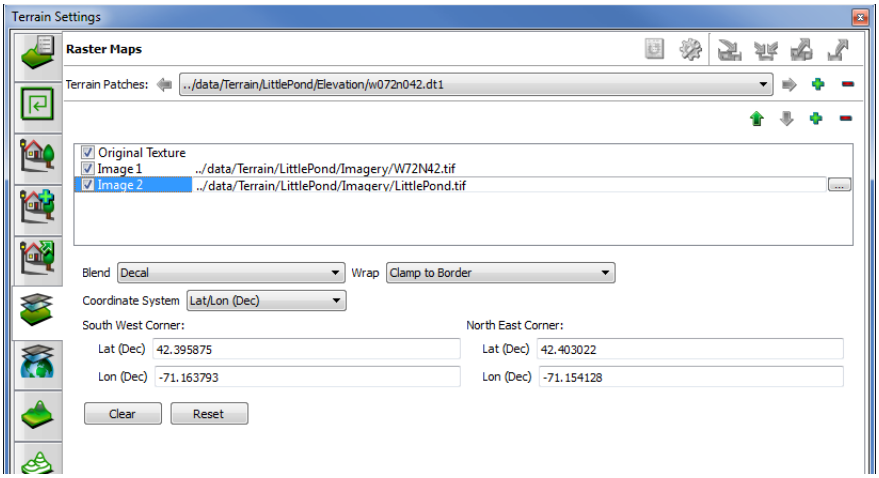


Figure 56-8. Raster Maps page with both images added

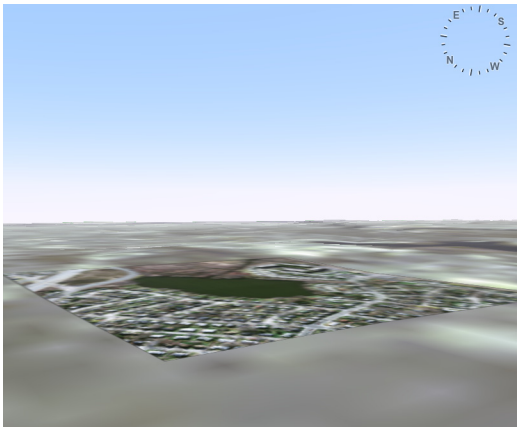


Figure 56-9. Little Pond image

Finding the High Resolution Image

It is not easy to find the Little Pond image unless you know where to look. To make things easier, we have saved an observer view of the Little Pond image. It will be the starting point for the rest of the tutorial.

To view the Little Pond image:

1. If the Observer Views Panel is not already displayed, choose **View** → **Observer Views Panel**.
2. On the Observer Views Panel, click the Load View button ( ). The Load and Set Observers dialog box opens.
3. Select *LittlePondTutorial.osrx*.
4. Click Open. The view should jump to an image similar to Figure 56-9.



As an alternative to this procedure, you could hide the low resolution image by clearing its selection on the Raster Maps page. Then you could scroll through the terrain until you find the high resolution image.

56.1.3. Add a Feature Layer

The image of Little Pond shows lots of buildings and greenery, but these are just images, not 3D objects. Now we want to add props to the terrain. To do that, we need to add a feature layer that has objects that we can extract.

To add the feature layer:



1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Add Props page (Figure 56-10).

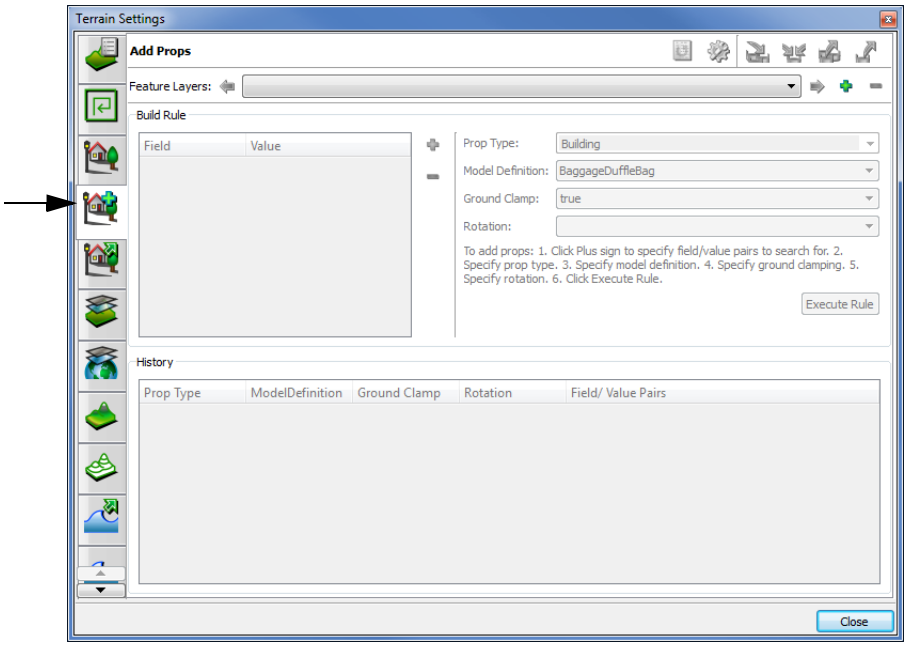


Figure 56-10. Add Props page

3. Click the Add button (+) next to the Feature Layers list. The Select Source File for Feature Layer dialog box opens.
4. Navigate to *./data/Terrain/LittlePond/Features*.
5. Select *BuildingsP.shp*.
6. Click Open. The layer is added to the Add Props page.

56.1.4. Add Props

When you add a prop, VR-Forces extracts data from a feature layer and maps it to a model definition. You have to tell VR-Forces what to extract and what to map it to. You do this by building a rule.

To build a rule and extract feature data:

1. In the Terrain Settings dialog box, on the Add Props page, in the Build Rule group box, click the Add button (+) (Figure 56-11). The Build Field/Value Pair dialog box opens.

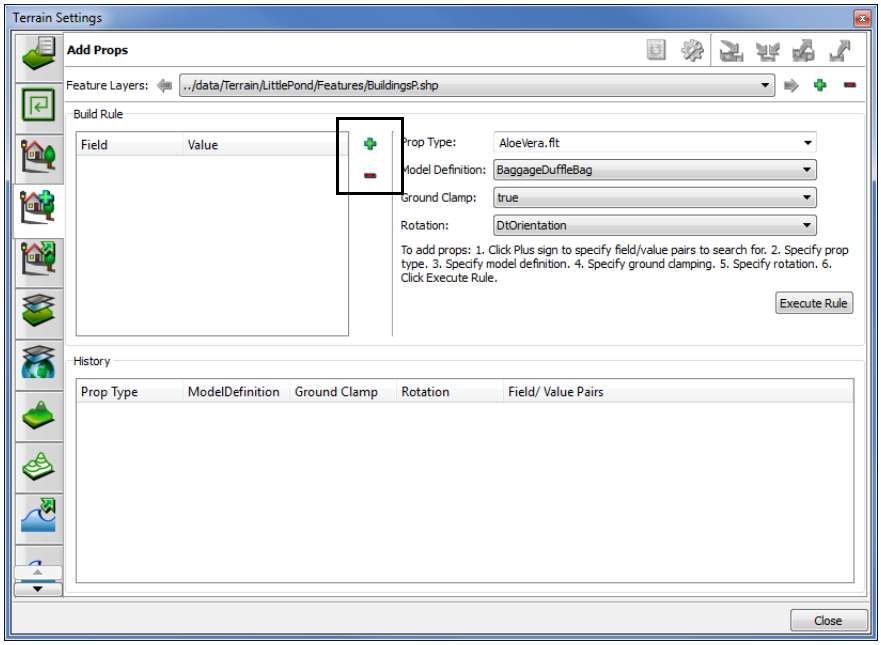


Figure 56-11. Add build rule

2. In the Field list, select TYPE.
3. In the Value list, select HousesColonialGarrison (Figure 56-12).

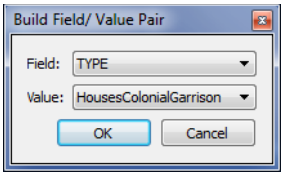


Figure 56-12. Build Field/Value Pair dialog box

4. Click OK.
5. On the Add Props page, in the Prop Type list, select the prop type for the rule. In this case, select Building.

6. In the Model Definition list, select HousesColonialGarrisonClapboardGreen.
7. In the Ground Clamp box, specify whether or not you want the prop to be ground-clamped.
8. In the Rotation list, select the orientation of your model. By default, if the GDB or Shapefile has a field called Rotation or DtOrientation, it is automatically selected. The completed rule is illustrated in Figure 56-13.

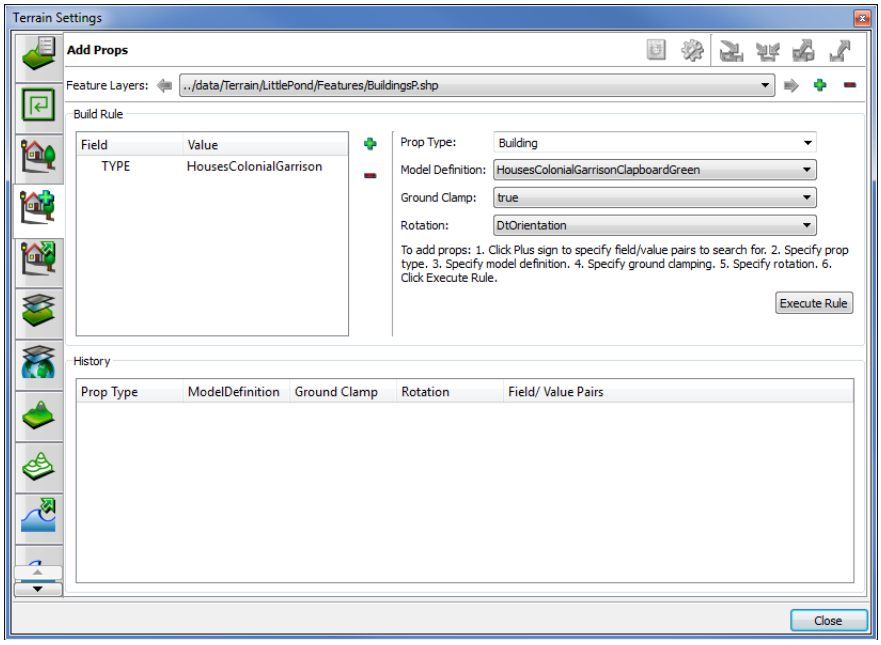


Figure 56-13. Completed rule

9. Click Execute Rule. The rule is executed. The Add Props page is updated (Figure 56-14) and houses are added to the terrain. Zoom in for a better look at them (Figure 56-15). (You might need to close the Terrain Settings dialog box or move it out of the way to see the terrain.)

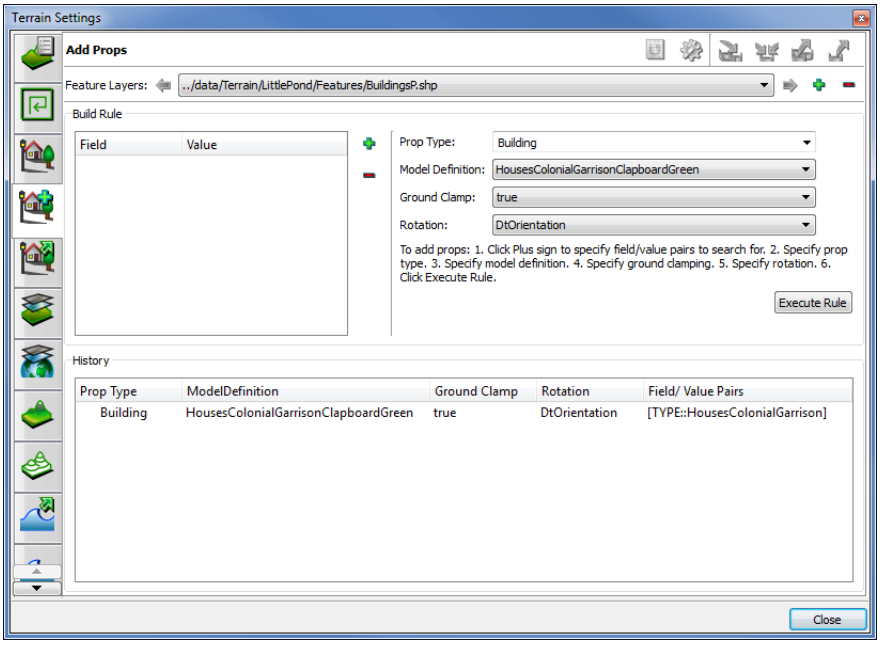


Figure 56-14. Build rule implemented

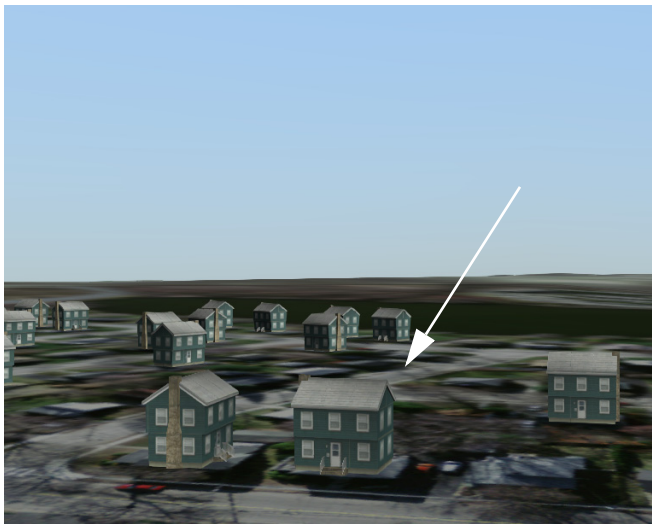


Figure 56-15. Props added to terrain

If you want to, you can add more buildings by creating rules for some of the other building types available.

Add Vegetation

We will continue the tutorial by adding another feature type.

To add vegetation to the terrain:

1. If the Terrain Settings dialog box is not open, choose **Settings** → **Terrain**.
2. Select the Add Props page.
3. Click the Add button (+) next to the Feature Layers list. The Select Source File for Feature Layer dialog box opens.
4. Select *./data/Terrain/LittlePond/features/VegetationP.shp*.
5. Click Open. A new feature layer is added to the list.
6. In the Build Rule group box, click the Add button (+). The Build Field/Value Pair dialog box opens.
7. In the field list, select VEG.
8. In the Value list, select TreesBigLeafMaple.
9. Click OK.
10. In the Prop Type list, select Tree.
11. In the Model Definition list, select TreesBigLeafMapleST. The build rule should look like Figure 56-16.

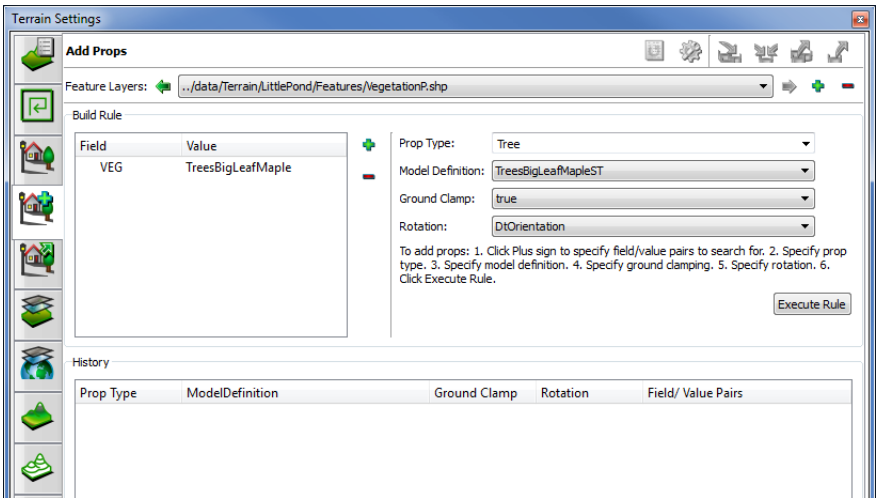


Figure 56-16. Rule for big leaf maples

12. Click Execute Rule. The page is updated and trees are added to the terrain (Figure 56-17).

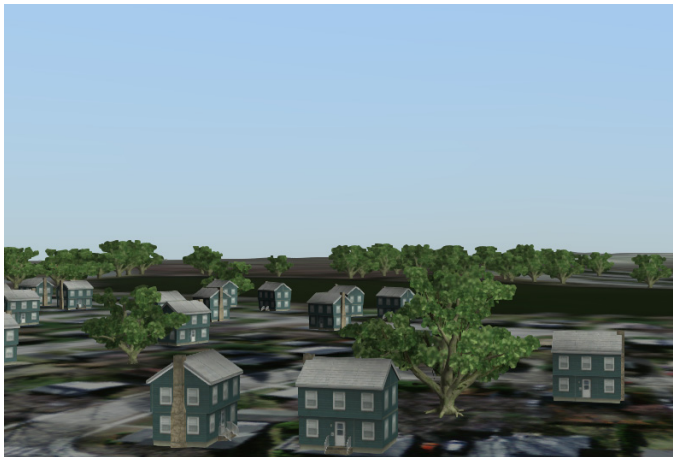


Figure 56-17. Little Pond with maples

13. Save the terrain.

You can build additional rules to add other types of vegetation to the terrain. When you are done, you can create another feature layer with the architecture shapefile (*InfrastructureP.shp*) and add light posts, fire hydrants, and street signs to the terrain.

56.2. Creating a Terrain with an Earth File

In "Composing a Terrain through the GUI," on page 56-2, we showed you how to compose a terrain by adding various terrain components in the VR-Forces GUI and saving the resulting terrain as an MTF file. Another way to load terrain into VR-Forces is by using a terrain server. Terrain servers are configured in earth files, which 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 or elsewhere on a network or the internet. Data is streamed over the network and paged in as needed.

In this tutorial we use the same local data we used for the Little Pond terrain in the previous tutorial, except that instead of adding the data through the GUI, we create an earth file to load it.

VR-Forces comes with a terrain *VR-TheWorld Online - Little Pond.earth* that merges the 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.

56.2.1. 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>
```

56.2.2. Add Elevation Data to the Earth File

Elevation data is configured using an `elevation` element. We want to specify the same elevation data here that we did in the previous Little Pond tutorial. 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 location of data in an earth file is provided in a `url` element. We need to use the gdal driver to load elevation data.

56.2.3. Add Imagery to the Earth File

Image data is configured using `image` elements. 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>
```

The Little Pond terrain has two image files, a large low-resolution image and a smaller higher resolution image. 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 code, *LittlePond.tif* would be underneath *W72N42.tif* and you would not be able to see it.

We use the gdal driver to load the images.

56.2.4. 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 the previous Little Pond tutorial, we added feature layers using *BuildingsP.shp* and *VegetationP.shp*. Then we added one kind of building (green colonial garrison house) and one kind of plant (big leaf maple tree). 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:

```
<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>
```

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.

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 *VR-TheWorld Online - Little Pond.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 56-18). Remember that in the first Little Pond tutorial, you built a rule for extracting props from the feature layer. You chose TYPE in the field list. That is because the TYPE column lists the different types of buildings in the shapefile. You do the same thing with the `selector` element.

	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 56-18. *BuildingsP.shp*

Add Vegetation to the Earth File

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>
```

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 TreesBigLeafMaple. VR-Forces adds ST to form TreesBigLeafMapleST, finds the model definition TreesBigLeafMapleST, and loads the model specified in the model definition.

56.2.5. 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>
```

56.2.6. Save the Earth File

The convention for saving earth files is to save them in the *./userData/servers directory*. Save the file as *LittlePond.earth*.

56.2.7. Load the Earth File and Save is as an MTF File

Your earth file for Little Pond is complete. Now you can load it in VR-Forces and compare it to the terrain you created through the GUI. 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 *LittlePond.earth*.
6. Click Connect. The terrain loads. It is visible as a small patch on the earth's sphere (Figure 56-19).

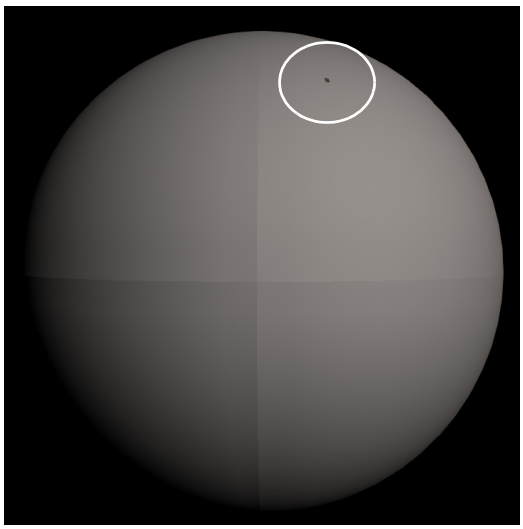


Figure 56-19. Little Pond terrain

7. Zoom in on the terrain (Figure 56-20). Compare it to the terrain you created in the first Little Pond tutorial. Notice that the buildings are the same, but there are more types of vegetation.

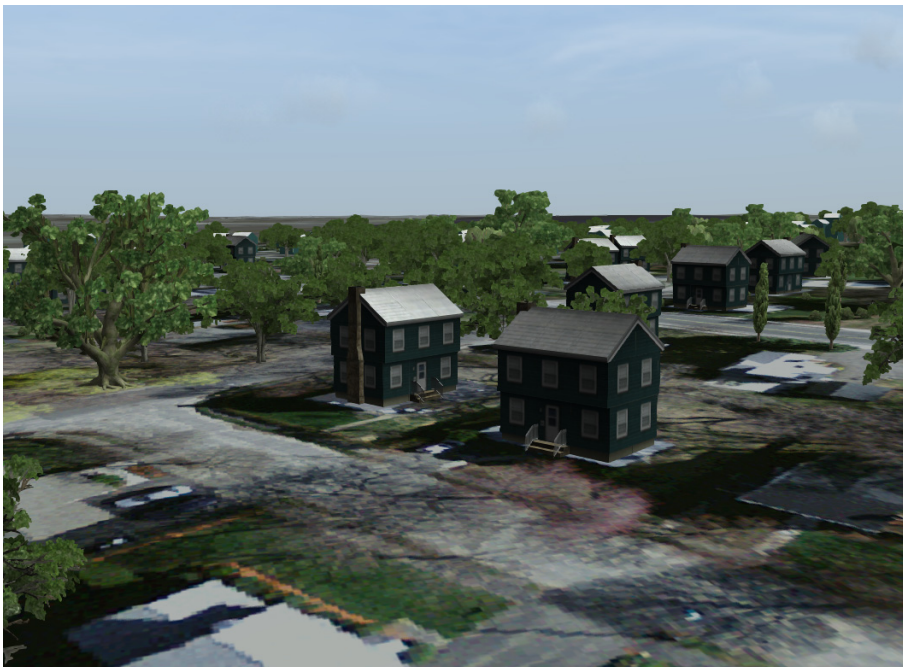


Figure 56-20. Little Pond terrain

8. Choose **File** → **Save Terrain**. The Save Terrain dialog box opens.
9. Give the terrain a name. Do not use Little Pond unless you want to overwrite the Little Pond terrain provided with VR-Forces.

56.3. Extracting Props from a Terrain

In the tutorial "Composing a Terrain through the GUI," on page 56-2, we extract props from a feature layer by building a rule in the Add Props page. In this tutorial, we demonstrate the other way 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 56-21).
2. Select the external references that you want to extract as props and extract them ("Extract the Props," on page 56-21).
3. Optionally, change the model definitions ("Change the Model Definition for a Prop," on page 56-24).

56.3.1. 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.

56.3.2. 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 56-21.



Figure 56-21. 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 56-22). 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.

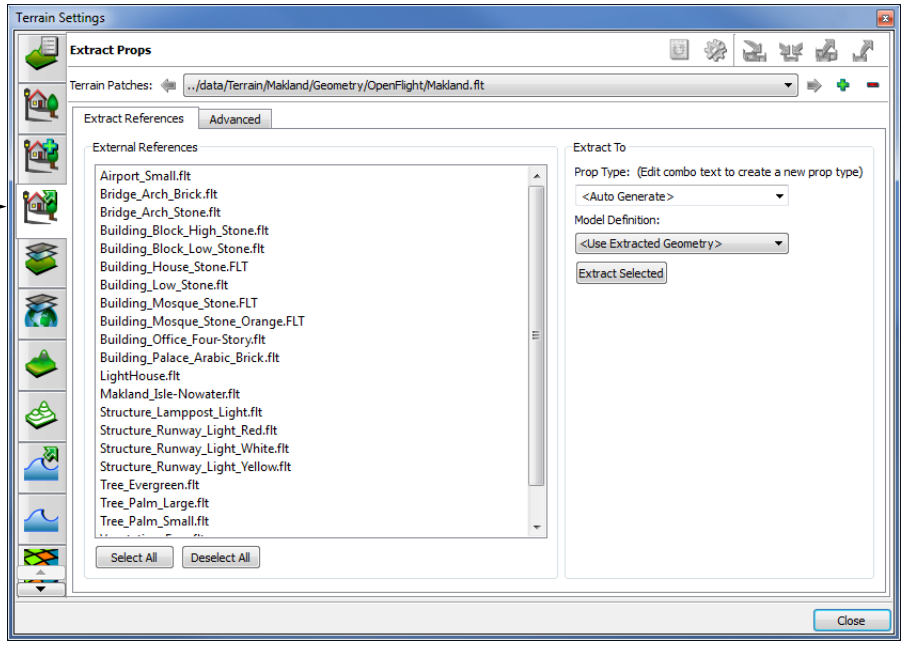


Figure 56-22. Extract props page

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 56-23). 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 56-23. Selected props

56.3.3. 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 56-24).

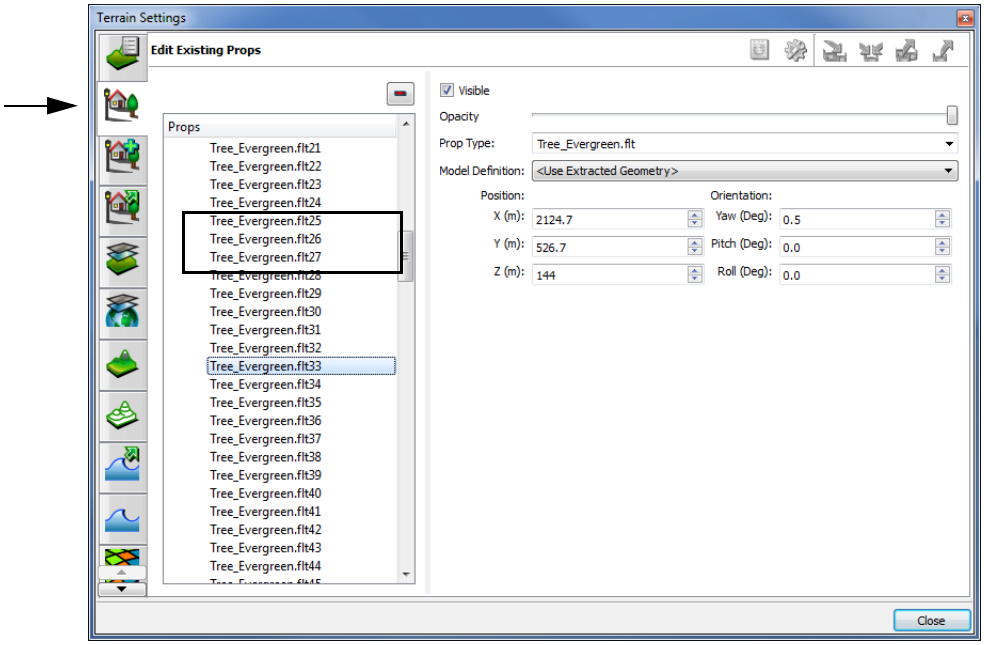


Figure 56-24. Edit Existing Props page

4. In the Model Definition box, select a model definition from the list, such as Trees-GiantRedwoodST. The tree changes to the new model definition (Figure 56-25).



Figure 56-25. 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.



XI. Configuring Visual Models

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. The chapters in this section explain how models and iconography are configured and mapped to object types.

XI.1. Introduction to VR-Forces 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 58-2.)

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 56-4.)

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.

Schemas define the most basic scene elements, such as entity, SpeedTree, 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 56-1 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 (4). The TrackedM1A2DESERT model definition has a **filename** parameter that specifies *M1A2.medf* as the model to use for entities that use this model definition. The TrackedM1A2DESERT model definition has a **filename** parameter because it uses the Entity schema and the Entity schema requires a **filename** (5).

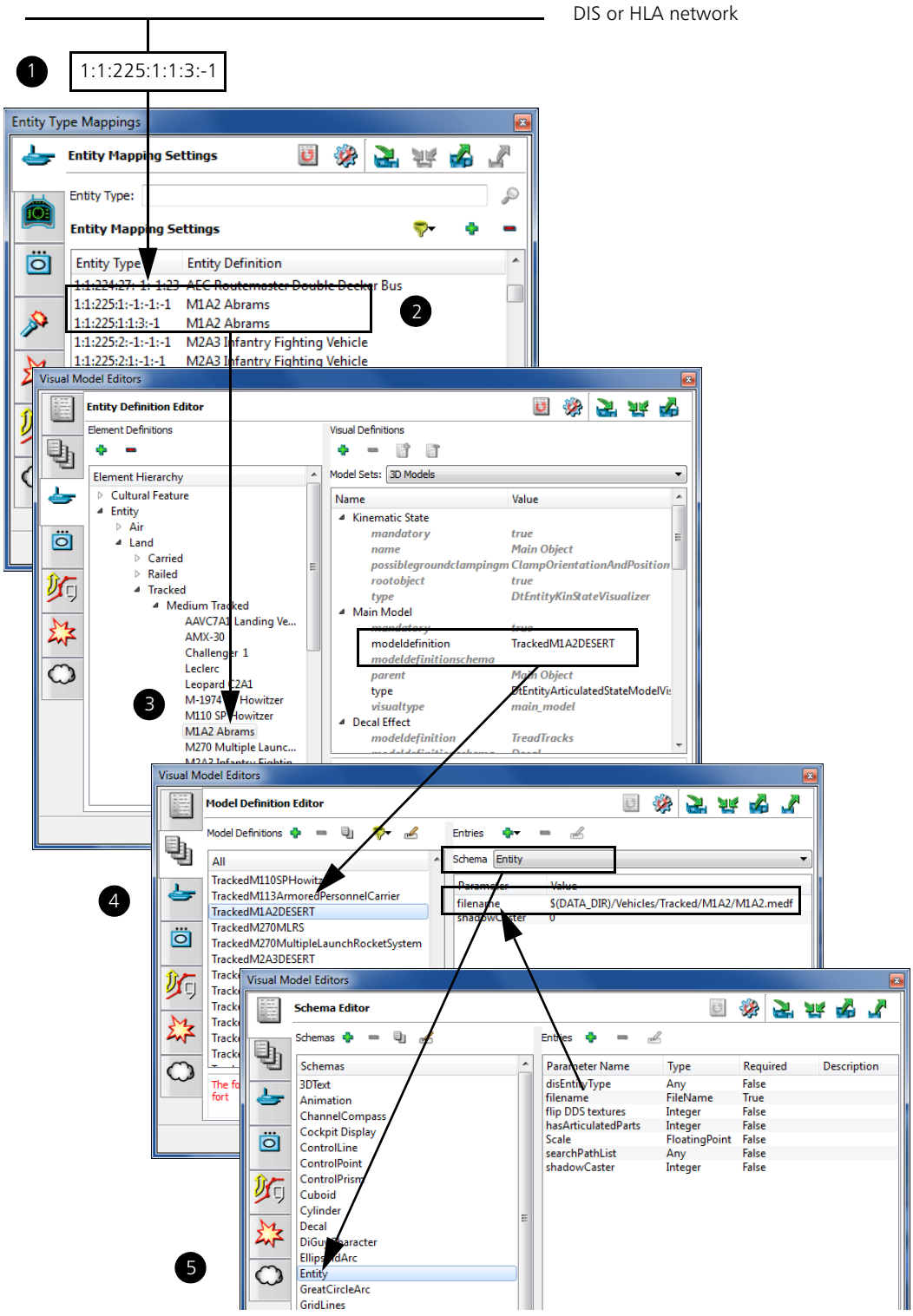


Figure 56-1. Element definitions and model mapping



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.

XI.1.1. 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 59-23.

You can provide your own models if they are in one of the following supported formats:

- OpenFlight.
- 3ds.
- OBJ.
- COLLADA (*.dae*).
- OSG (*.osg*, *.osgb*, and *.ive*).
- NewTek LightWave (*.lwo* and *.lws*).
- MÄK GDB.

XI.1.2. 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 43-11.



57. Model and Element Definitions

In Section XI, "Configuring Visual Models," 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. "Adding a Model to VR-Forces," on page 62-2 walks you through the process of adding a model.

- Managing Model and Element Definitions 57-2
- Creating and Editing Schemas 57-3
  - Creating Schemas 57-4
  - Deleting Schemas 57-5
  - Copying Schemas 57-5
  - Adding a Parameter to a Schema 57-6
  - Deleting a Parameter from a Schema 57-7
  - Editing a Schema Parameter 57-7
- Creating and Editing Model Definitions 57-8
  - Creating a Model Definition 57-9
  - Deleting a Model Definition 57-10
  - Copying a Model Definition 57-11
  - Filtering the List of Model Definitions 57-11
  - Editing a Model Definition 57-12
  - Saving Model Definitions 57-14
- Creating and Editing Element Definitions 57-14
  - Creating an Element Definition 57-15
  - Editing an Element Definition 57-16
  - Deleting an Element Definition 57-20
  - Saving Element Definitions 57-20

57.1. Managing Model and Element Definitions

This chapter 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 Model Editor to configure the new simulation object.

57.2. 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 4-23.
- Changes to element definitions do not take effect until you restart VR-Forces.

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
- SpeedTree
- TacticalSmoke
- Terrain
- Tree
- WidgetTheme.

57.2.1. 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 57-1).

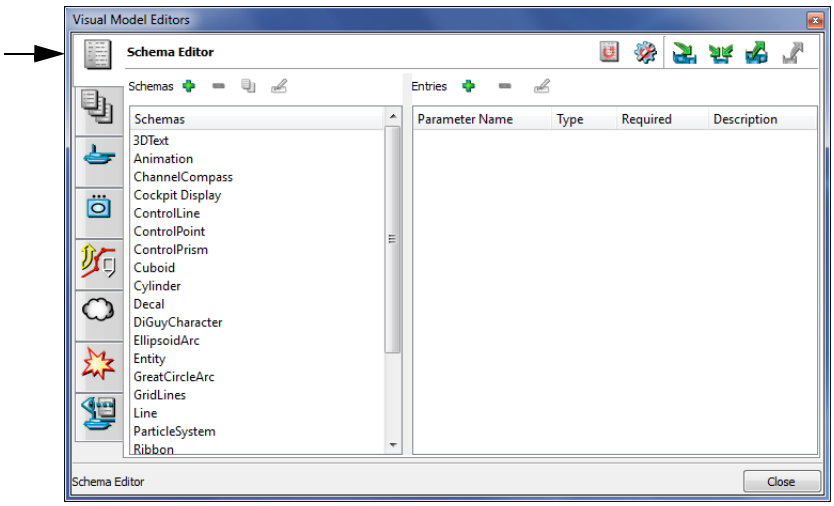


Figure 57-1. Schema Editor page

3. Click the Add button (+) next to the Schemas label. The Create Instance dialog box opens (Figure 57-2).

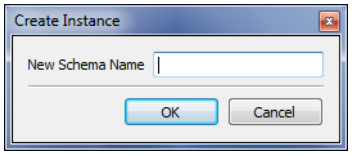


Figure 57-2. Create Instance dialog box

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 57-6.

57.2.2. 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 57-1).
3. In the Schemas list, select the schema you want to delete.
4. Click the Delete button () next to the Schemas label. You are prompted to confirm the deletion.
5. Click Yes. The schema is removed from the list.

57.2.3. 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 57-1).
3. Select the schema that you want to copy.
4. Click the Duplicate button (). The Duplicate Schema dialog box opens.
5. Type a name for the copied version of the schema.
6. Click OK.
7. The new schema is added to the list.
8. Edit the parameters as described in "Editing a Schema Parameter," on page 57-7.

57.2.4. 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 (Figure 57-1).
3. In the Schemas list, select the schema to which you want to add a parameter.
4. Click the Add button (+) next to the Entries label. A parameter called "key" is added to the parameter list (Figure 57-3).

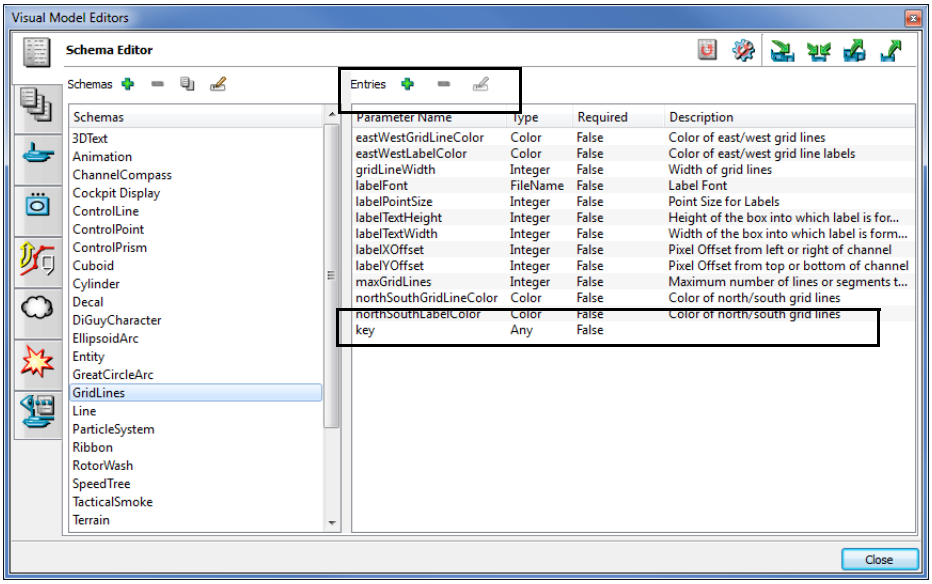


Figure 57-3. New schema parameter

5. Double-click "key". The Rename Parameter dialog box opens.
6. Type a name for the parameter.
7. If the parameter has a specific type, click the value in the Type column and edit it.
8. If the parameter is required, click the value in the Required column and change it to True.
9. 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.

57.2.5. 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 57-1).
3. In the Schemas list, select the schema from which you want to delete a parameter.
4. Select the parameter that you want to delete.
5. Click the Delete button (−) next to the Entries label.

57.2.6. 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 57-1).
3. In the Schemas list, select the schema for which you want to edit a parameter.
4. Click the parameter attribute that you want to edit. The attribute becomes editable.
5. Type the new attribute value.

57.3. 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 57-3.) 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.

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 57-4).

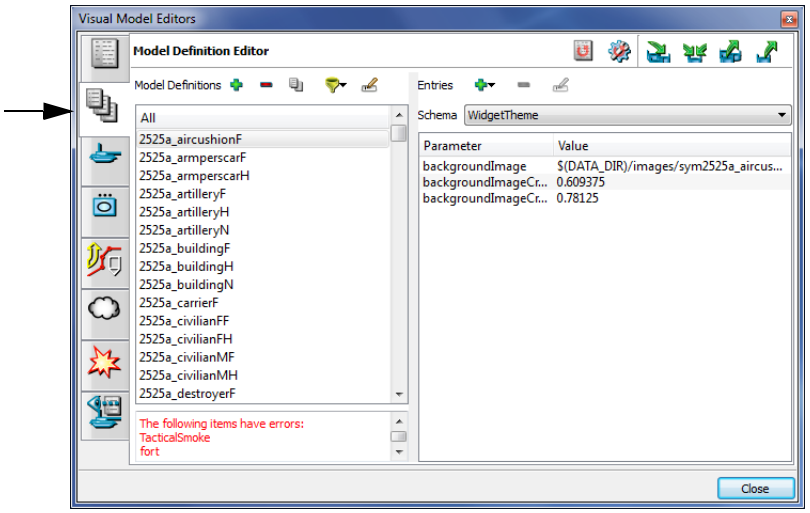


Figure 57-4. Model Definition Editor page

- 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 64-16.
- Changes to element definitions do not take effect until you restart VR-Forces.

57.3.1. Creating a Model Definition

To create a model definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 57-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 57-11.)
4. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens (Figure 57-5).

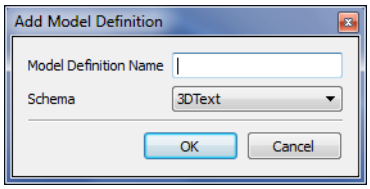


Figure 57-5. Add Model Definition dialog box

5. 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.

---

6. In the Schema list, select a schema for the model. (This list matches the list of schemas on the Schemas page.)
7. Add parameters to the model, as described in "Adding Parameters to a Model Definition," on page 57-12.

57.3.2. Deleting a Model Definition



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 57-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 57-11.)
4. Select the model that you want to delete.
5. Click the Delete button () next to the Model Definitions label. You are prompted to confirm the deletion.
6. Click Yes. The model definition is removed from the list.

57.3.3. 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.

To copy a model definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 57-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 57-11.)
4. Select the model that you want to copy.
5. Click the Duplicate button ( ). The Duplicate Model Definition dialog box opens (Figure 57-6).

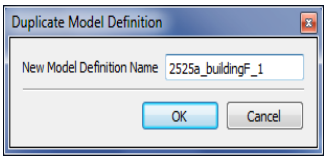


Figure 57-6. Copy model definition

6. Type a name for the copied version of the model.
7. Click OK.
8. The new model definition is added to the list.
9. Edit the parameters as described in "Editing a Model Definition," on page 57-12.

57.3.4. 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 to more easily work with a specific type of model.

➤ To filter the list of model definitions, click the Filter button ( ) and select a schema type from the list.

57.3.5. 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.



If you change the model definition for a simulation object that is already in the scene, the updated definition is not applied to the entity. To apply an edited model definition to all simulation objects, disconnect from the exercise and then reconnect.

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. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 57-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 57-11.)
4. Select the model for which you want to add a parameter.
5. Click the Add Parameter button (). A list of parameters is displayed (Figure 57-7).

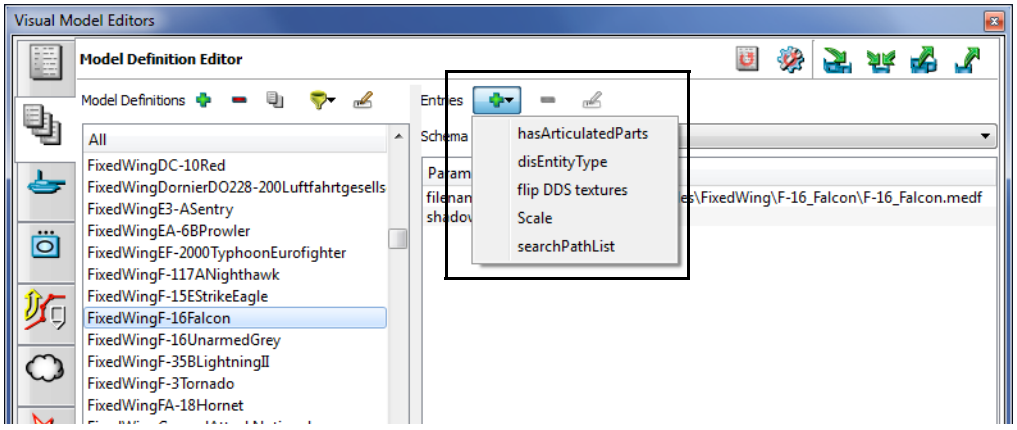


Figure 57-7. Add parameter list

6. Select the parameter you want to add. It is added to the parameter list.
7. Click the value for the parameter. The field becomes editable (Figure 57-8).

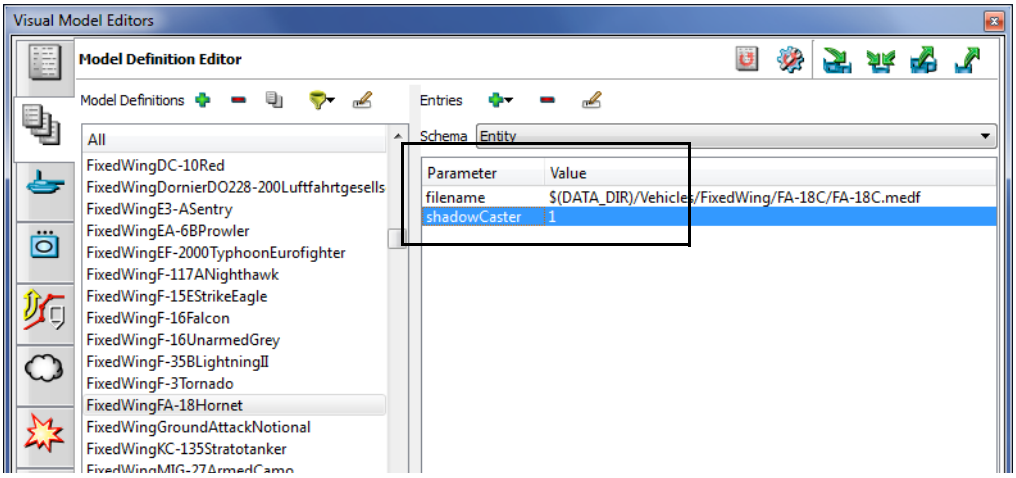


Figure 57-8. Editable parameter

8. Type or select a value for the parameter.
9. Click away from the field.

Editing a Model Definition Parameter

To edit a parameter:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 57-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 57-11.)
4. Select the model for which you want to edit a parameter.
5. Click the value for the parameter you want to edit. The field becomes editable (Figure 57-8).
6. Change the value for the parameter.
7. Click away from the field.

Deleting Parameters from a Model Definition

! 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 57-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 57-11.)
4. Select the model 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. The parameter is removed.

57.3.6. Saving Model Definitions

Like all VR-Forces settings, when you change a model definition, the change is automatically saved. You can also use the common settings management controls for exporting, loading, and merging settings. For details about the common settings, please see "Managing VR-Forces Settings," on page 4-23.

57.4. 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).

Element definitions use a tree-style hierarchy to assign visual definitions. 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.



Changes to element definitions do not take effect until you restart VR-Forces.

57.4.1. Creating an Element Definition

To create 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 create.
3. Expand the element definition hierarchy to the level at which you want to add an element definition and select the parent element (Figure 57-9).

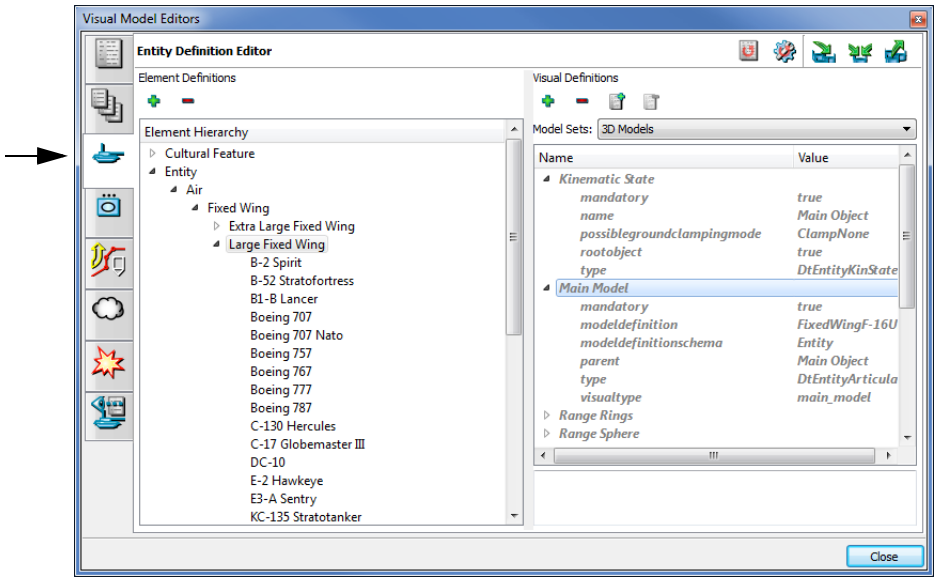


Figure 57-9. Entity Definition Editor page

4. Click the Add button (✚). The Create Element Definition dialog box opens.
5. Type a name for the element definition. (You cannot edit the name after you create it, so try to avoid typing errors.)
6. Click OK. The new element is added to the list.
7. Optionally, edit the visual definitions assigned to the element. For details, please see "Editing a Visual Definition," on page 57-17.
8. Optionally, add a visual definition to the element. For details, please see "Adding a Visual Definition," on page 57-16.
9. Optionally, save the element to a file. For details, please see "Saving Element Definitions," on page 57-20.

57.4.2. 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.

Adding a Visual Definition

To add a visual definition to 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 edit (Figure 57-9).
3. Select the element definition that you want to edit.
4. Optionally, on the Model Sets list, select the model set you want to use.
5. Click the Add button (✚) next to the Visual Definitions label. The Create Visualizer Definition dialog box displays a list of available visual definitions.

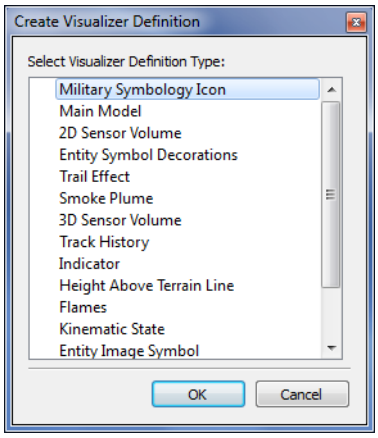


Figure 57-10. Create Visualizer Definition dialog box

6. Select the visual definition that you want to add.
7. Click OK. The visual definition is added to the element.
8. Optionally, edit the parameters for the visual definition. For details, please see "Editing a Visual Definition," on page 57-17.

Editing a Visual Definition

Visual definitions and their attributes are displayed either as gray italicized text or as black text (Figure 57-11). 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.)

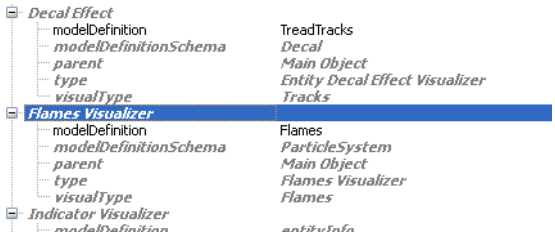


Figure 57-11. Visual definition entry

To edit a visual definition parameter:

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 edit (Figure 57-9).
3. Select the element definition that you want to edit.
4. 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 57-12), or the field becomes editable.

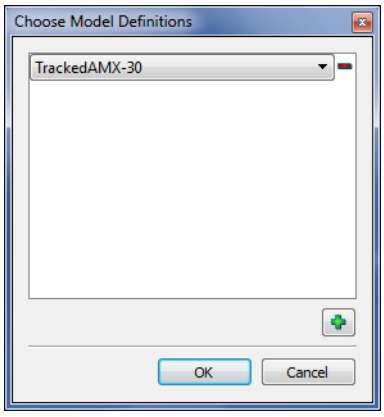


Figure 57-12. Choose Model Definitions dialog box

5. 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.

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. 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 edit (Figure 57-9).
3. Select the element definition that you want to edit.
4. 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 57-13).

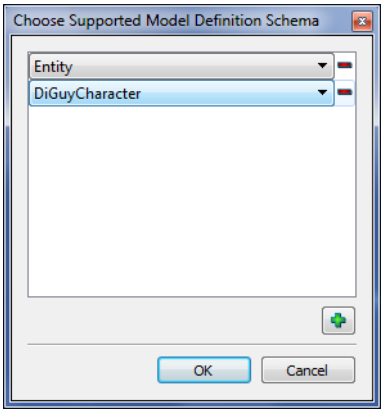


Figure 57-13. Parameter with multiple values

5. To add a value, click the Add button (+). A new list is added to the window.
6. Select a value from the new list.
7. To remove a value, click the Delete button (−) next to the list for that value.
8. 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.

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.

To add an attribute to a visual definition:

1. Select the visual definition that you want to edit.
2. Click the Add Attribute button ( ). The Add Visualizer Definition Attribute window opens. It lists the attributes that are available for the visual definition (Figure 57-14).

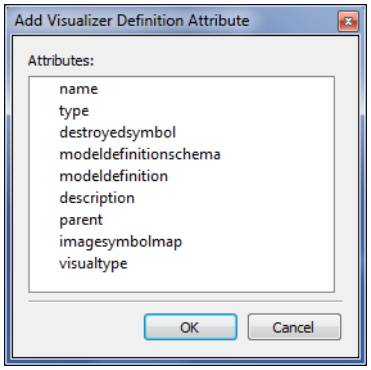


Figure 57-14. Add Visualizer Definition Attribute window

3. Select the attribute that you want to add.
4. Click OK. The attribute is added to the visual definition.
5. 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.

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. Select the visual definition that you want to edit.
2. Select the attribute that you want to delete.
3. Click the Delete Attribute button ( ).

57.4.3. Deleting an Element Definition



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 57-9).
3. Select the element definition that you want to delete.
4. Click the Delete button (−) next to the Entity Definitions label.
5. Confirm the deletion.

57.4.4. Saving Element Definitions

Changes to element definitions get saved automatically. You can also save element definitions using the standard VR-Forces options for saving settings. For details, please see "Managing VR-Forces Settings," on page 4-23.



58. Mapping Models and Effects

This chapter explains how to map element definitions to DIS and HLA object types. Chapter 62, *Model Tutorials* walks you through the process of adding a model.

Introduction to Object Type Mapping 58-2
- Adding an Object Type Mapping 58-3
- Editing an Object Type Mapping 58-4
- Filtering the Element Definition List 58-5
- Deleting an Object Type Mapping 58-6
- How VR-Forces Maps DI-Guy Models 58-6

Clearing the Model Instancing Cache 58-7

58.1. 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 section 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.

58.1.1. 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. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 58-1). 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.

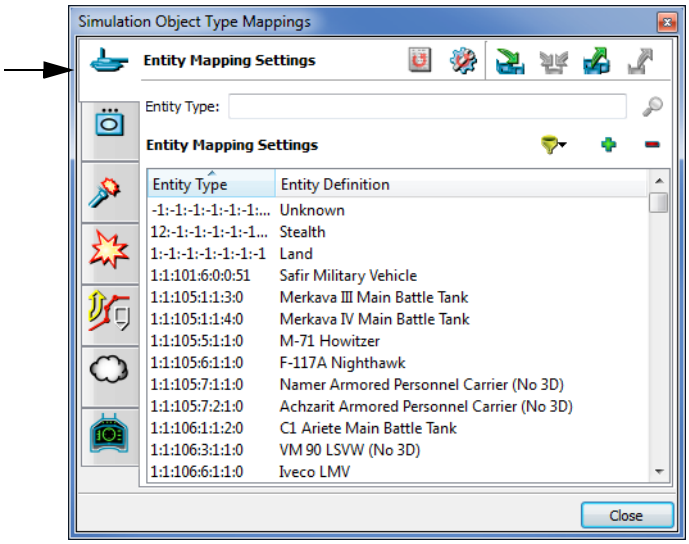


Figure 58-1. Entity Mapping Settings page

3. Click the Add button (+). The Create New Entity Type Model Mapping dialog box opens (Figure 58-2). It lists a default object type enumeration.

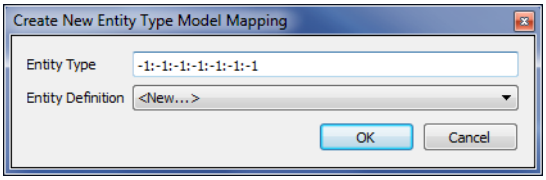


Figure 58-2. Create New Entity Type Model Mapping dialog box

4. In the Entity Type text box, type the enumeration you want to map to a model. (Separate each value with a colon(:).)
5. In the Entity Definition list, select the entity definition that you want to map to this object type.

6. Click OK.
7. The new model mapping is added to the list.

58.1.2. 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. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 58-1).
3. Select the model mapping that you want to edit. If you know the entity enumeration that you want to edit, you can type the enumeration in the Entity Type text box and click the Search button (🔎). VR-Forces highlights the entry for that enumeration, if it is present.
4. 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.
5. 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 57-11.)
   b. In the Entity Definition column, click the element definition to make it an editable list (Figure 58-3).
   c. Select the new element definition from the list.
   d. Click someplace else in the window.

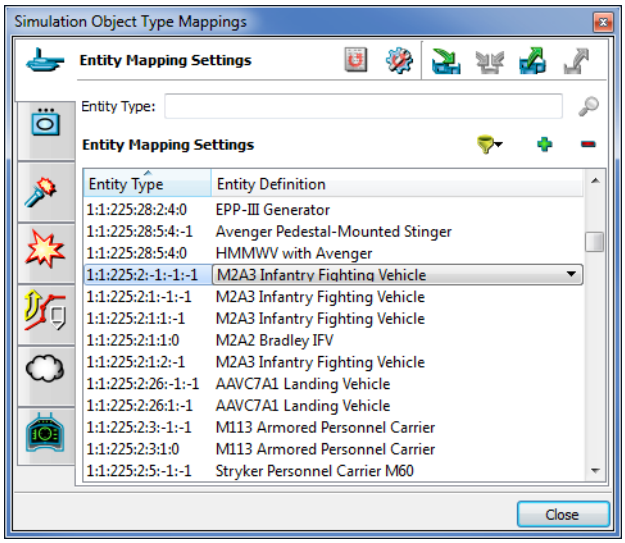


Figure 58-3. Editable entity definition

58.1.3. Filtering the Element Definition List

You can filter the list of element definitions that are available in the Entity Definition (Figure 58-3) 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 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.

58.1.4. 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 58-1).
3. Select the model mapping that you want to delete. If you know the entity enumeration that you want to edit, you can type the enumeration in the Entity Type text box and click the Search button (🔎). VR-Forces highlights the entry for that enumeration if it is present.
4. Click the Delete button (➖).

58.1.5. 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.

58.2. 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 58-4).

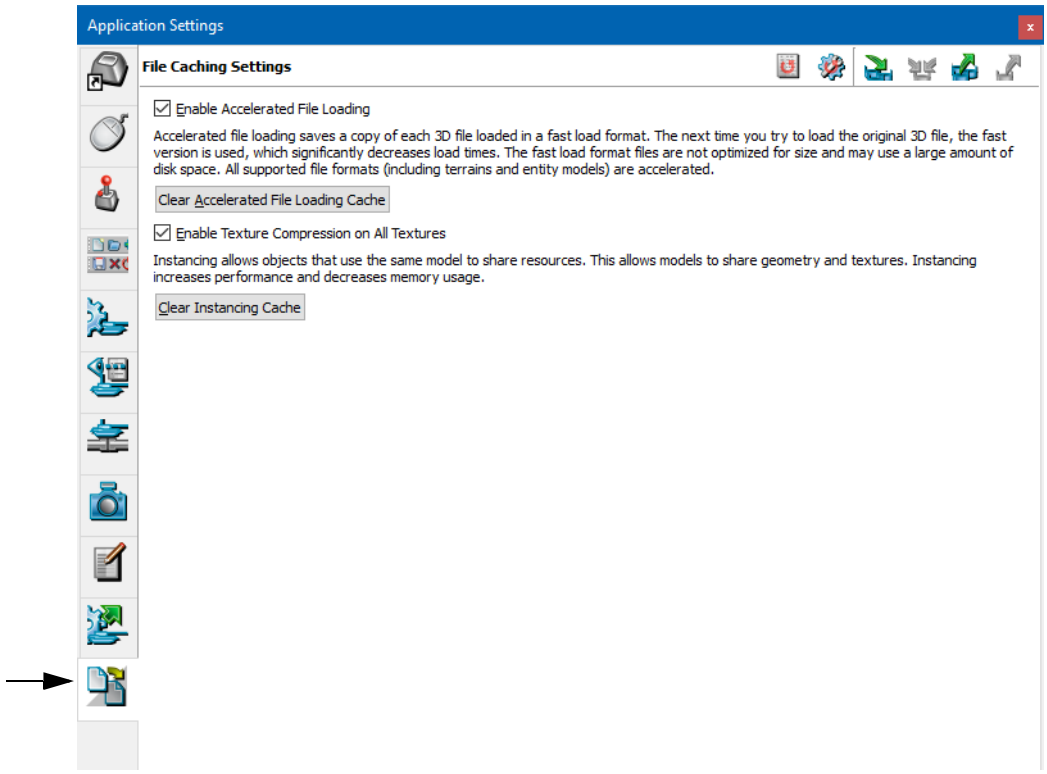


Figure 58-4. File Caching Settings page

3. Click Clear Instancing Cache.



59. Configuring Specific Visual Models

Chapter 57, *Model and Element Definitions* 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 59-2
Editing Font-Based 2D Icons 59-2
Using Images for 2D Icons 59-4
Adding Images for Simulation Object Icons 59-9
Adding New Cockpit Display Models 59-10
Installing a Cockpit DLL 59-11
Creating a Model Definition for a Cockpit 59-11
Adding Object Models as Props 59-13
Adding a Category to the Props Palette 59-13
Configuring Wakes 59-14
Configuring Tidal Stream Wakes 59-16
Adding Wind-based Controls to Models 59-16
Flipping DDS Textures for a Model 59-17
Configuring SpeedTree Trees 59-18
Randomizing the Size of SpeedTrees 59-20
Best Practices for Creating Models for VR-Forces 59-22
Compressing Model Files 59-23

59.1. 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 64-16.

59.1.1. 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.
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 59-1).

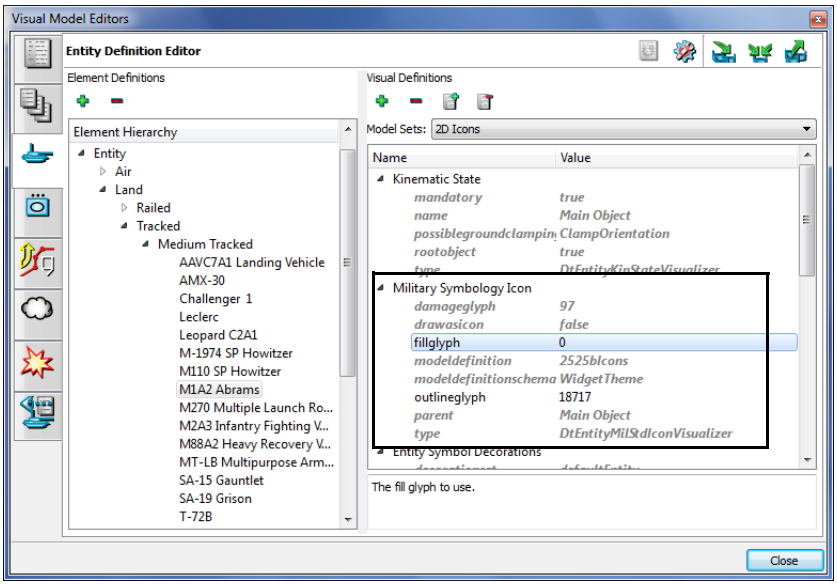


Figure 59-1. Military Symbol Icon visual definition

b. Click the value in the Value column. The Choose Font Glyph dialog box opens (Figure 59-2).

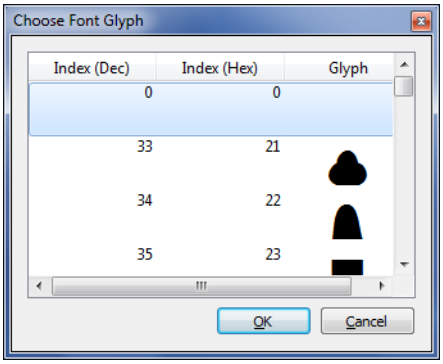


Figure 59-2. Choose Font Glyph dialog box

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.

59.1.2. 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 entity 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. (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 list, select Entity. (If you are editing unit images, in the Element Definitions list select Aggregate. If you are editing tactical graphics images, in the Element Definitions list select Waypoint. For all other tactical graphics, skip to step 8.)
5. In the Visual Definitions list, select Military Symbology Icon (Figure 59-3).

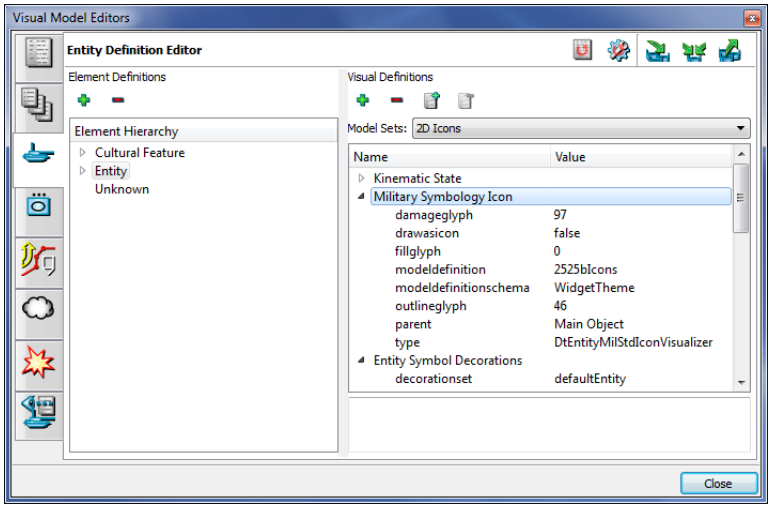


Figure 59-3. Entity element definition with Military Symbology Icon visual definition

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 59-4).

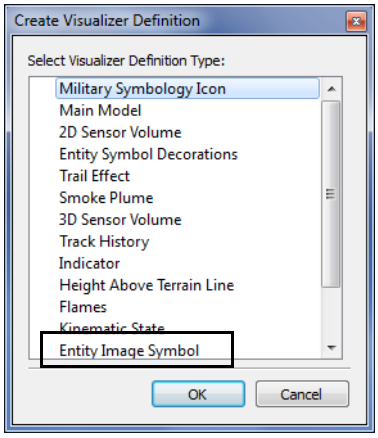


Figure 59-4. Create Visualizer Definition window

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 59-5).

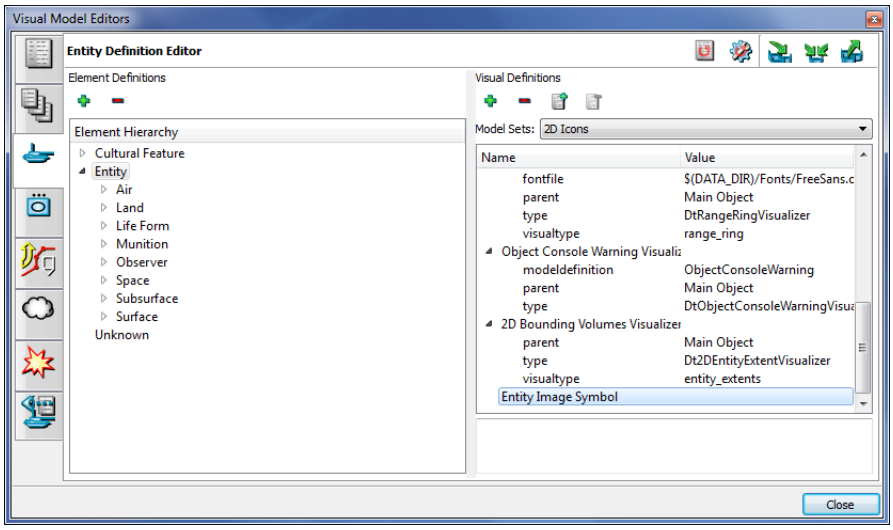


Figure 59-5. Entity element definition with Entity Image Symbol visual definition

11. Click the Add Attribute button (). The Add Visualizer Definition Attribute window is displayed (Figure 59-6).

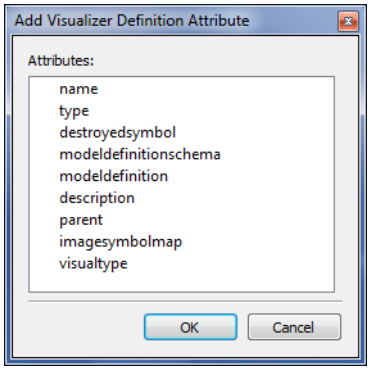


Figure 59-6. Add Visualizer Definition Attribute window

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 59-7).

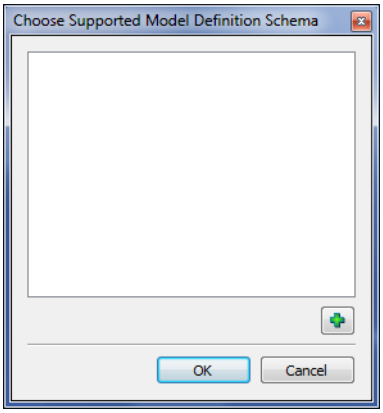


Figure 59-7. Choose Supported Model Definition Schema dialog box

20. Click the Add button. A list is added to the dialog box.
21. In the list, select **WidgetTheme** (Figure 59-8).

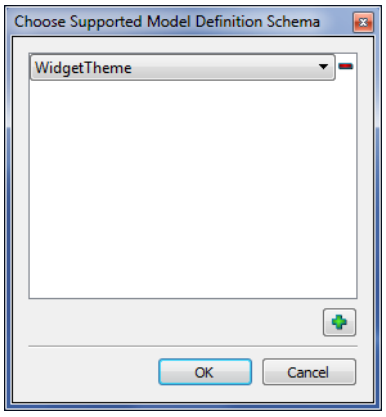


Figure 59-8. Completed Choose Supported Model Definition Schema dialog box

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 59-9).

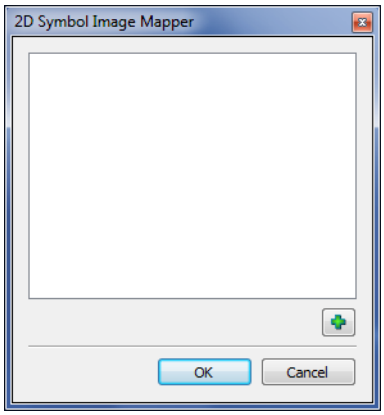


Figure 59-9. 2D Symbol Image Mapper dialog box

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 59-10).

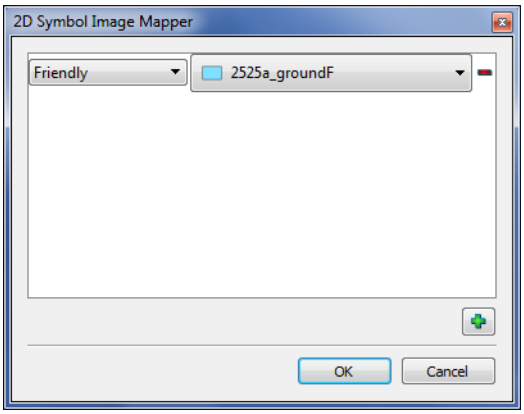


Figure 59-10. 2D Symbol Image Mapper dialog box with force chosen

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 59-11.

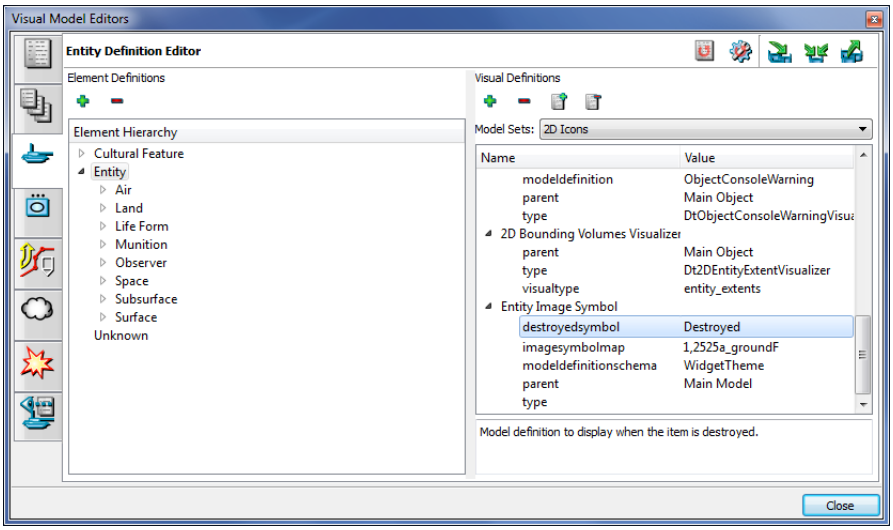


Figure 59-11. Entity Image Symbol visual definition

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.

59.1.3. 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 57-4).
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 57-8. 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.

59.2. 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 59-12 shows the model definition for a cockpit.

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.

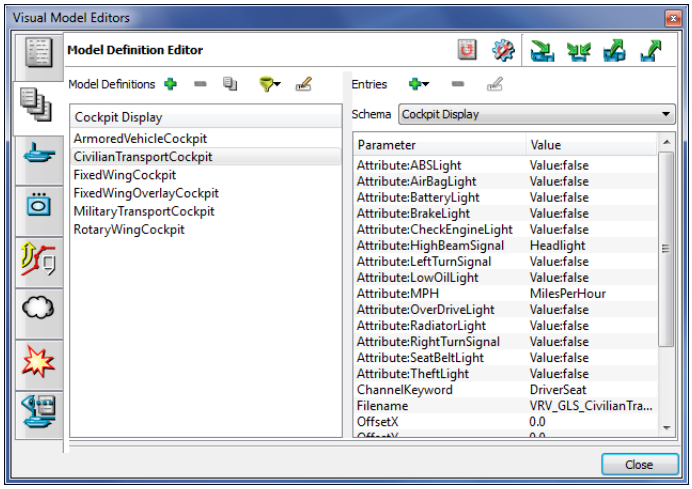


Figure 59-12. Cockpit model definition

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 58-3.

59.2.1. 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.

59.2.2. 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 57-9. Be sure to select Cockpit Display as the schema. A new model definition is added to the display (Figure 59-13). The message window lists the parameters that you must add. These are the parameters defined by the schema.

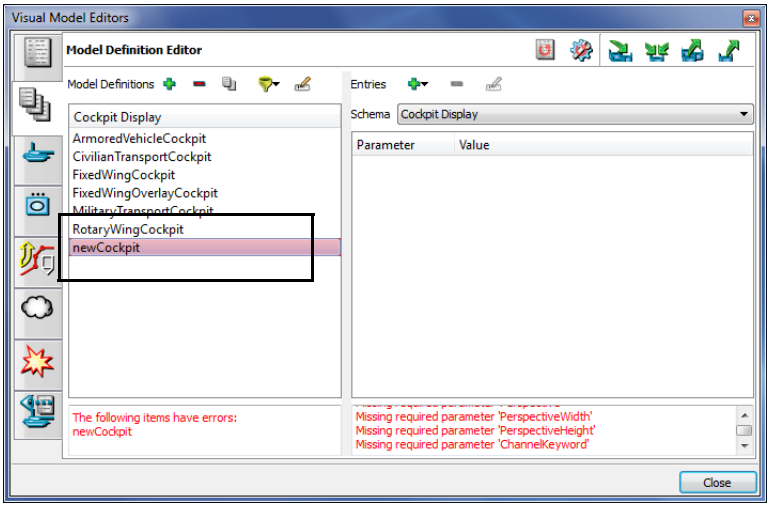


Figure 59-13. New cockpit model definition

2. Add the required parameters, as described in "Adding Parameters to a Model Definition," on page 57-12. Table 59-1 describes the required parameters.

Table 59-1: 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.

Table 59-1: Required parameters for cockpit models

Parameter	Description
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 57-12.
   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 59-2.
4. Map an object type to your definition, as described in "Adding an Object Type Mapping," on page 58-3.

Table 59-2: 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).
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).

Table 59-2: Updaters provided

Updater	Description
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.

59.3. Adding Object Models as Props

Props are terrain objects that you can select independently of the terrain. "Extracting Props from a Terrain Patch," on page 50-24 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 57-8. 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.)

59.3.1. 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.

59.4. 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.

To configure a ship wake:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 59-14).

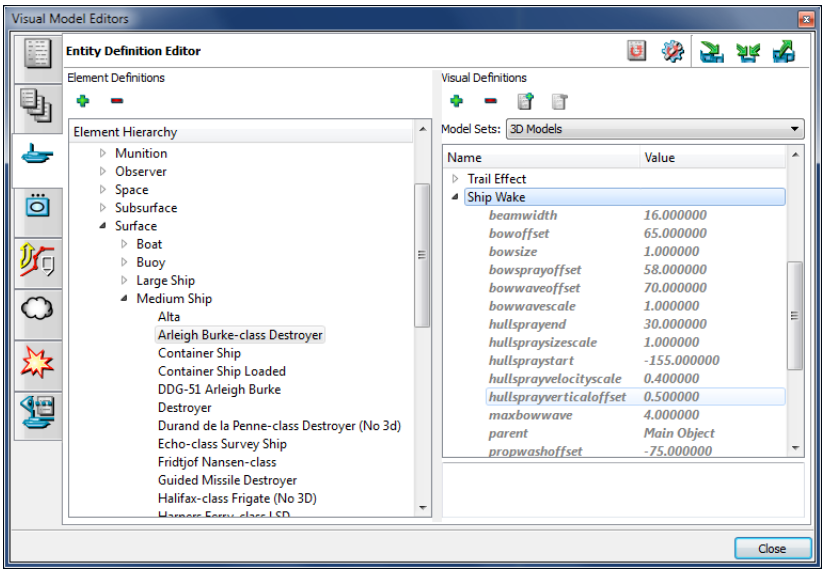


Figure 59-14. Ship Wake visual definition

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 59-3 describes each parameter.

Table 59-3: 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.

Table 59-3: Ship Wake visual definition attributes

Parameter	Description
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.
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.
propwashoffset	An offset from the ship position from which to generate propeller backwash effects.
shiplength	The length of the object generating the 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.
sprayvelocityscale	A scaling factor for spray effects at the bow of the ship. This is applied to the initial velocity of the spray particles.
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.

59.4.1. 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, BeaconsLattice-Wake, 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 11-15.

59.5. 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 meta files.

A meta file 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 meta file 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.

Meta files are user-editable, but require understanding of the annotations and naming conventions in the model file itself.

The meta files are JSON configuration files. They 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.

Each section has a `Linkage` entry and a `FeatureData` entry. The `Linkage` entry specifies an OSG node that should have the feature specified in the `FeatureData` entry.

Linkages can be specified by node name or WRM Specification comment (for example, @dis articulated_part 7872). When a model or terrain file is loaded and a meta file is associated with that file, the model or terrain file is searched for any node or nodes specified in any `Linkage` entries in that meta file and the necessary instances to implement the `FeatureData` indicated in that `Linkage`/`FeatureData` pair are created.

`FeatureData` sections indicate a **FeatureName** (which is mapped to a creator that will instantiate the appropriate object to manage that feature) and any additional parameters needed to manage the feature. The meta file used for wind features supports two features - WindSwitchPart and WindDirPart.

- WindSwitchPart. The WindSwitchPart feature is used to change the state of a switch node based on the current wind speed. The flag models used by MÄK 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 is used to modify 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 counterclockwise direction.

For information about adding new meta files or new meta file features, please see section 8.6, Transforming Nodes Programmatically, in VR-Forces *Front-End Developers Guide*.

59.6. 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 50-34.)

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 57-4).
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.

59.7. 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 73-10. Table 59-4 describes the settings you can change.

Table 59-4: 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.
Transmission Scalar	Multiplies the built-in transmission lighting value in each SpeedTree by the specified amount.
Alpha Scalar	Multiplies the built-in alpha 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 59-15).

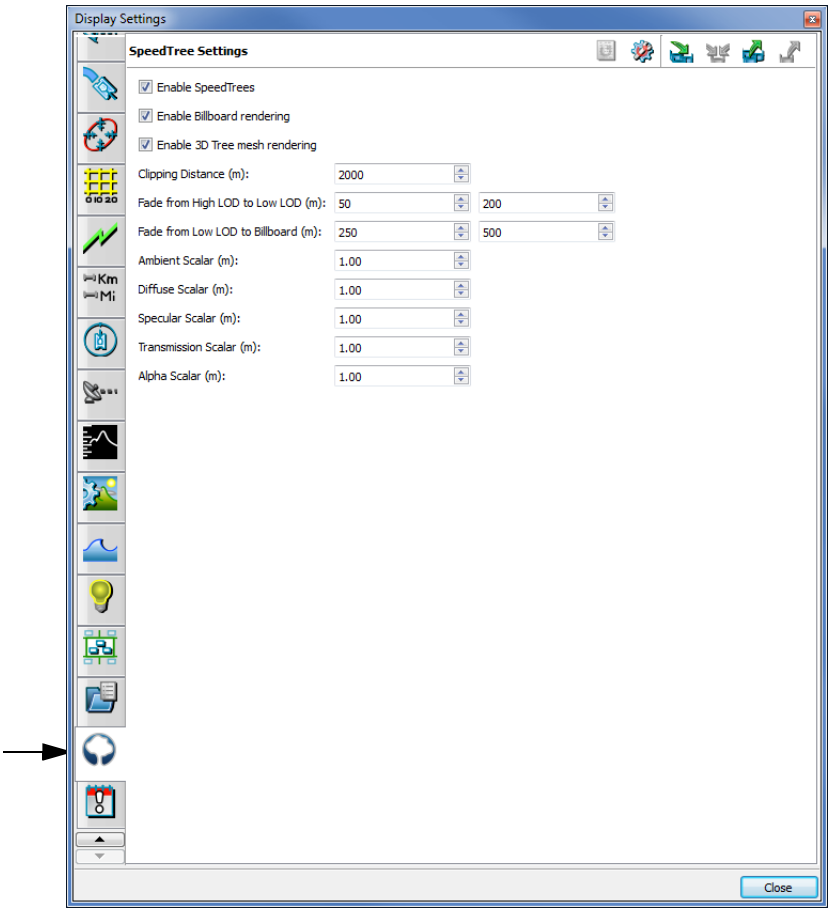


Figure 59-15. SpeedTree Settings page

3. Update parameters as desired.
4. Click Close.

59.7.1. 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 59-16).

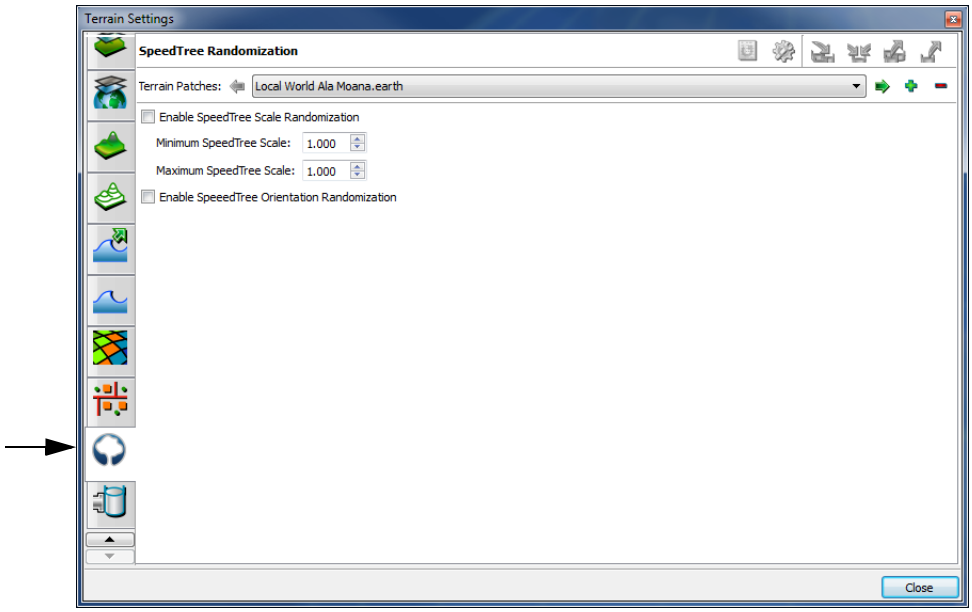


Figure 59-16. SpeedTree Randomization page

3. To enable randomization of the SpeedTree scale, select the Enable SpeedTree Scale Randomization check box.
4. If you enabled Enable SpeedTree Scale Randomization, specify values for the Minimum Tree Scale and Maximum Tree Scale.
5. To enable randomization of tree orientation, select the Enable SpeedTree Orientation check box.

59.8. Best Practices for Creating Models for VR-Forces

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.

59.9. Compressing Model Files

The models and image files provided with VR-Forces are shipped in compressed formats (MÄK Encrypted Data Format (MEDF) and MÄK 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, 3DS, and OBJ, 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 [-q simple_filename ... -p <integer> -s directory
   ... -z optimization -i -o -m extensions ...
   -x extensions ... --norecurse --directory directory
   -f filename -- -v -h]
```

Table 59-5 describes the arguments.

Table 59-5: medfTool arguments

Argument	Description
`(-- | --ignore_rest)`	Ignore all arguments after this one.
`--directory` *directory*	Specifies the directory to search for image and geometry files to convert into MEDF and MEIF. All subdirectories will be visited unless the `--norecurse` command line option is used.
`(-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.
`(-h | --help)`	Displays usage information.
`(-i | --ignore_errors)`	Ignore errors in the encryption process and continue without exiting.
`(-m | --image_extensions)` *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.
`(-o | --onlyIfNew)`	Only generates an encrypted file if the expected output file does not exist.

Table 59-5: medfTool arguments

Argument	Description
`(-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.
`(-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.
`(-v | --version)`	Display version information and exit.
`(-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
```



60. Configuring Emitter Volumes

This chapter explains how to configure display of emitter volumes.

Configuring Emitter Volumes 60-2
Configuring Emitter Volume Color 60-2
Controlling Emitter Volume Radius 60-3
Configuring Emitter Volume Segments 60-4

60.1. 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.

60.1.1. 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 60-1 describes the emitter volume visualizer attributes that affect emitter volume color.

Table 60-1: 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.
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.

60.1.2. 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 );
```

60.1.3. Configuring Emitter Volume Segments

VR-Forces displays a 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 60-1 shows segments set to 10° (the default) and 30°.

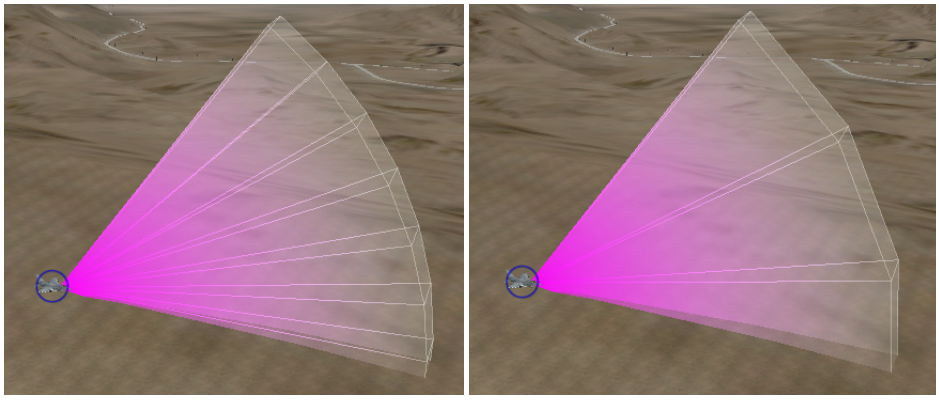


Figure 60-1. Emitter volume segments

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. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page.
3. Click the Filter button (▾) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
4. Select DefaultEllipsoidArc (Figure 60-2).

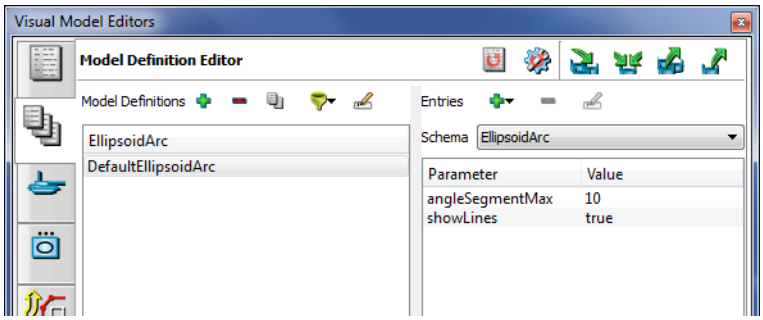


Figure 60-2. EllipsoidArc model definition

5. Change the value of the **angleSegmentMax** parameter. The minimum value is 10 (degrees); the maximum is 45.

Displaying Segment Outlines

By default, VR-Forces displays opaque outlines showing the segment edges along the surface of the emitter volume.

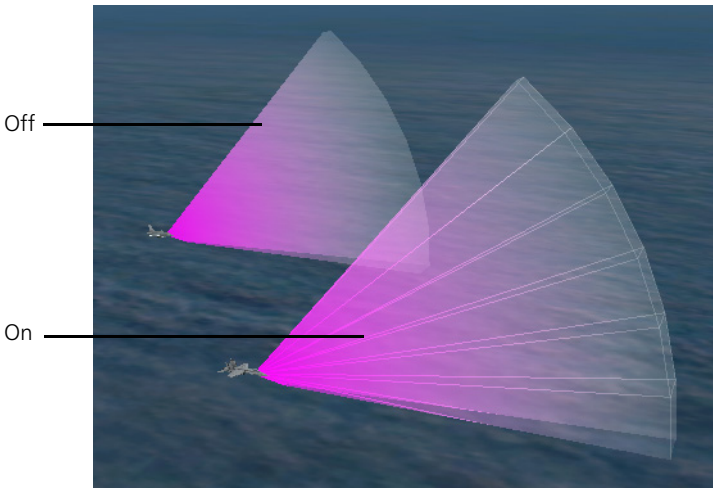


Figure 60-3. Segment outlines

To display or hide segment outlines:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page.
3. Click the Filter button (▾) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
4. Select DefaultEllipsoidArc.
5. Change the value of the **showLines** parameter.

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. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page.
3. Click the Filter button (▾) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
4. Select DefaultEllipsoidArc.
5. Click the Duplicate Model Definition button (). The Duplicate Model Definition dialog box opens with a default definition name.
6. Accept the generated name or type a new name for the model definition.
7. Click OK. The model definition is added to the list.
8. Change the parameter values for the new model definition as desired.
9. Select the Entity Definition Editor page.
10. Select the entity whose emitter volume you want to change.
11. Select the 3D Sensor Volume visualizer for the entity.
12. Click the Value column for the **modeldefinition** attribute. The Choose Model Definition dialog box opens. It has a list of model definitions.
13. Select the model definition that you just created.
14. Click OK.
15. If you are connected to an exercise, disconnect and then reconnect to enable the new model definition.



61. 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 MÄK Technologies (now VT MÄK), 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 used in the HLA RPR FOM.

Introduction 61-2
The OpenFlight File Format 61-2
Node Name and Comment Fields 61-3
External References and Instancing 61-3
Modeling for DIS Interoperability 61-3
The DIS Attribute Lexicon (DAL) 61-3
DAL Keywords 61-4
Model Coordinate Systems 61-5
Articulated Parts 61-7
Entity Appearances 61-8
Animation 61-11
Animation 61-11

61.1. 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. You can obtain the Model Specification at http://www.presagis.com/files/standards/wrm.pdf.

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.

61.2. 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.

61.2.1. 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.

61.2.2. 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.

61.3. 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.

61.3.1. 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.

61.3.2. 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 61-3.

  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 61-2.

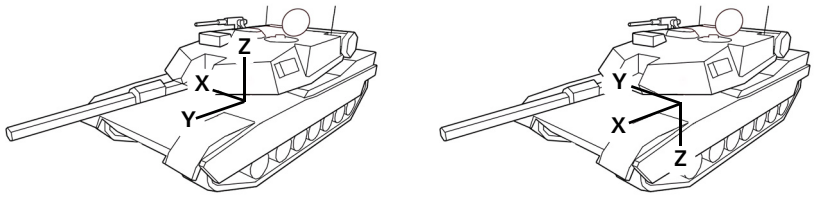
- `state` *`state`*,*`state`*... . Specifies the state of the appearance attribute with a DIS appearance state code. This keyword is placed in a Group node just below the switch node. For a list of supported values, please see Table 61-2.

61.3.3. Model Coordinate Systems

OpenSceneGraph (OSG) and the DIS Standard use different right-hand coordinate systems. Table 61-1 describes the two systems. Figure 61-1 illustrates them.

Table 61-1: Model coordinate systems

Coordinate	OpenSceneGraph	DIS
X	Right	Forward
Y	Forward	Right
Z	Up	Down



OSG coordinate system DIS coordinate system

Figure 61-1. 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.

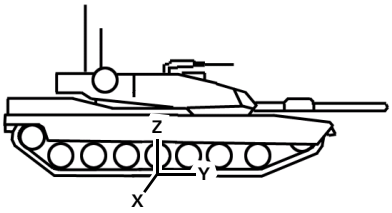
Location of the Origin for Models

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.

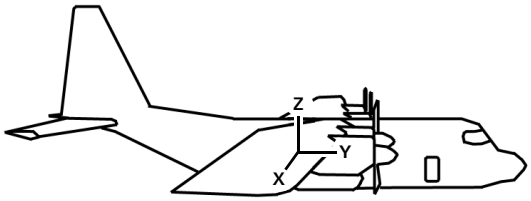
Figure 61-2 illustrates the location of a model's origin for each platform domain.

Land



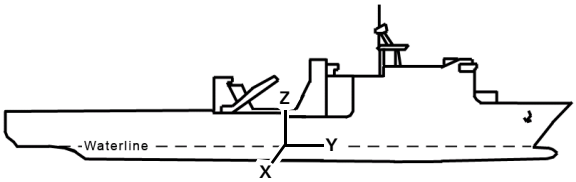
On the plane of ground contact, at the axis of yaw rotation.

Air



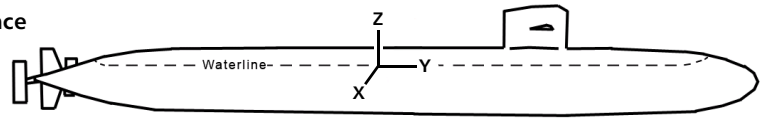
At the center of gravity.

Surface



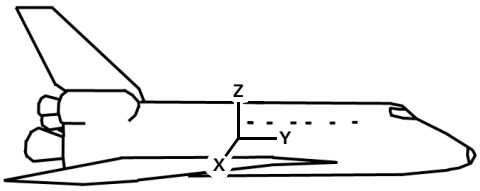
On the axis of the center of buoyancy, in the plane of the waterline for an unloaded ship.

Subsurface



On the axis of the center of buoyancy, in the plane of the waterline of a surfaced submarine.

Space



At the center of gravity.

Figure 61-2. Location of origin by domain

61.4. 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.

61.5. 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 61-3, 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.

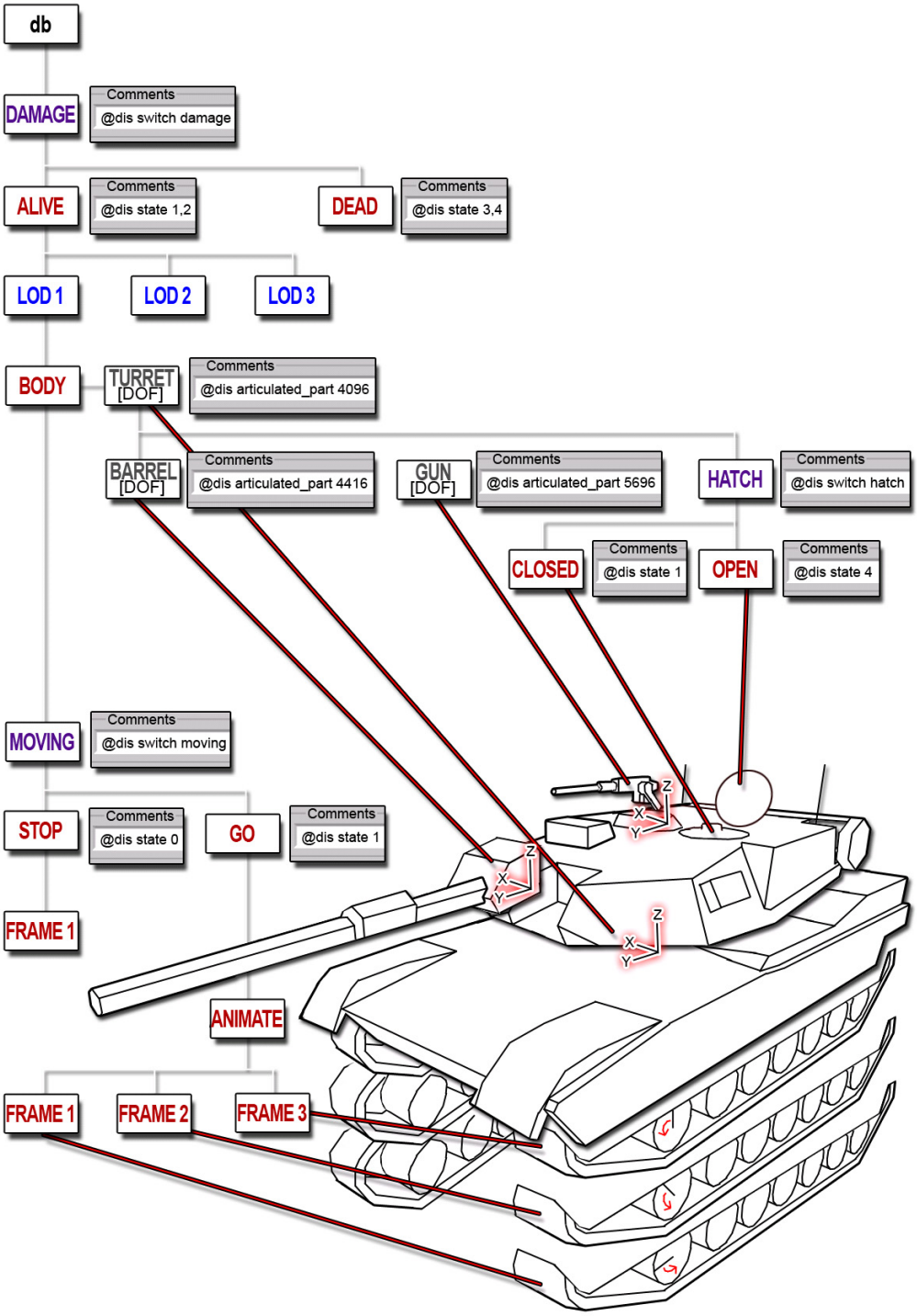


Figure 61-3. 3D model structure

61.6. 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.

61.7. DIS Keyword Values

Table 61-2 lists the values that you can use for the `@dis switch` keyword. 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.)

Table 61-2: DIS Appearance and States (Sheet 1 of 4)

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
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
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	

Table 61-2: DIS Appearance and States (Sheet 2 of 4)

Values	Description	State/ Description	Bit
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
density	Density of environmental.	0 = Clear 1 = Hazy 2 = Dense 3 = Very Dense 4 = Opaque 5 = Unused	
fire_power	Characteristics of fire-power kill.	0 = No 1 = Yes	2
flaming	State of flames rising from an entity.	0 = No 1 = Yes	15
flashlight	State of lights.	0 = Off 1 = On	
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
head_lights	State of headlights.	0 = No 1 = Yes	12
health	State of injury.	0 = None 1 = Slight 2 = Moderate 3 = Fatal	

Table 61-2: DIS Appearance and States (Sheet 3 of 4)

Values	Description	State/ Description	Bit
interior_lights	State of interior lights.	0 = Off 1 = On	29
landing_lights	State of landing lights.	0 = Off 1 = On	12
launch_flash	Describes the presence of a guided munitions launch flash.	0 = No 1 = Yes	
launcher	State of the platform's primary launcher.	0 = Not Raised 1 = Raised	16
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	
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
ramp	State of a ramp.	0 = Down 1 = Up	25

Table 61-2: DIS Appearance and States (Sheet 4 of 4)

Values	Description	State/ Description	Bit
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
state	State of an entity.	0 = Active 1 = Deactivated	23
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
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 61-3 lists the supported values for the `@dis articulated_part` keyword. String values are part of the WRM specification, but are not currently supported.

Table 61-3: 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.

Table 61-3: DIS Articulations (Sheet 2 of 5)

Value	String	Description
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.
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.

Table 61-3: DIS Articulations (Sheet 3 of 5)

Value	String	Description
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.
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.

Table 61-3: DIS Articulations (Sheet 4 of 5)

Value	String	Description
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.
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.

Table 61-3: DIS Articulations (Sheet 5 of 5)

Value	String	Description
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.
3104	tail_hook	Tail hook on a plane for aircraft carrier landings.
7584	wing_fold	Fold on plane's wings.



62. Model Tutorials

Although the individual procedures for managing model definitions and model mappings are fairly straightforward, understanding how they fit together to help you achieve your visualization goals is not always obvious. This chapter contains several tutorials that demonstrate how to combine various procedures to add models.

Adding a Model to VR-Forces 62-2
Create a Model Definition 62-2
Add an Element Definition 62-5
Add a Model Mapping 62-8
Test the Model Mapping 62-10

62.1. Adding a Model to VR-Forces

This tutorial shows how to add a new model to VR-Forces. To help you with the tutorial, VR-Forces includes a model of an F/A-18 Hornet that is not mapped to any entities.

> You can also add new models in the Simulation Object Editor. Using the Simulation Object Editor is the preferred way to add new models.

The basic steps are:

1. Create a model definition ("Create a Model Definition," on page 62-2).
2. Add an element definition ("Add an Element Definition," on page 62-5).
3. Map the new model to a simulation object or object type enumeration ("Add a Model Mapping," on page 62-8).
4. Test the new model ("Test the Model Mapping," on page 62-10).

62.1.1. Create a Model Definition

A model definition is an instance of a schema. It specifies values for the parameters required by the schema.

To create the model definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 62-1).

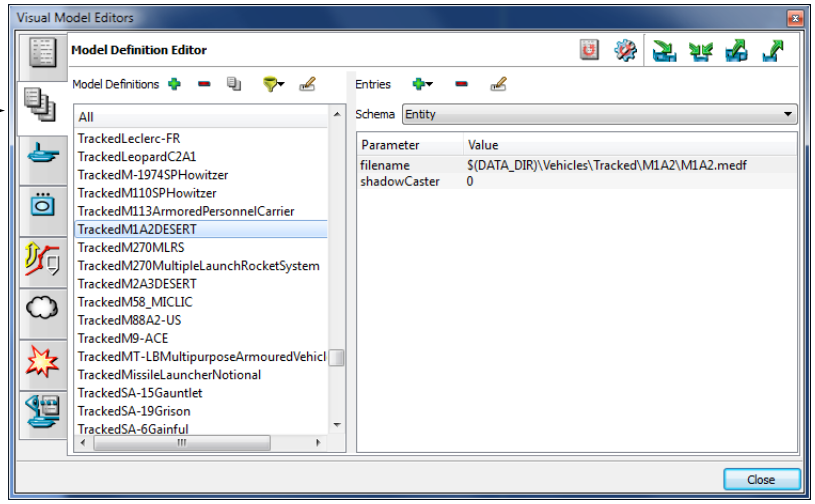


Figure 62-1. Model Definition Editor page

3. Optionally, filter the model definitions in the list to just show entity model definitions. (For details, please see "Filtering the List of Model Definitions," on page 57-11.)
4. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens (Figure 62-2).

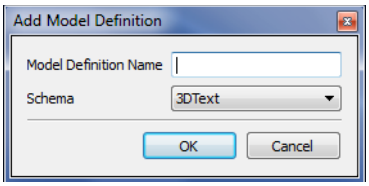


Figure 62-2. Add Model Definition dialog box

5. Type a name for the new model definition, for example, myF18.
6. If you filtered the model definition list, the Schema box says Entity. If not, in the Schema list, select Entity. The dialog box now looks like Figure 62-3.

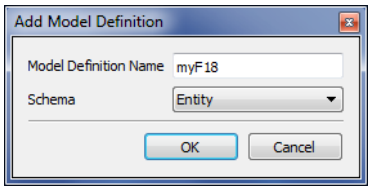


Figure 62-3. New model definition

7. Click OK. The new definition is added to the list (Figure 62-4). It does not have any parameters specified yet, so VR-Forces prints an error message. Now you must specify the model file for this model definition.

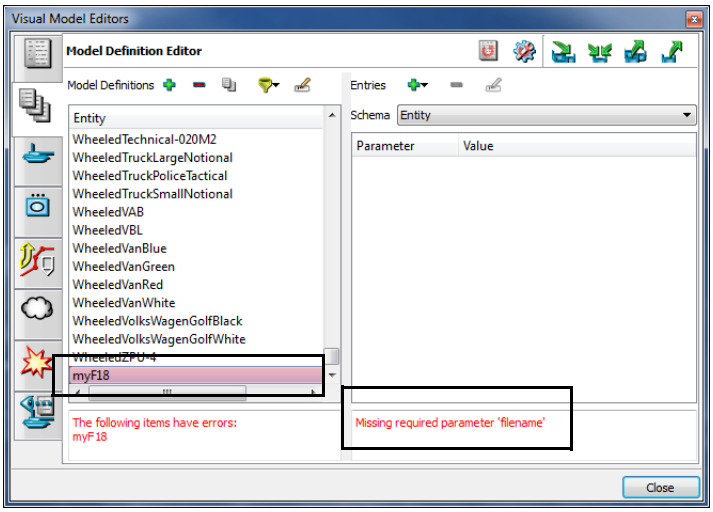


Figure 62-4. New model definition in list

8. Select the new model definition.
9. Click the Add Parameter button (). A list of parameters is displayed (Figure 62-5).

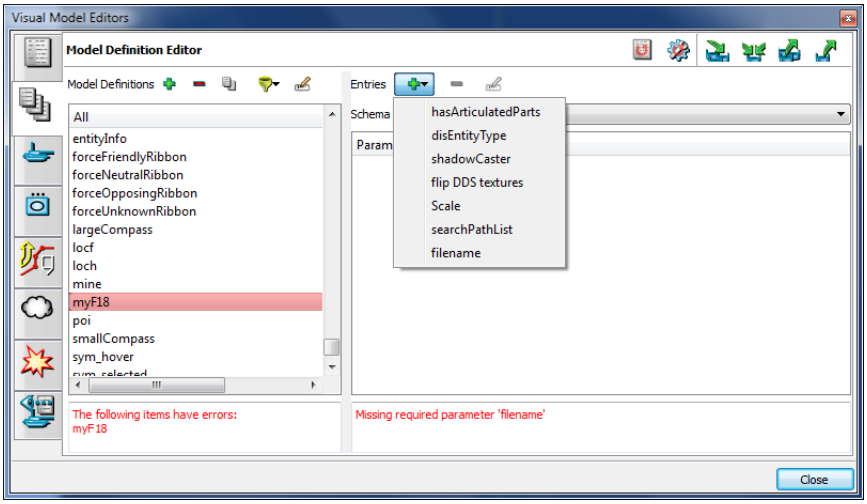


Figure 62-5. List of parameters

10. Select **filename**. It is added to the parameter list.
11. Click the value for the parameter. A file selection dialog box opens.
12. Navigate to *./data/Vehicles/FixedWing/FA-18_Hornet* and select *FA-18_Hornet.flt*.
13. Click OK. The value is updated (Figure 62-6).

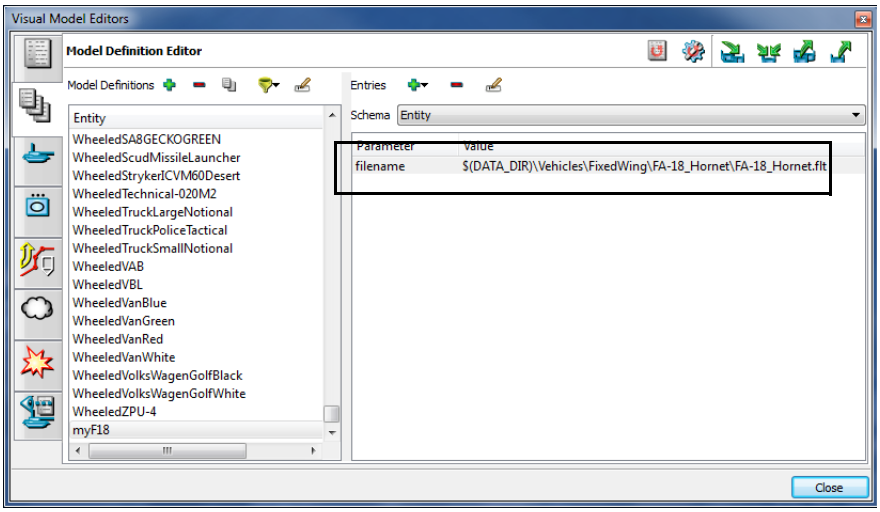


Figure 62-6. Model definition parameter

62.1.2. Add an Element Definition

An element definition configures the visualizers used to display an element and its effects. You must specify the model definition to use for the Model Visualizer.

To add an element definition:



1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 62-7).

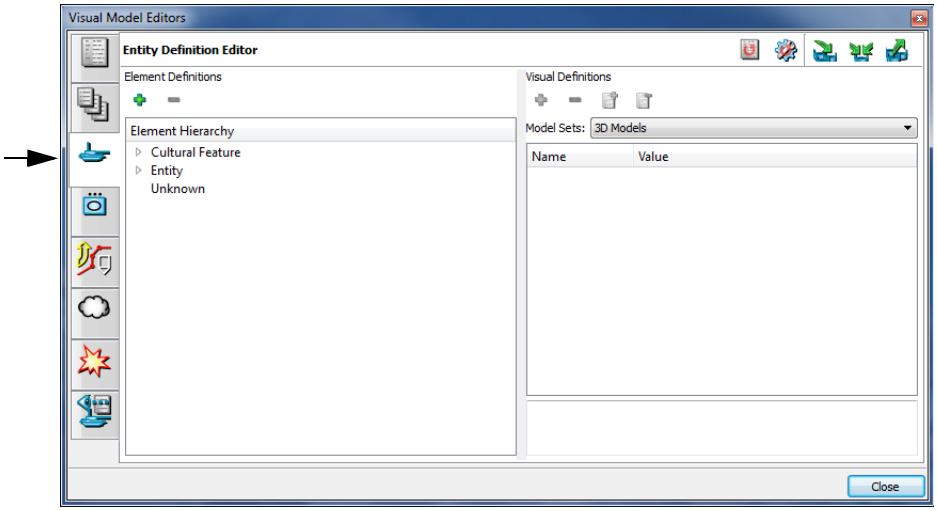


Figure 62-7. Entity Definition Editor page

3. Expand the Entity section of the hierarchy and then expand the Air section.
4. Select Fixed Wing.
5. Select Medium Fixed Wing.
6. Click the Add button (+) next to the Entity Definitions label. The Create Element Definition dialog box opens.
7. Type a name for the new entity, for example, myF18.
8. Click OK. An entry called myF18 is added to the list of fixed wing entities (Figure 62-8). It has the same list of visualizers as the fixed wing parent level.

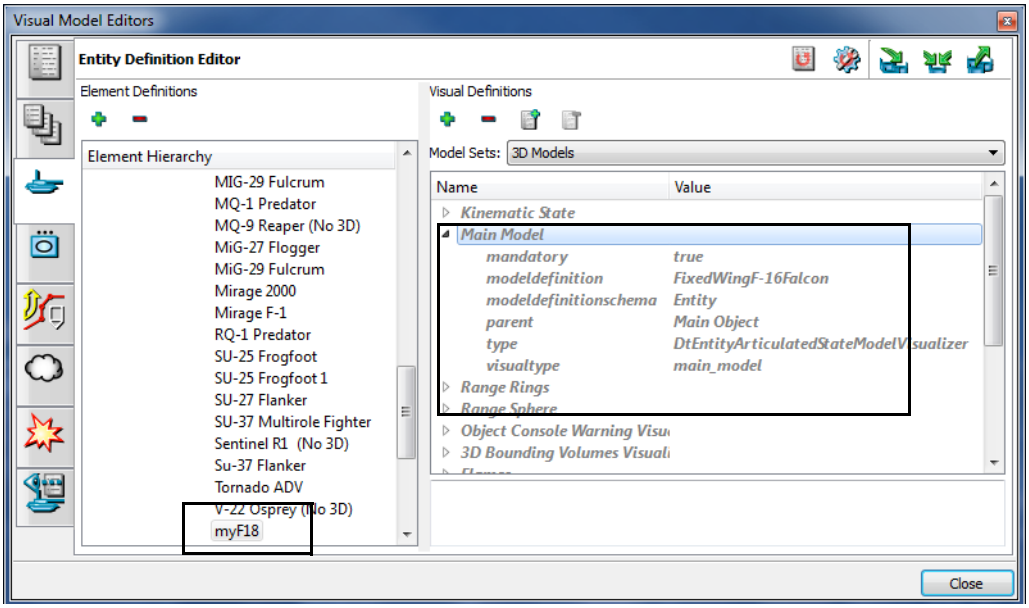


Figure 62-8. New entity definition

9. Scroll through the Visual Definitions list until you find the Main Model visualizer. This visualizer specifies the model definition that this element should use. It has the model definition of the parent element. We need to change it to use the myF18 model definition.
10. Click the Value column for the **modelDefinition** attribute to make it editable. The Choose Model Definitions dialog box opens (Figure 62-9). It shows the current model definition as a selection in a list.

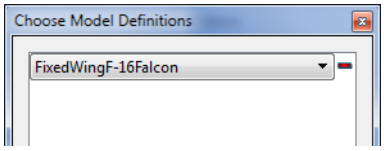


Figure 62-9. Choose Model Definitions dialog box with list of definitions

11. Click the Down Arrow to expand the model definition list and select the new model definition (myF18).
12. Click OK. The model definition is now specified (Figure 62-10).

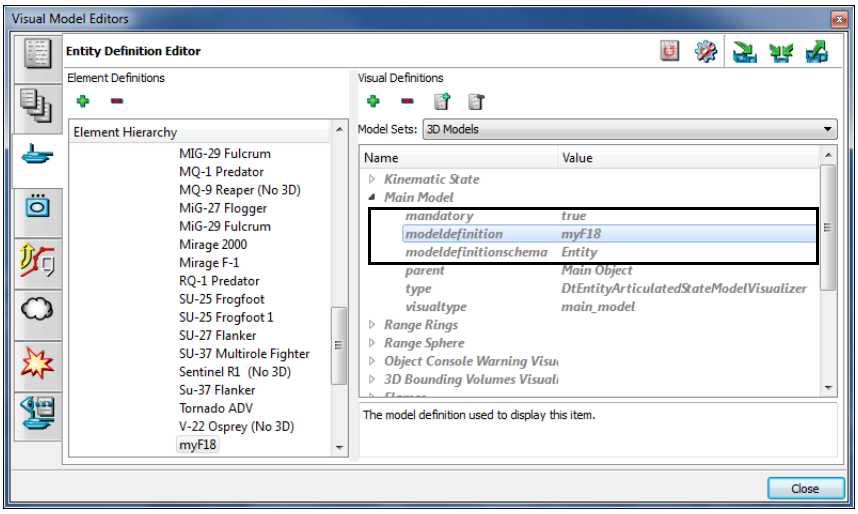


Figure 62-10. Model Definition for Model Visualizer

13. Close the Visual Model Editors dialog box.

62.1.3. Add a Model Mapping

Now, you have to map the new model definition to an object type enumeration.

To add a model mapping:

1. Choose **Settings** → **Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 62-11).

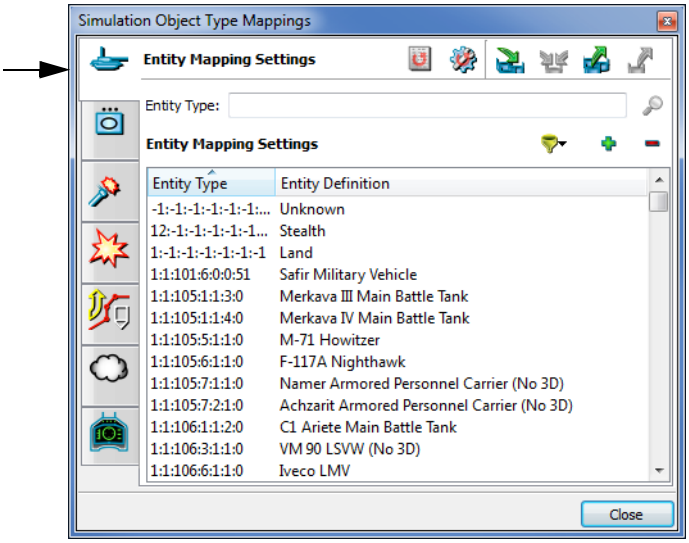


Figure 62-11. Entity Mapping Settings page

3. Click the Filter button ( ).
4. In the list, select **Entity** → **Air** → **Fixed Wing** → **Medium Fixed Wing**. (This does not change the Entity Mapping Settings list; it filters the list in the Create New Entity Type Model Mapping dialog box.)
5. Click the Add button ( ). The Create New Entity Type Model Mapping dialog box opens (Figure 62-12). It lists a default entity type enumeration.

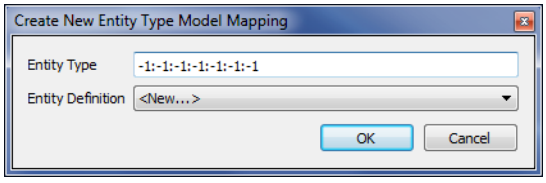


Figure 62-12. Create New Entity Type Model Mapping dialog box

6. In the Entity Type text box, type the enumeration you want to map to the new model mapping. (The enumeration for an F/A18 is 1:2:225:1:9:1:-1.)

VR-Forces already has an F/A18 mapped to this enumeration. However you can map multiple model definitions to an enumeration. When an object type that has multiple model definitions mapped to it is discovered in a simulation, VR-Forces randomly assigns a model definition from among those mapped to the object type.

7. In the Entity Definition list, select the model definition that you added in the previous procedure (myF18) (Figure 62-13).

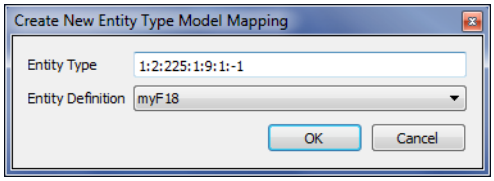


Figure 62-13. Create new model mapping

8. Click OK.
9. The new model mapping is added to the list.

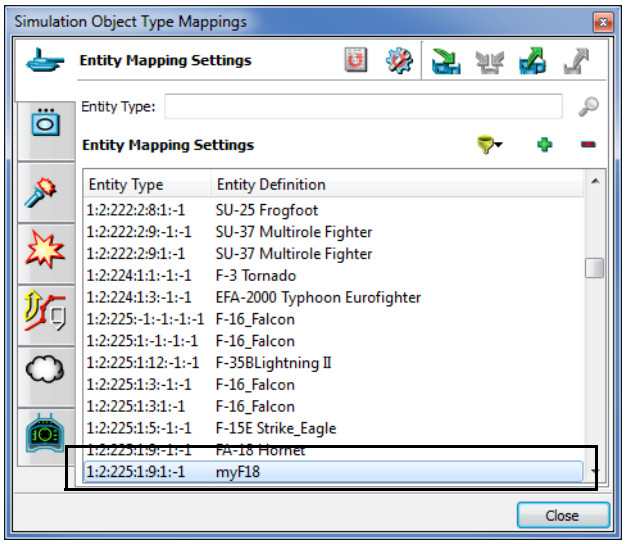


Figure 62-14. New model mapping

10. Restart VR-Forces. Changes to model definitions and mappings do not take effect until you restart.

62.1.4. Test the Model Mapping

It is a good idea to test the new model mapping to see if the correct model is being displayed. You can test a model by running a simulation that has the entity or object type, or you can perform the simple test described here.

To test a model:



1. Open a terrain database that has easily identifiable props. For the purposes of this tutorial, open the Makland terrain.
2. In the VR-Forces window, move the observer so that you are viewing the palace on top of the hill near the town in Makland.
3. Click the palace to select it (Figure 62-15).

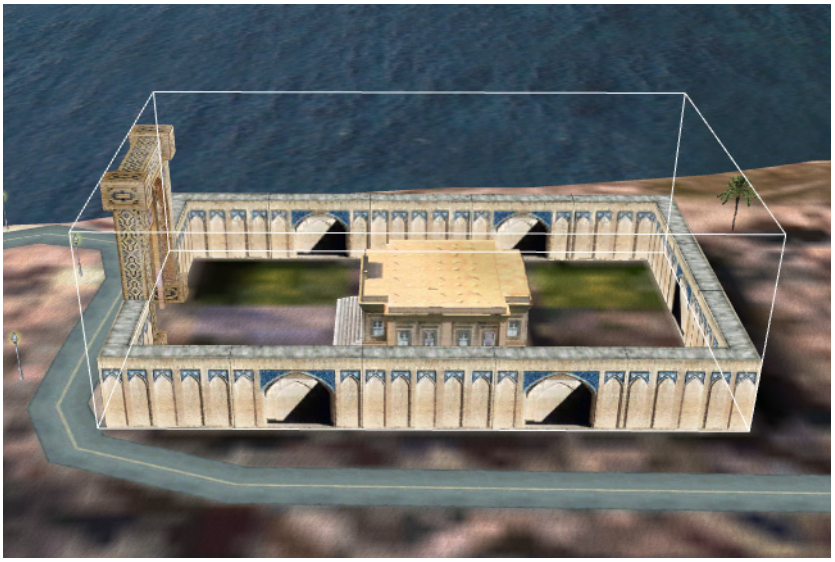


Figure 62-15. Palace

4. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
5. Select the Edit Existing Props page. The list of props shows that Building26, the palace, is selected (Figure 62-16).

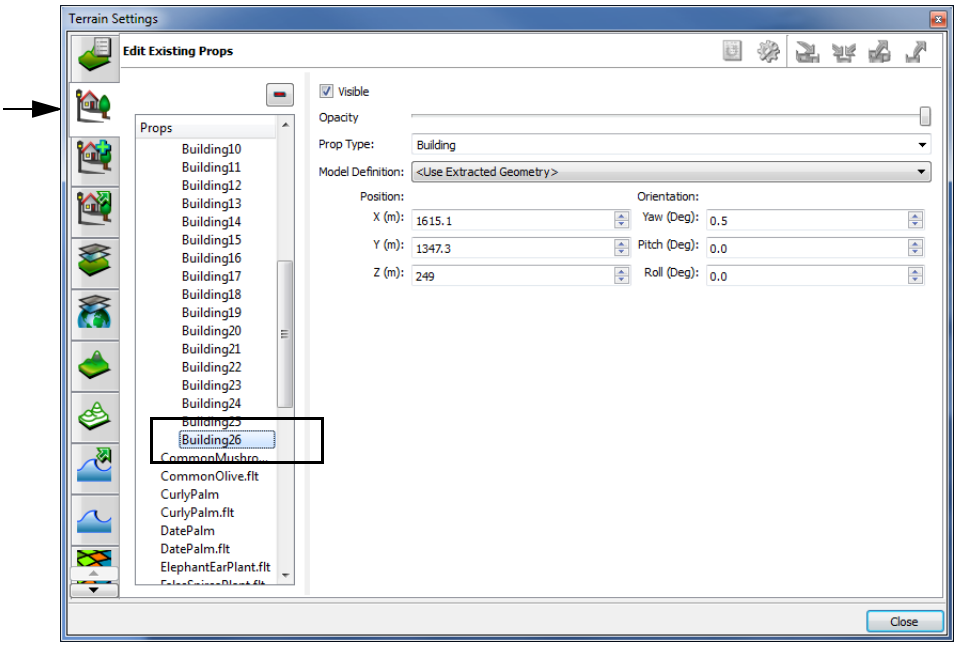


Figure 62-16. Edit Existing Props page

6. In the Model Definition list, select the model definition that you added earlier in this tutorial (Figure 62-17). The model of the palace is replaced with the model for the F/A-18 (Figure 62-18). If this is the model that you wanted to add, then you know that you completed the procedure correctly. If this is not the model, you expected, check the filename you specified for the model definition.

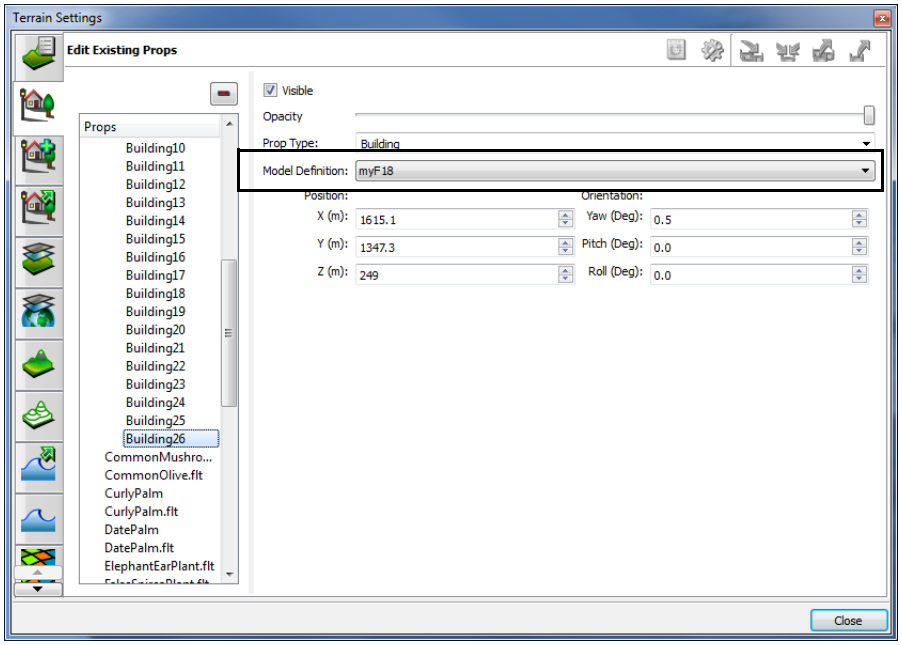


Figure 62-17. Changed model definition

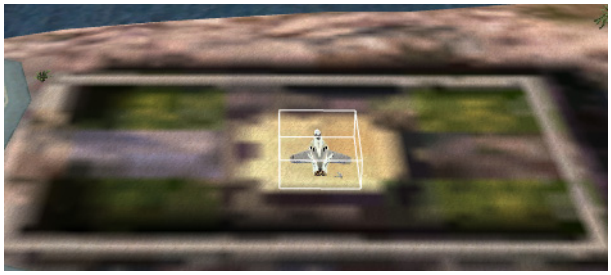


Figure 62-18. F/A-18 replaces the palace

7. If you are going to use this terrain for simulations, change the model definition back to Use Extended Geometry.



XII. Creating and Editing Simulation Models

This section explains how to use the Simulation Object Editor and OPD 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.



63. 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 63-2
Starting the Simulation Object Editor 63-3
Simulation Model Sets 63-5
- Including Simulation Model Sets in other Simulation Model Sets 63-6
- SMS Priority for Included and Multiple SMSs 63-8
- Lua Scripts in Included SMSs 63-9
- Application Level Files in Included SMSs 63-9
- Search Paths for Referenced Files in an SMS 63-10
- Promoting a Model to the Loaded SMS 63-11

Migrating an SMS to a New Release 63-12
- Upgrading Old Simulation Model Sets 63-13

Opening a Simulation Model Set 63-15
Creating a New Simulation Model Set 63-16
- Creating a Completely New Simulation Model Set 63-17
- Specifying the Default Simulation Model Set for Scenarios 63-17

Applying Platform Updates 63-17

63.1. 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.



The Simulation Object Editor primarily lets you edit properties that affect simulation. Entities also have visualization models, which are configured in the front-end in the Visual Model Editors and Entity 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 Chapter 58, *Mapping Models and Effects* and Chapter 57, *Model and Element Definitions*.

The Simulation Object Editor does not let you add or remove individual components or edit most of the more detailed entity parameters. For these advanced editing tasks, you can use the OPD Editor, which is described in Chapter 69, *Using the OPD Editor*, or edit specific configuration files, which are described throughout this manual.

If you want to add new simulation objects or customize the simulation objects supplied with VR-Forces, start with the Simulation Object Editor. Use the OPD Editor or a text editor only if the Simulation Object Editor does not let you make the changes you need to make.

63.2. Starting the Simulation Object Editor

➤ To start the Simulation Object Editor in Windows, choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.4.1** → **Tools** → **Simulation Object Editor**. The Simulation Object Editor opens (Figure 63-1).

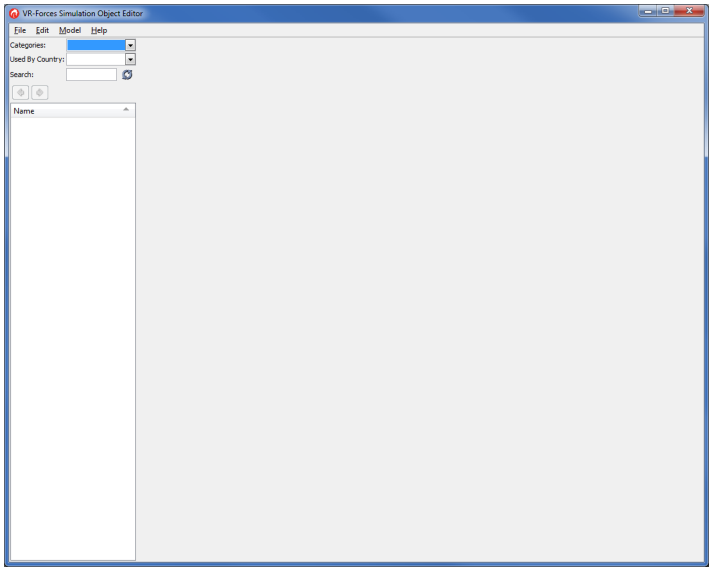


Figure 63-1. Simulation Object Editor

To start the Simulation Object Editor from the command-line in Windows or Linux:

1. Change directory to *./bin64*.
2. Run:

```
simulationObjectEditor options
```

Table 63-1 describes the command-line arguments for the Simulation Object Editor.

Table 63-1: 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.
`--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.
`--opdEditorLocation` *string*	Location of the OPD Editor application.
`--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.

63.3. 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 15-9.

When you use the Simulation Object Editor, you are editing a simulation model set. We 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 based on one of the provided SMSs 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 a New Simulation Model Set," on page 63-16.

63.3.1. 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.

To include an SMS in another SMS:

1. Choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.4.1** → **Tools** → **Simulation Object Editor**. The Simulation Object Editor opens (Figure 63-1).
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 **Edit** → **Included Simulation Model Sets**. The Edit Included Simulation Model Sets dialog box opens.

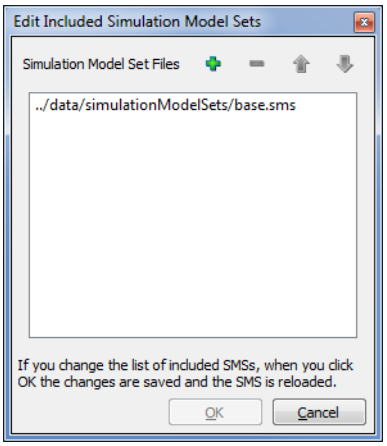


Figure 63-2. Edit Included Simulation Model Sets dialog box

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.

63.3.2. 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 63-3.

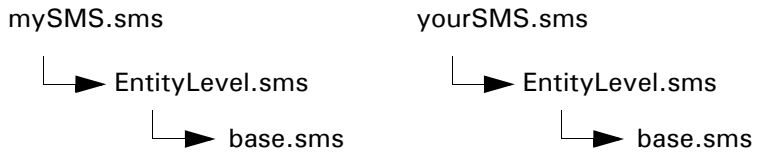


Figure 63-3. Multiple SMSs

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. 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*. If *yourSMS.sms* has objects that conflict with *mySMS.sms*, then *yourSMS.sms* overrides them.

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")
```

63.3.3. 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 will use 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.

63.3.4. 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*.

63.3.5. 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, damage tables, ammunition selection tables 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 63-3, 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*.

63.3.6. 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 63-4 shows the Space category for *addEntity.sms*. (It is in the developer toolkit examples for the addEntity example.) The Space Shuttle entity that is part of the parent SMS is editable, but the two entities included from *EntityLevel.sms* are not. Note in the figure that most of the parameters for the Space Shuttle in *EntityLevel.sms* are read-only.

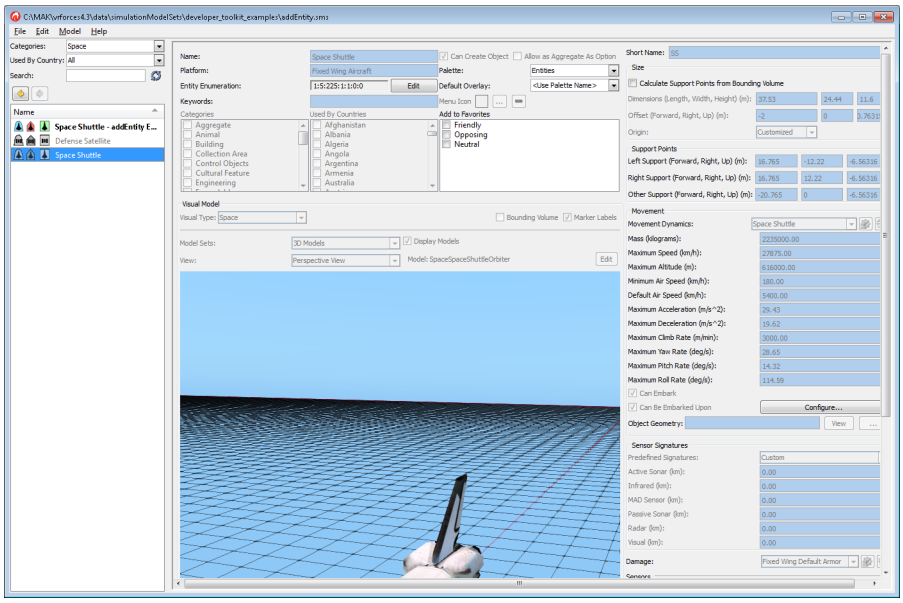


Figure 63-4. addEntity.sms

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. Select the object you want to promote.
3. Choose **Model** → **Promote to Main Simulation Model Set**. 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.

63.4. Migrating an SMS to a New Release

The Simulation Object Editor includes an option for upgrading old SMSs. You can also update them by hand.

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. It is possible that you may add or edit some of the files that entity systems reference, such as damage files. You might also create or modify scripts that are stored in an SMS.

If you upgrade to a new version of VR-Forces, 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 VR-Forces and then import them into the new release.

Over time, the SMS format has changed. The Simulation Object Editor has a process for upgrading older SMSs. For details, please see "Upgrading Old Simulation Model Sets," on page 63-13.

63.4.1. Upgrading Old Simulation Model Sets

The SMS organization changed considerably from VR-Forces 4.1.x to 4.2. At that time, SMSs started using the *.entity* file to configure individual simulation objects. In VR-Forces 4.3, the ability to include SMSs in other SMSs was added. The SMS upgrade tool helps you migrate SMSs through these different transitions.

If you need to migrate a pre-VR-Forces 4.2 SMS, the upgrade tool converts your old SMS into the XML file format used for VR-Forces 4.2 and later. If you upgrade an SMS from VR-Forces 4.2 to 4.3 or later, it takes into account the transition from the *default.sms* to *EntityLevel.sms*, which includes *base.sms*.

When you upgrade a 4.2 or later SMS, you create a completely new SMS. The upgrade tool:

- Copies all files except *.entity* and *.sog* files from a current SMS (that is, an SMS from the latest version of VR-Forces) into the new SMS. (Optionally, you can copy all *.entity* and *.sog* files to the new SMS. The tool does not copy all *.entity* files by default because users often create SMSs with new simulation objects or a small subset of the simulation objects in the SMSs provided with VR-Forces. The default behavior is to create a new SMS that only has the simulation objects in the SMS being upgraded.)
- Copies all of the files from the old SMS, including *.entity* files, but not *.opd* or *.mst* files, into the new SMS, if necessary. (Files that are part of an included SMS are not copied.) If *.entity* files in the old SMS match the names of files in the current SMS, the old files overwrite the current files.
- If the following file types with the same name have different content, the tool warns you about these differences. You must reconcile them manually.
  - xml
  - lua
  - sysdef
  - mtl
  - tsk
  - ui.

To upgrade an old SMS:

1. Open the Simulation Object Editor.
2. Choose **File** → **Upgrade Old Simulation Model Set**. The Select SMS to Upgrade dialog box opens (Figure 63-5).

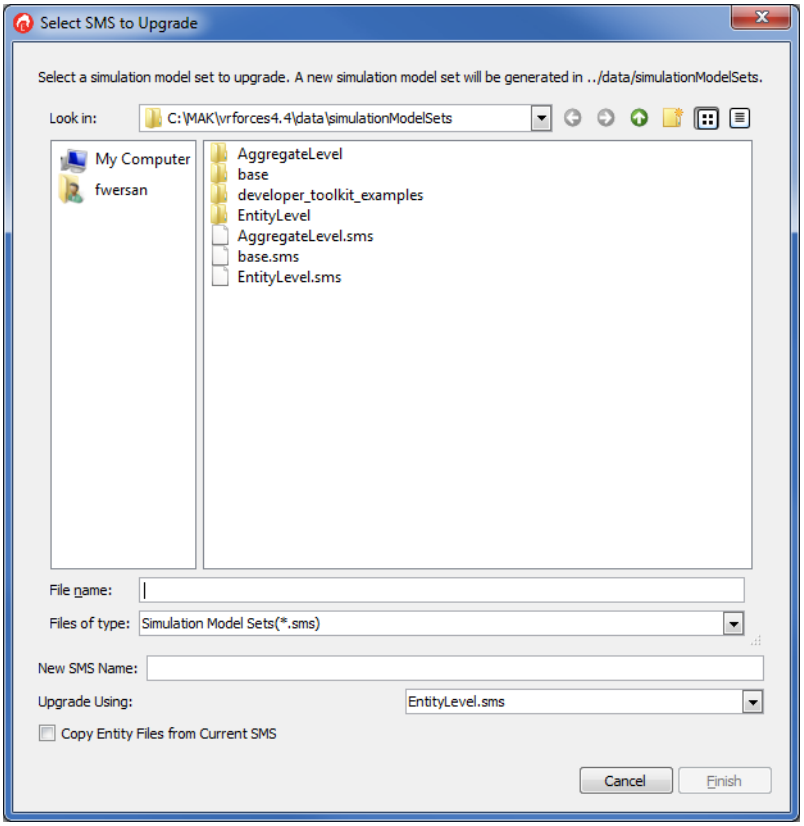


Figure 63-5. Select SMS to Upgrade dialog box

3. Select the SMS you want to upgrade.
4. Specify a name for the new SMS.
5. Specify the SMS you want to base the new SMS on – *EntityLevel.sms* or *AggregateLevel.sms*.
6. Optionally, select Copy Entity Files from Current SMS.
7. Click Finish. The upgrade tool runs.

63.5. 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.

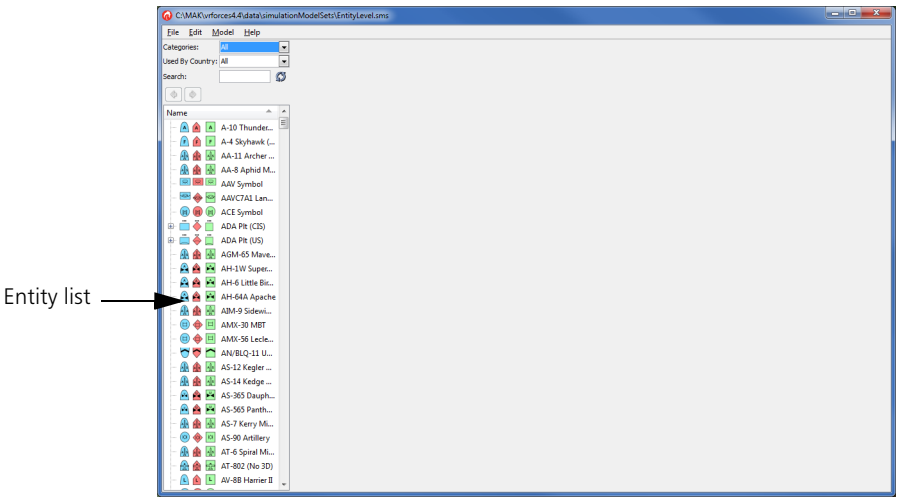


Figure 63-6. Simulation Object Editor with simulation model set loaded

63.6. Creating a New Simulation Model Set

You can create new SMSs by copying and editing an existing SMS or you can create a completely new SMS and create the simulation objects that you want to include in it. When you create a new SMS, you can specify that it include other SMSs.

To create a new simulation model set from an existing SMS:

1. Choose **File** → **New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 63-7).

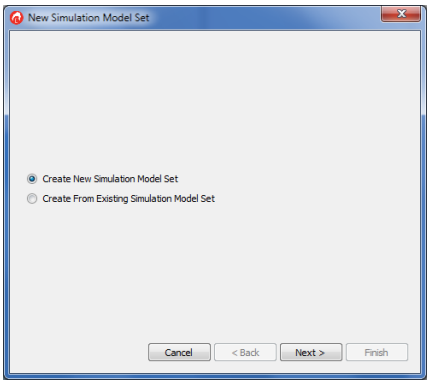


Figure 63-7. New Simulation Model Set dialog box

2. Select Create From 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.
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 63-6, 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**.

63.6.1. Creating a Completely New Simulation Model Set

To create a completely new simulation model set:

1. Choose **File** → **New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 63-7).
2. Select Create New Simulation Model Set.
3. Click Next. The New Model Set Name dialog box opens.
4. In the File name box, type a name for the new model set.
5. Click Finish. The Simulation Object Editor loads systems information from the default model set.
6. Create the simulation objects that you want to have in the model set, as described in "Creating a New Entity," on page 64-31.
7. Choose **File** → **Save Simulation Model Set**.

63.6.2. Specifying the Default Simulation Model Set for Scenarios

When you create a new scenario, the New Scenario dialog box lists *EntityLevel.sms* as the SMS to use. You can change the default SMS. For details, please see "Specifying the Default SMS Configuration," on page 7-11.

63.7. Applying Platform Updates

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 68-21, that would be a tedious and error-prone activity. The Simulation Object Editor can apply all updates to all files automatically.

To apply platform updates:

1. Choose **File** → **Apply Platform Updates**. An information prompt opens.
2. Click OK.



64. Editing Simulation Object Models

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 65, *Editing Units for Entity-Level Scenarios* and Chapter 66, *Editing Aggregate-Level Objects*.

Editing Generic Object Parameters 64-3
Selecting an Object to Edit 64-3
Editing an Object's Categories 64-3
Changing an Object's Object Type Enumeration 64-4
Changing an Object's Name 64-6
Specifying Whether an Object Can Be Created 64-6
Specifying the Object Creation Palette 64-7
Specifying the Default Overlay 64-7
Undoing (Reverting) Changes to an Object 64-8
Editing Parameters that are Specific to Simulation Objects 64-8
Editing a Simulation Object's Used By Countries List 64-8
Editing a Simulation Object's Bounding Volume (Size) 64-9
Displaying a Simulation Object's Bounding Volume 64-10
Specifying Support Points 64-11
Excluding Entities from Batch Bounding Volume and Support Point Updates 64-11
Editing Behavioral Parameters 64-11
Adding Object Geometry to an Entity 64-13
Editing a Lifeform's DI-Guy Character, Appearance, and Animation. 64-14
Editing a Simulation Object's Visual Model 64-16
Changing the 3D Model or Colorized Model 64-17
Changing the 2D Icon Using a Font 64-18

Changing the 2D Icon Using an Image 64-19
Adding a New Visual Model or Image 64-20
Displaying Details about the Visual Model 64-21
Editing a Simulation Object's Systems 64-22
Adding a System 64-23
Selecting a Damage Model 64-24
Selecting a Movement Model 64-24
Editing a System's Properties 64-24
Editing a System in the OPD Editor 64-25
Removing a System 64-25
Configuring Scripted Entity Movement 64-26
Configuring Ballistic Missile Movement 64-27
Configuring Animated Movement 64-28
Editing Movement File Configurations 64-30
Creating a New Entity 64-31
Creating a New Entity from an Existing Entity 64-32
Configuring Embarkation 64-33
Adding Ingress and Egress Points 64-35
Configuring Slots 64-38
Creating a Simulation Object Group 64-41
Setting the Simulation Connection when Creating an SOG 64-42
Deleting an Object Model 64-42
Navigating Among Edited Simulation Objects 64-42
Editing Tactical Graphics 64-43
Editing a Tactical Graphic's Menu Icon 64-43
Setting the Default Values for Tactical Graphics Properties 64-44
Specifying the Properties that can be Edited in the Front-End 64-45
Editing the Visual Model for a Tactical Graphic 64-46
Editing the Category List 64-48
Adding a New Category 64-48
Removing a Category 64-49
Renaming a Category 64-49
Undoing Category Changes 64-49
Managing the Forces List 64-50
Adding a Force 64-50
Editing a Force 64-51
Removing a Force 64-52
Importing Entity Files 64-52

64.1. 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.



If one of the object parameters described in this section does not apply to a particular object type, it is either blank or uneditable in the Simulation Object Editor.

64.1.1. 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 category for the type of object you want to edit.
2. Select the object in the list.

64.1.2. Editing an Object's Categories

Objects are organized by categories on the Create menu and the object creation palettes. You can change the categories that an object is part of. For details about adding new categories to the list, please see "Adding a New Category," on page 64-48.

➤ To assign an object to a category, in the Categories list, select the check boxes for the categories you want it to be part of and clear the check boxes for the categories that you do not want (Figure 64-1).

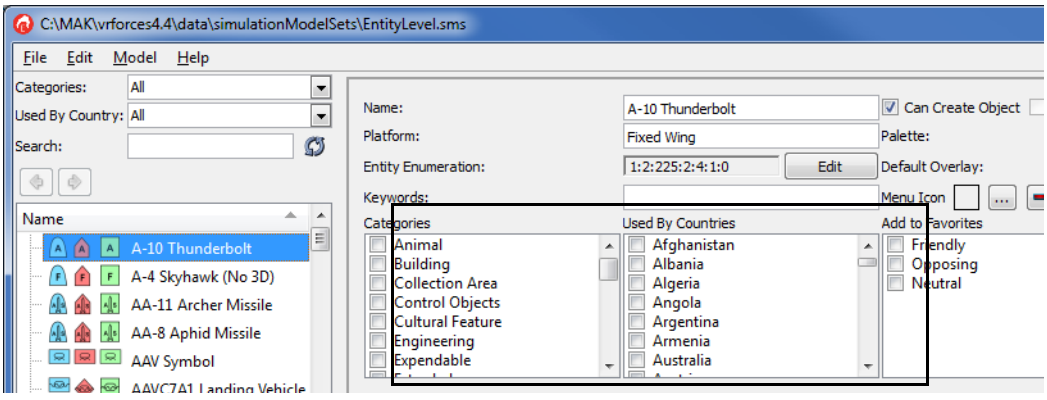


Figure 64-1. Categories list

64.1.3. 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 68-6.) 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. Click Edit next to the Entity Enumeration text box (Figure 64-2). The Change Enumeration dialog box opens (Figure 64-3).

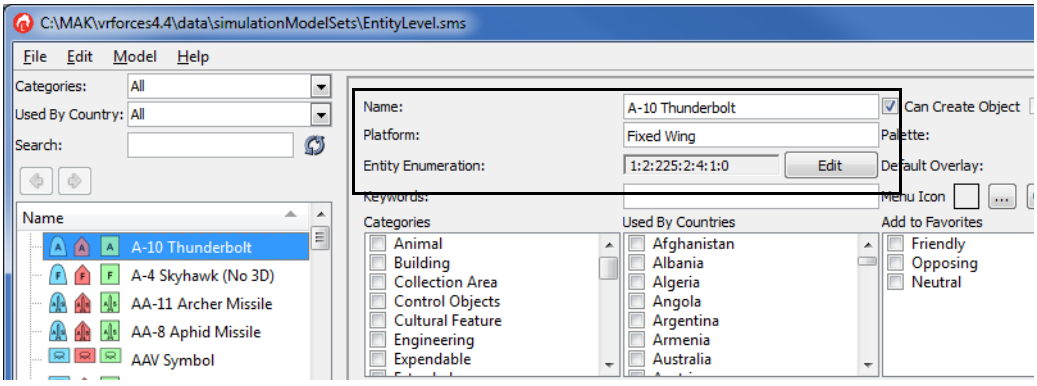


Figure 64-2. Object enumeration

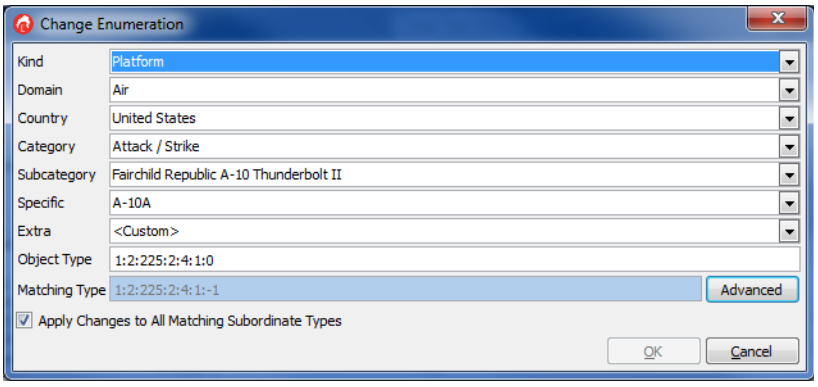


Figure 64-3. Change Enumeration dialog box

3. Edit the enumeration by selecting options from the lists for enumeration elements or type a new enumeration in the Object Type text box.
4. Optionally, specify the Matching Type as follows:
   a. Click Advanced. The dialog box expands (Figure 64-4).

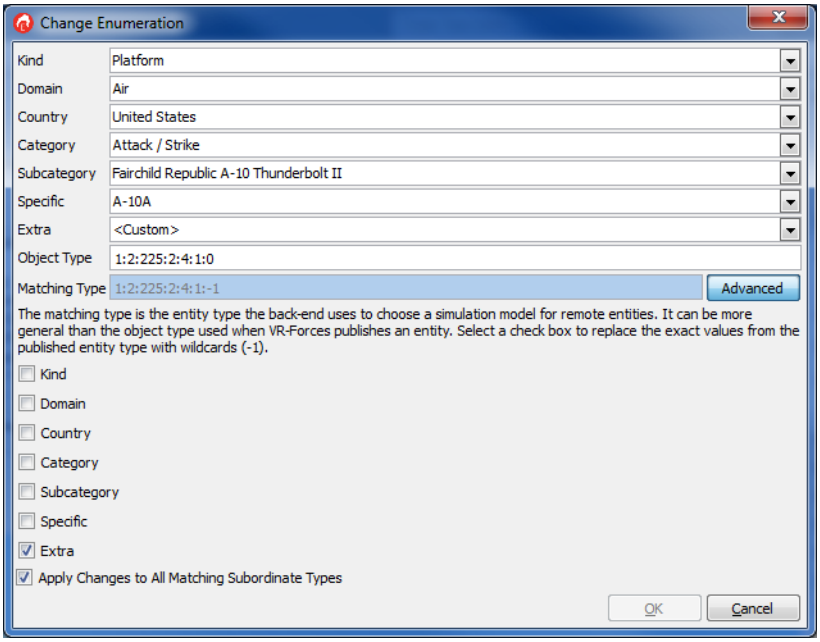


Figure 64-4. Change Enumeration dialog box, Advanced

   b. For each field in the matching type enumeration that you want to be a wild card (-1), select the appropriate check box.

5. 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.
6. Click OK. The enumeration changes.

64.1.4. 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 in the Attributes section of the Simulation Object Editor (Figure 64-8).

To change an object's name:

1. Select the object in the object list.
2. Type a new name in the Name text box (Figure 64-2).

64.1.5. 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, select or clear the Can Create Object check box (Figure 64-5).

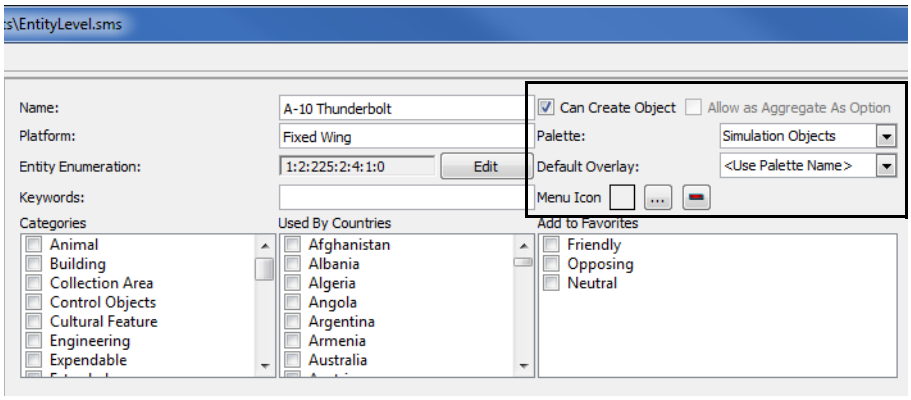


Figure 64-5. Can Create Object check box

64.1.6. 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. In the Palette list (Figure 64-5), select the palette you want to put it on.

64.1.7. 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. Users 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 34-2 and "Specifying an Object's Properties Before You Create It (Click to Locate)," on page 16-13.

To specify the default overlay:

1. Select the object in the objects list.
2. Select an option from the Default Overlay list (Figure 64-5).

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.

64.1.8. 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**.

64.2. Editing Parameters that are Specific to Simulation Objects

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.)

64.2.1. Editing a Simulation Object's Used By Countries List

The country code in a vehicle's entity enumeration specifies the country in which the vehicle is manufactured. It does not indicate where the vehicle is used, and many vehicle types are used in multiple countries. When creating a scenario, it can be convenient if the Simulation Objects Palette only lists the vehicles used by the country for which you are laying down forces. The Used By Countries list in the Simulation Object Editor lets you specify the countries that use a vehicle. Users 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 Used By Countries list works exactly like the Categories list. For details about how to configure the list of countries, please see "Editing the Category List," on page 64-48.

➤ To assign a simulation object to a country of use, in the Used by Countries list (Figure 64-1), select the check boxes for the countries that use it.

64.2.2. 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 group box. You can display a simulation object's bounding volume in the Visual Model group box. 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. Change any of the values in the Size group box (Figure 64-6).

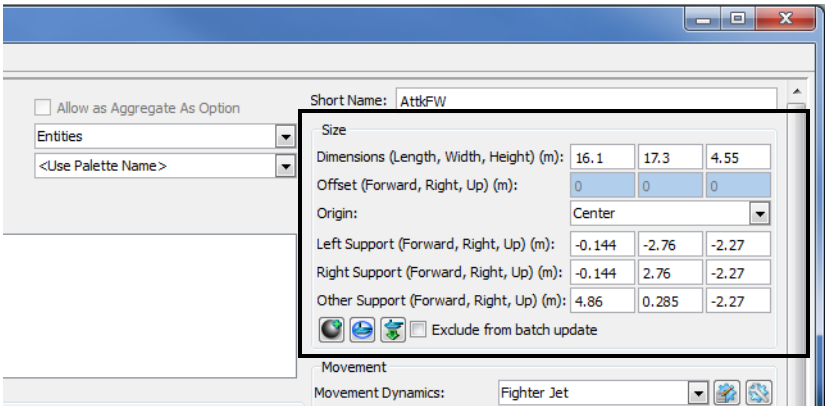


Figure 64-6. Size group box

➤ 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**.

64.2.3. 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 group box, on the Model Sets list, select 3D Models.
3. Select the Bounding Volume check box. The bounding volume is displayed (Figure 64-7).

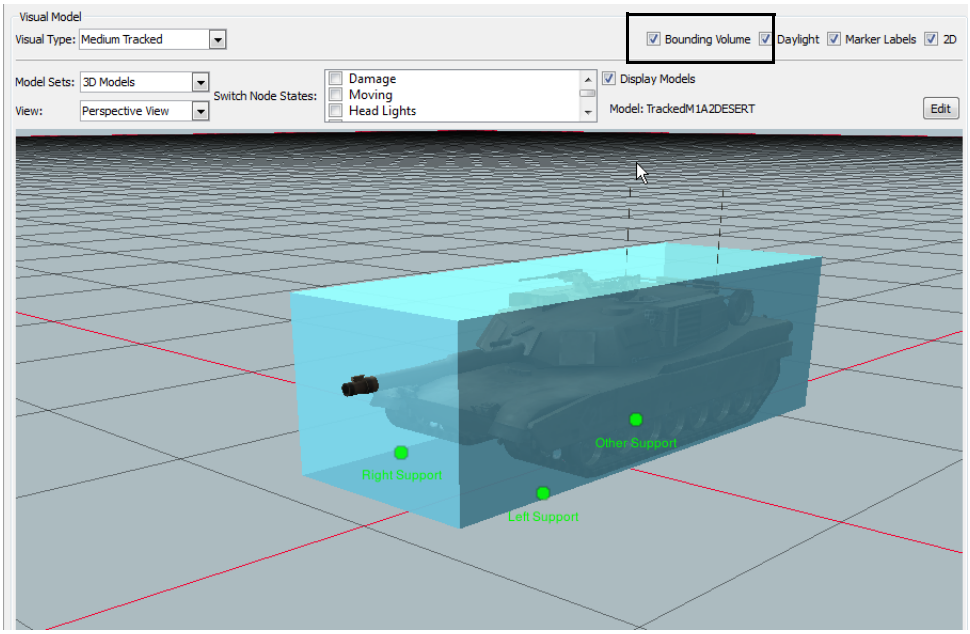


Figure 64-7. Bounding volume

64.2.4. 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, in the Size group box (Figure 64-6), type the desired values.

➤ To calculate support points based on the model, in the Size group box, 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**.

64.2.5. 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. In the Size group box, select the Exclude From Batch Update check box.

64.2.6. Editing Behavioral Parameters

The Simulation Object Editor lets you edit some of the parameters that affect simulation object behavior. For detailed information about these parameters, please see the parameter descriptions in the OPD Editor online help.

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 64-22.

➤ To edit simulation object parameters, edit any of the parameters in the Attributes panel (Figure 64-8). (The panel reconfigures itself as you select different types of simulation objects.)

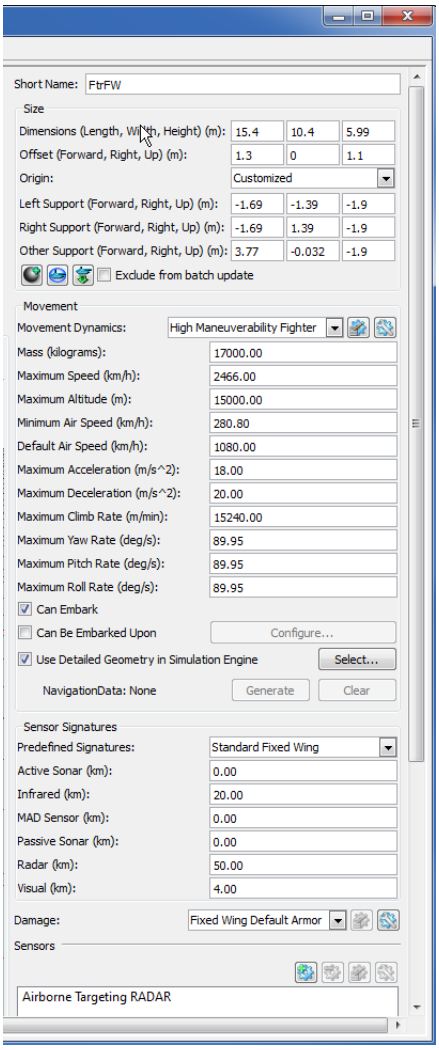


Figure 64-8. Attributes panel

64.2.7. Adding Object Geometry to an Entity

Object geometry is to an entity as terrain is to the earth. It allows embarked entities to move about on the entity on which they are embarked. For example, adding object geometry to an aircraft carrier lets planes taxi on the deck of the carrier. You can add object geometry to entities 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 53-8.



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 an entity:

1. Select the entity in the Object List.
2. In the Movement group box, select Use Detailed Geometry in Simulation Engine.
3. Click Select. The Select Object Geometry dialog box opens.
4. Click the Browse button. The Object Geometry dialog box opens.
5. Select the file that contains the object geometry for this entity.
6. Click Open.
7. Click OK.

64.2.8. Editing a Lifeform's DI-Guy Character, Appearance, and Animation

VR-Forces can specify the character, appearance, 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 categories).

To specify the default DI-Guy characteristics for a lifeform:

1. In the Categories list, select the Human or Animal category.
2. Select a lifeform in the object list.
3. In the Visual Model group box, on the Model Sets list, if 3D Models is not selected, select it.
4. In the Visual Model group box, click Edit. The Modify Visuals dialog box opens (Figure 64-9). The Use DI-Guy option should be selected. If it is not selected, select it.

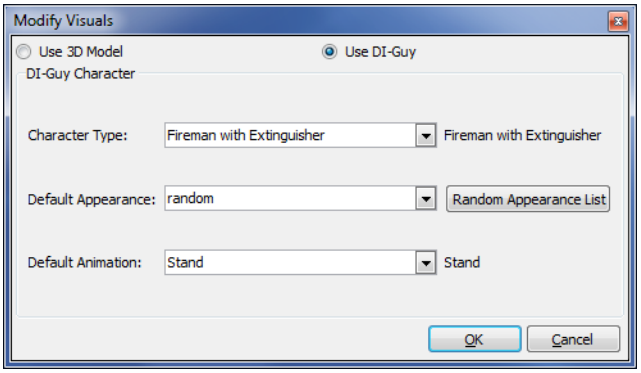


Figure 64-9. DI-Guy Character group box

5. Select the DI-Guy character you want to use from the Character Type list.
6. Select the default appearance from the Default Appearance list. One of the default options is to use a random appearance.
7. If you selected random appearance, optionally, specify the list of appearances to choose from. For details, please see "Configuring the Random Appearance List," on page 64-15.
8. 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 Appearance List

If you specify that a lifeform use a random appearance for DI-Guy models, then VR-Forces chooses from a list of appearances when it specifies the appearance to use. By default, the list is all appearances available for the chosen character. However some appearances might not be appropriate for a scenario, so you can specify a subset of appearances to use.

To configure the random appearance list:

1. In the Categories list, select the Human category (or Animal category).
2. Select a lifeform in the object list.
3. In the Visual Model group box, on the Model Sets list, if 3D Models is not selected, select it.
4. In the Visual Model group box, click Edit. The Modify Visuals dialog box opens (Figure 64-9). The Use DI-Guy option should be selected. If it is not selected, select it.
5. Click Random Appearance List. The Random Appearances List dialog box opens (Figure 64-10). The appearances available for the character type are listed.

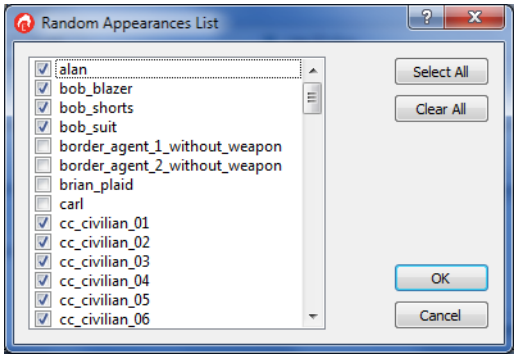


Figure 64-10. Random Appearances List dialog box

6. Select the appearances that you want to include in the random appearance list.
7. Clear any appearances that you do not want to include in the random appearance list.
8. Click OK.
9. Click OK to close the Modify Visuals dialog box.

64.3. 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 Chapter 57, *Model and Element Definitions*.) 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.

! Image and icon mapping files are global, not SMS-specific. Therefore if you change the visual model or icon in one SMS, it affects all other SMSs.

To change the visual model for a simulation object:

1. In the object list, select the object list whose model you want to change. The model for the currently selected model set is displayed in the Visual Model group box.

---

i 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. Optionally, select a visual type in the Visual Type list. This choice 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.
4. Change the 3D model, 2D icon, or 2D image as described in the following sections.

64.3.1. Changing the 3D Model or Colorized Model

To change a 3D model or 3D colorized model:

1. Click Edit. The Modify Visuals dialog box opens (Figure 64-11).
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 64-11). (You can add new models. For details, please see "Adding a New Visual Model or Image," on page 64-20.)

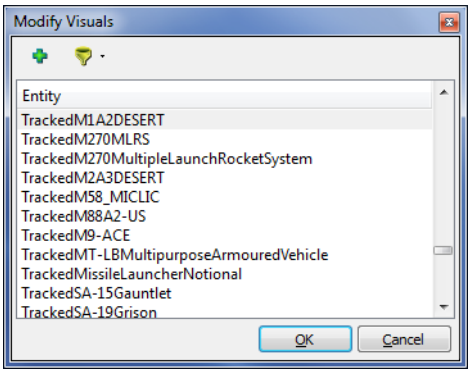


Figure 64-11. Modify Visuals dialog box (3D)

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.

64.3.2. Changing the 2D Icon Using a Font

To change a 2D icon using the MIL-STD font:

1. Click Edit. The Modify Visuals dialog box opens (Figure 64-12).

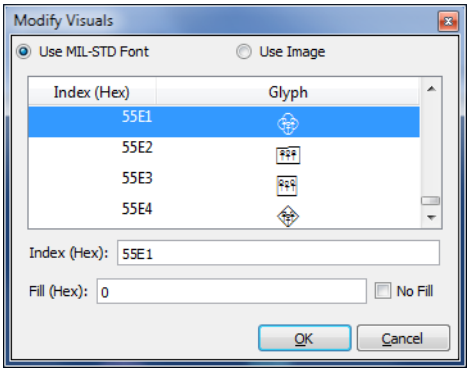


Figure 64-12. Modify Visuals dialog box (2D)

2. Select the Use MIL-STD Font option. The dialog box lists index values and associated glyphs.
3. Select the icon glyph that you want to use. The new icon is displayed in the panel
4. Click OK to close the Modify Visuals dialog box.

64.3.3. Changing the 2D Icon Using an Image

To change a 2D icon using an image:

1. Click Edit. The Modify Visuals dialog box opens (Figure 64-12).
2. Select the Use Image option. The dialog box lists the forces that are configured for this simulation model set (Figure 64-13).

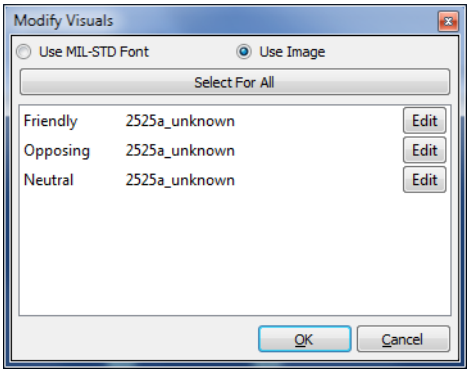


Figure 64-13. Modify Visuals dialog box (2D image)

3. 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 64-14). 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 64-20.)

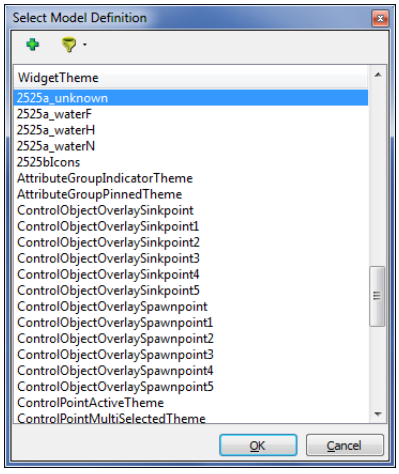


Figure 64-14. Select Model Definition dialog box

4. Select the image that you want to use. The new image is displayed in the panel.
5. Click OK.

6. Click OK to close the Modify Visuals dialog box.

64.3.4. 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 and images to the list. The procedure is essentially the same for both cases.

To add a new model or image to the visual models list:

1. In the Model Sets list, select the model set that you want to edit.
2. For 3D models or 3D colorized models:
   a. Click Edit. The Modify Visuals dialog box opens (Figure 64-11).
   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 Edit. The Modify Visuals dialog box opens (Figure 64-12).
   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 64-13).
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.
4. Click OK.
5. Click OK to close the Modify Visuals dialog box.

64.3.5. Displaying Details about the Visual Model

The Visual Model group box displays the model used for the selected simulation object. Using the lists and check boxes, you can change how the model is displayed and show information about it, as follows:

- Display Models check box. Enables or disables display of the visual model.
- Bounding Volume check box. Displays the bounding volume.
- Daylight check box. Changes the lighting of the model between noon and 9 P.M.
- Marker Labels check box. Displays labels for the markers for sensor and weapon positions.
- 2D check box. Show markers in 2D instead of 3D.
- 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.
- View. Perspective view or view the model from the specified side (left, front, top, and so on.)

64.4. 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 64-15). For example, a lifeform can have an M-16 or an RPG, but not a cruise missile.

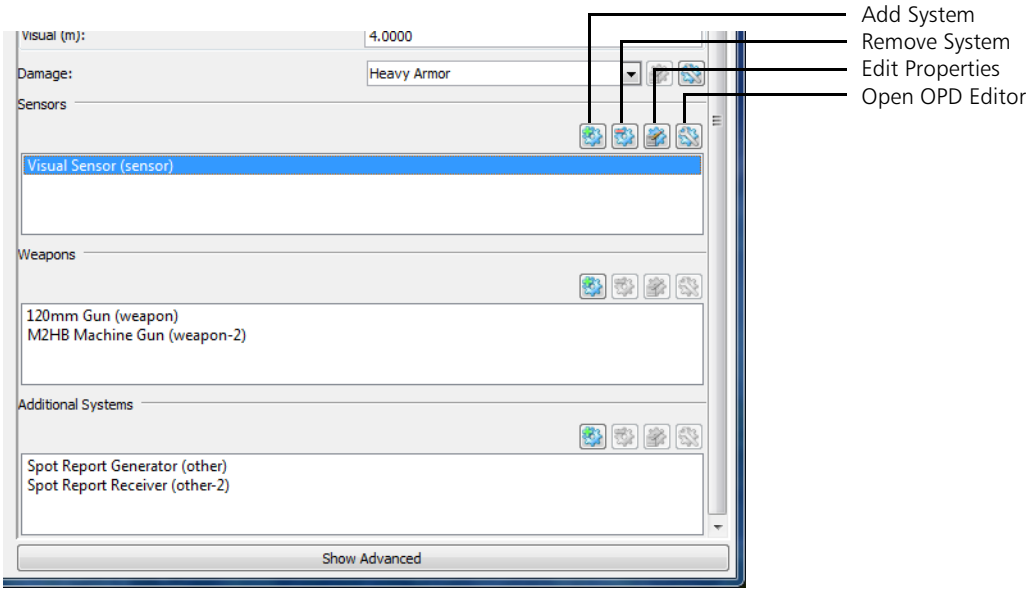


Figure 64-15. Entity systems

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 71-1.) 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.

64.4.1. 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. Click the Add System button () for the system type you want to add. The Add System dialog box opens (Figure 64-16). 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.

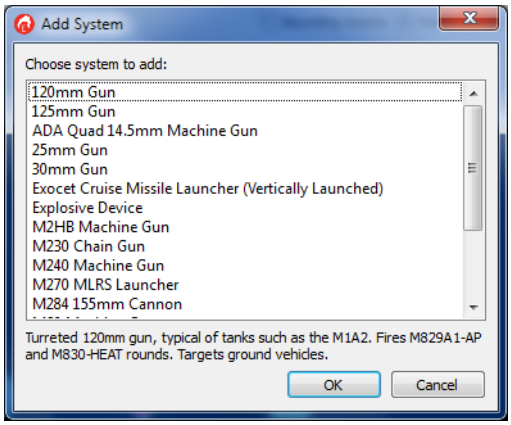


Figure 64-16. Add System dialog box

2. Select the system you want to add.
3. Click OK. The system is added to the list in the Simulation Object Editor.

64.4.2. Selecting a Damage Model

You can only have one damage model specified for a simulation object.

➤ To specify a damage model, select an option from the Damage list.

64.4.3. Selecting a Movement Model

You can only have one movement model specified for a simulation object.

➤ To specify a movement model, in the Movement group box, select an option from the Movement Dynamics list.

64.4.4. 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. 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.
3. Click the Edit Properties button (). A Properties dialog box opens. The 3D model in the Visual Models group box displays a green dot on the simulation object where the system is located. In some cases there are multiple locations.

   Optionally, in the Visual Models group box, select the Marker Labels check box. A label is added to the green dot to identify it.
4. Edit the properties as desired.
5. Click OK.

64.4.5. Editing a System in the OPD Editor

Systems may have attributes that you cannot edit in the Simulation Object Editor. You can jump directly to the OPD Editor from the Simulation Object Editor to edit these attributes. When you open the OPD Editor from the Simulation Object Editor, only the Systems page is displayed. You cannot edit other attributes.

Before you jump to the OPD Editor, save any changes that you have made to the simulation model set. If you make any changes in the OPD Editor, the Simulation Object Editor is automatically updated after you save the changes in the OPD Editor.

To jump to a system in the OPD Editor:

1. Save your changes to the SMS.
2. Select the system that you want to edit.
3. Click the Open OPD Editor button () next to the system.

64.4.6. 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. Select the system you want to remove.
2. Click the Remove System button ().

64.5. 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 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 64-28). 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.

64.5.1. 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 64-17).

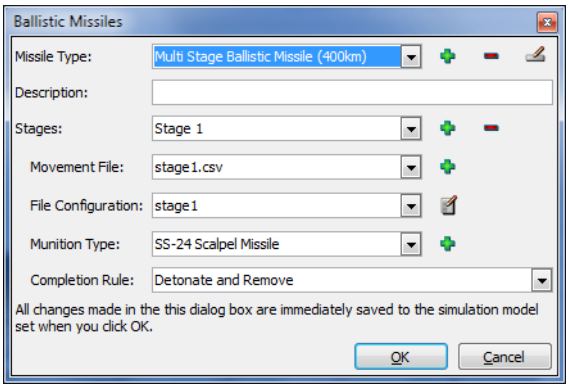


Figure 64-17. Ballistic Missiles dialog box

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 64-30.

- Munition Type. The missile to use.
- Completion Rule. The action to take when the movement in the CSV file is complete.

5. Click OK.

64.5.2. 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 64-18).

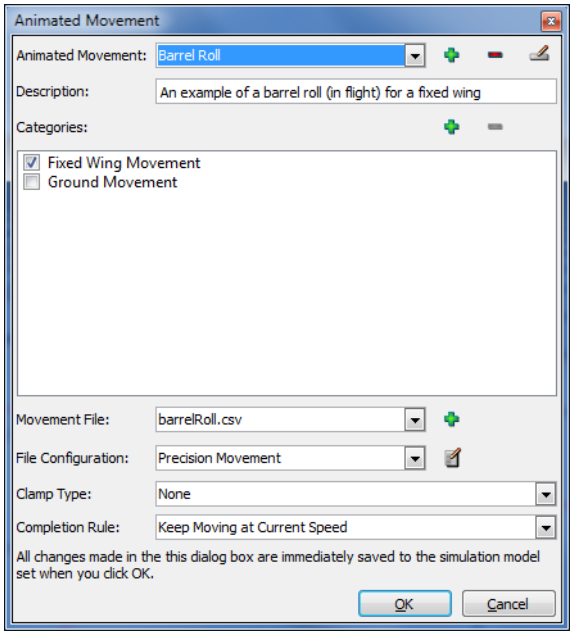


Figure 64-18. Animated Movement dialog box

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 64-30.
7. In the Clamp Type list, select an option for ground clamping.
8. In the Completion Rule list, select the action you want the entity to take after it completes the animated movement task.

64.5.3. 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 64-19).

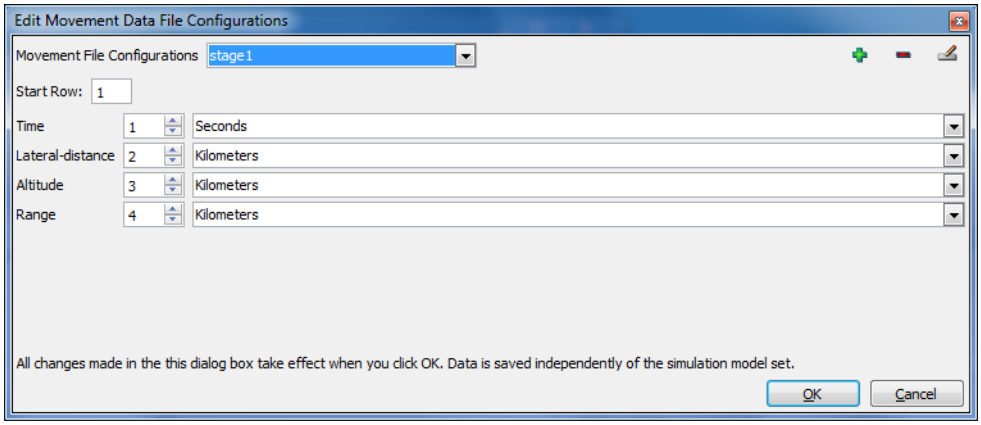


Figure 64-19. Edit Movement Data File Configurations dialog box

- 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.

64.6. Creating a New Entity

You can create a new entity by using an existing entity as a template or by building a completely new entity.



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 entity:

1. Choose **Model** → **New Model**. The Create New Entity dialog box opens (Figure 64-20).

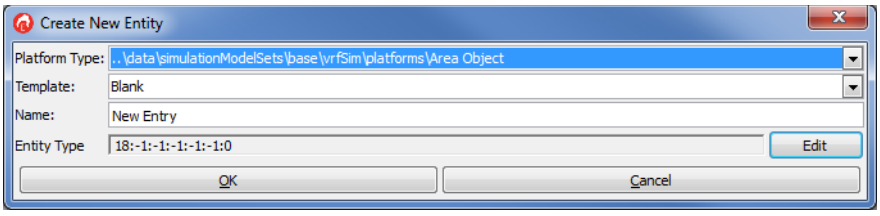


Figure 64-20. Create New Entity dialog box

2. 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. Type a name for the entity.
5. Optionally, 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. Click OK. The Simulation Object Editor adds a new entry to the object list. The editor shows default values for all the fields.
7. Optionally, edit the enumeration.
8. Optionally, specify that this entity is a favorite.
9. Specify 2D and 3D visual models.

10. Specify appropriate values for the entity parameters.
11. Type a name to be used in the entity's label (marking text) in the Short Name text box.
12. Specify a movement system.
13. Specify a damage model.
14. Optionally, add weapon systems.
15. Optionally, add sensors.
16. Save the simulation model set.

64.6.1. Creating a New Entity from an Existing Entity

If you create a new entity from an existing entity, the new entity starts with all of the behavioral characteristics and systems of the original entity. It increments the entity enumeration. You just have to name it and customize it for the new entity.



You can use this procedure to create new units.

To create a new entity from an existing entity:

1. In the object list, select the source entity.
2. Choose **Model** → **New Model from Existing**. The Create New Entity dialog box opens.
3. Type a name for the entity.
4. Optionally, edit the enumeration.
5. Click OK. The new entity 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 entity name.
6. Edit the parameters to provide the new entity the characteristics you want for it.
7. Save the simulation model set.

64.7. Configuring 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.
- 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 configure embarkation for an entity:

1. In the Simulation Object Editor object list, select the entity that you want to configure embarkation for.
2. In the Attributes panel, if Can Be Embarked Upon is not selected, select it.
3. Click Configure. The Configure Embarkation dialog box opens (Figure 64-21).

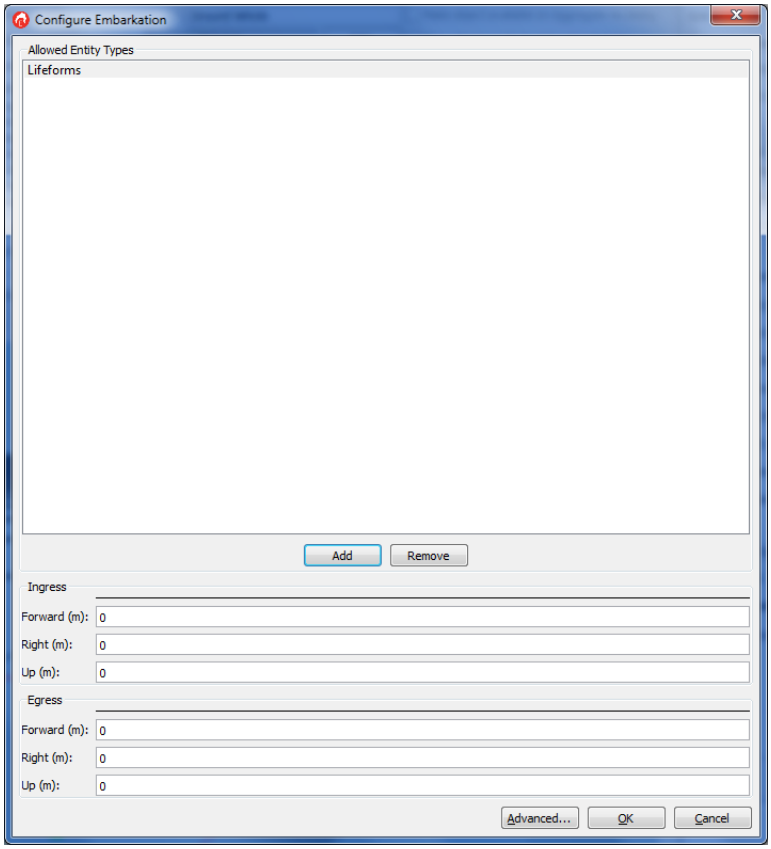


Figure 64-21. Configure Embarkation dialog box

4. Click Add. The Add Type dialog box opens.
5. Select a type of entity that can embark.
6. Optionally, add additional entity types.
7. 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 64-35.
8. Optionally, configure slots. For more information, please see "Configuring Slots," on page 64-38.
9. Click OK.

64.7.1. 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.

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. Click Configure. The Configure Embarkation dialog box opens (Figure 64-21).
3. Click Advanced. The dialog box redisplays.
4. 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 64-22). The coordinates for the ingress point are displayed at the bottom of the dialog box.

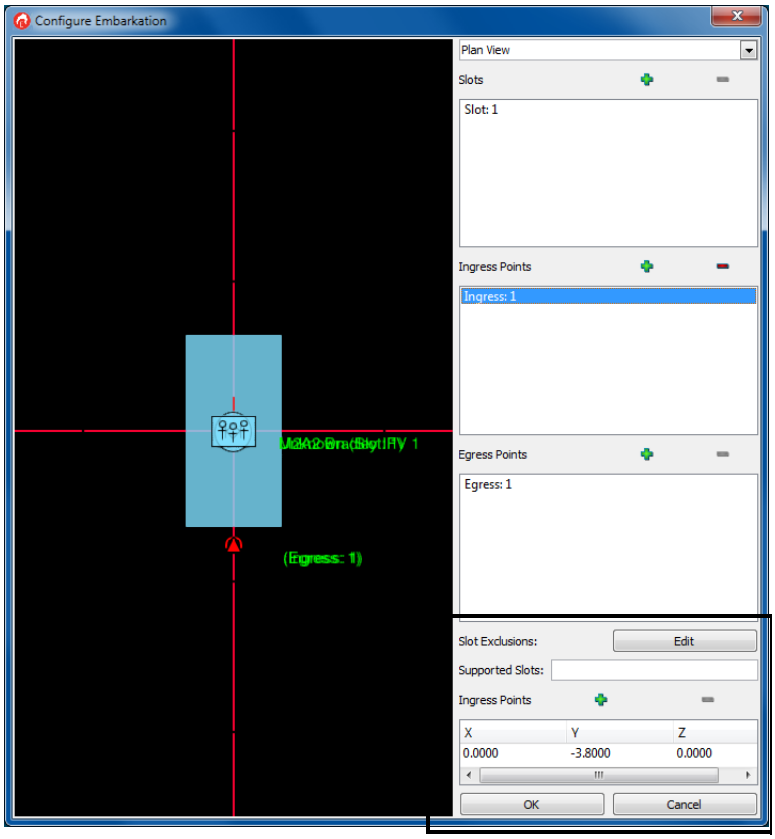


Figure 64-22. Ingress Point

5. Double click the X value for the Ingress point to make it editable.
6. Enter the X value.
7. Repeat for the Y and Z values.
8. 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 64-23).

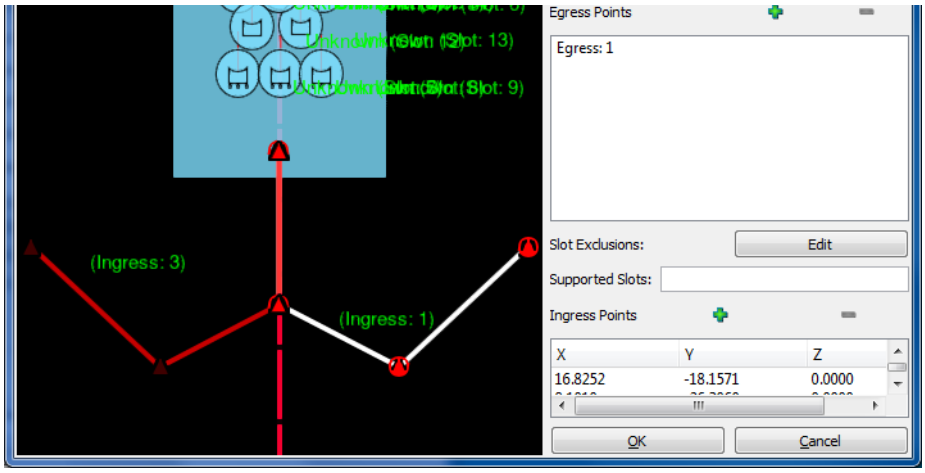


Figure 64-23. Ingress point configuration

9. 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 space.
10. Click OK.

64.7.2. Configuring Slots

The Simulation Object Editor adds a slot to the embarkation configuration for each allowable entity type that you add. The slot is configured with the entity type of the allowable entity type. You can configure several parameters for the slots. You can add and remove slots with the Add and Remove buttons.

To configure slots:

1. In the Simulation Object Editor object list, select the entity that you want to configure embarkation for.
2. Click Configure. The Configure Embarkation dialog box opens (Figure 64-21).
3. Click Advanced. The dialog box redisplays.
4. Select the slot that you want to edit. The dialog box redisplays to show the slot's parameters (Figure 64-24).

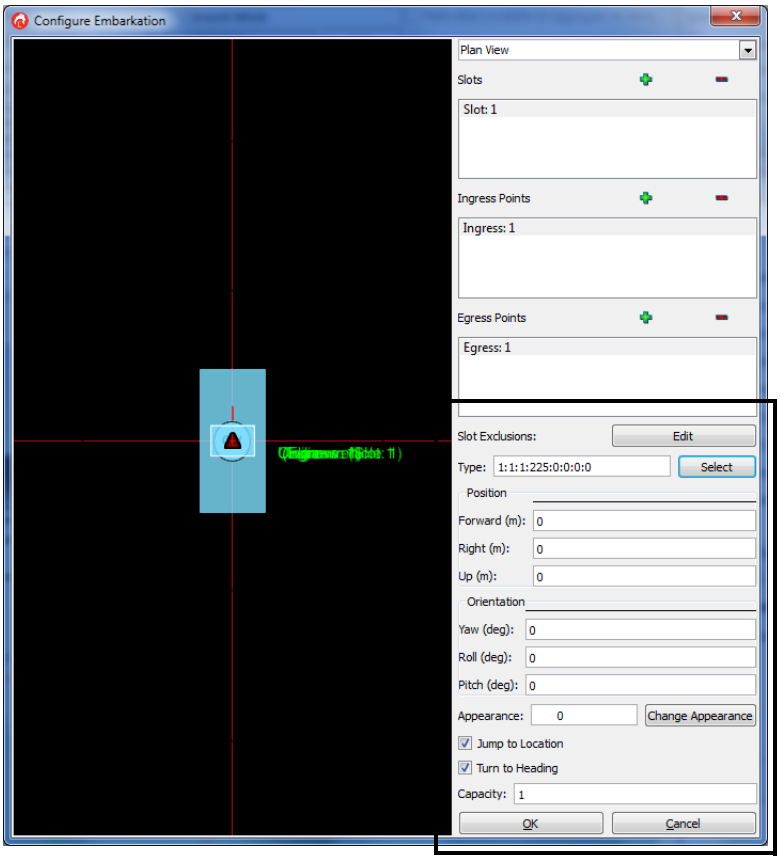


Figure 64-24. Slot parameters

5. Optionally, configure Slot Exclusions. For details, please see "Configuring Slot Exclusions," on page 64-40.

6. If you have added a new slot, in the Type field, specify the entity enumeration for the type of entity that can use this slot. If this is a previously created slot, you can edit the entity type. You can configure different entity types in different slots on an entity.
7. In the Position group box, specify the location of the slot relative to the center of the entity's bounding volume.
8. In the Orientation group box, specify the orientation of the slot relative to the center of the entity.
9. Click Change Appearance and specify the appearance of the entity type when it is in the slot. For example, a human embarked on a truck might have its posture set to Sitting.
10. If you want the embarked entity to immediately jump to the slot, select Jump To Location. If this check box is clear, the entity will move to the slot. For example, an aircraft will taxi to a slot on a carrier deck.
11. If you want the embarked entity to have the same heading as the parent entity, select Turn To Heading.
12. Specify the number of entities that can use this slot.

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 64-25).

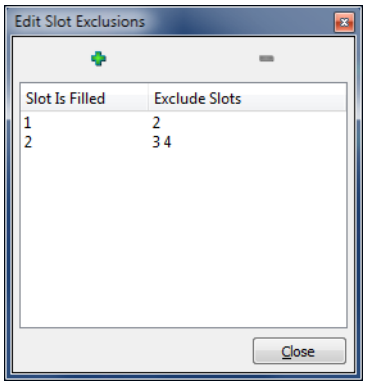


Figure 64-25. Edit Slot Exclusions dialog box

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.

64.8. 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 64-42.

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 category, 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 34-4.

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.

64.8.1. 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 MÄK extensions connection for *AggregateLevel.sms*. You can change the connection if you want to.

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.

64.9. 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`**.

64.10. Navigating Among Edited Simulation Objects

If you have viewed or edited several simulation objects and want to return to a previous object without finding it in the simulation objects list, you can use the Forward and Backward arrows to navigate among previously edited simulation objects. The arrows act like the forward and back buttons in a web browser.

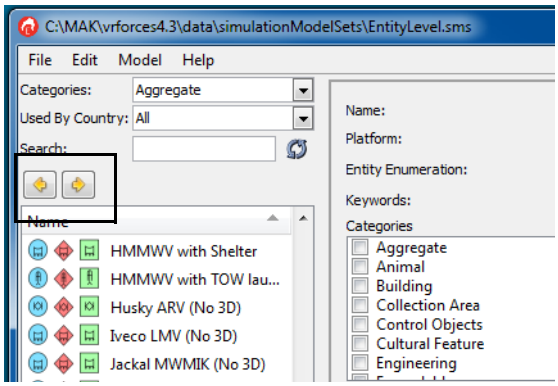


Figure 64-26. Forward and Back arrows

64.11. 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.

64.11.1. 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, click the Browse button (...) for the Menu Icon line (Figure 64-5) 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.

64.11.2. 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 64-45.

To set a default value for a tactical graphic property:

1. In the Categories list, select one of the tactical graphics categories.
2. In the list of tactical graphics, select the one that you want to edit. The applicable properties are enabled in the Visual Model group box (Figure 64-27).

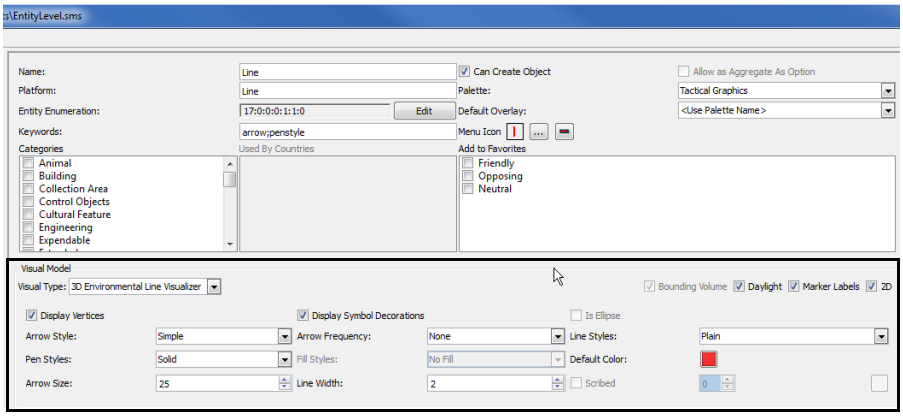


Figure 64-27. Tactical graphics properties

3. Set the default values.

64.11.3. 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, type the appropriate keyword in the Keywords text box. Separate each keyword with a semi-colon (;).

64.11.4. 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 Categories list, select the category for the tactical graphic that you want to edit.
2. In the list of tactical graphics, select the one that you want to edit.
3. In the Visual Model group box, in the Model Sets list, select the model set that you want to edit.
4. Click the Edit button. The Select Schema for Model Set *model_set* dialog box opens (Figure 64-28).

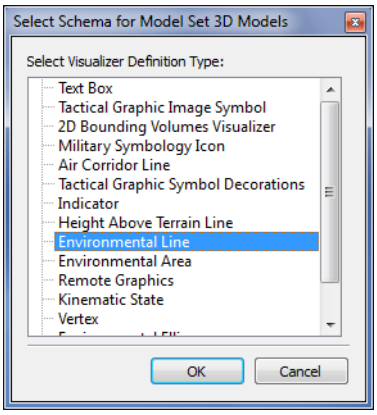


Figure 64-28. Select Schema for Model Set dialog box

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 Categories list, select one of the tactical graphics categories.
2. In the list of tactical graphics, select the one that you want to edit.
3. In the Visual Model group box, in the Model Sets list, select the model set that you want to edit.
4. Click the Texture button. The Select Texture dialog box opens (Figure 64-29).

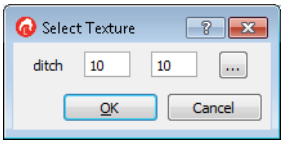


Figure 64-29. Select Texture dialog box

5. Click the Browse button (...). The Select Model Definition dialog box opens (Figure 64-30).

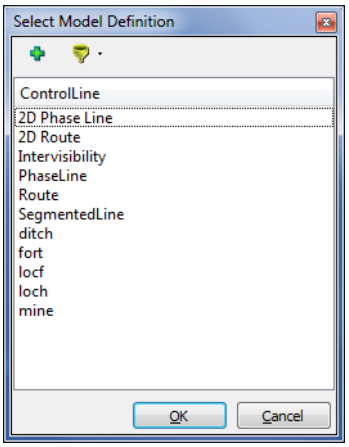


Figure 64-30. Select Model Definition dialog box

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.

64.12. Editing the Category List

If you want to organize simulation objects into categories that are not included in the default category list, you can add new categories. You can also remove and rename categories.



The procedures in this section also apply to editing the Used By Countries list. Select Used by Countries on the Edit menu instead of Categories.

64.12.1. Adding a New Category

To add a new category:

1. Choose **Edit** → **Categories**. The Categories Editor opens (Figure 64-31).

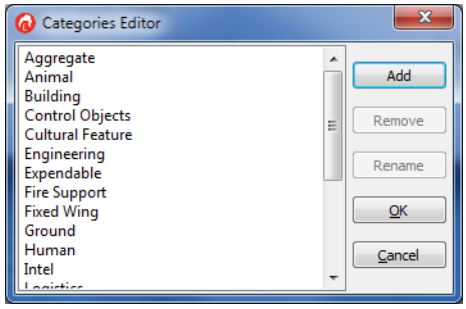


Figure 64-31. Categories Editor

2. Click Add. The New Category dialog box opens.
3. Type the name of the new category.
4. Click OK. The new category is added to the list in the Categories Editor.
5. Click OK.



If you add a category and do not assign a simulation object to it, it does not get saved when you save the simulation model set.

64.12.2. Removing a Category

If you remove a category, it gets removed from any entity that is assigned to it. You must save the SMS to make the change permanent.

To remove a category:

1. Choose **Edit** → **Categories**. The Categories Editor opens.
2. Select the category that you want to remove.
3. Click Remove.
4. Click OK.

64.12.3. Renaming a Category

When you rename a category, all simulation objects that are assigned to the old name are reassigned to the new name. The old name remains in the Categories list until you save the simulation model set, but it is not listed in the Categories Editor. If you assign a simulation object to the old category name before saving the SMS, the old name is retained and added to the list in the Categories Editor.

To rename a category:

1. Choose **Edit** → **Categories**. The Categories Editor opens.
2. Select the category that you want to rename.
3. Click Rename. The Rename dialog box opens.
4. Type the new name.
5. Click OK. The categories list is updated.
6. Click OK.

64.12.4. Undoing Category Changes

You can undo changes made to the category list and entity category assignments.

When you undo category changes, all changes to the categories list and all changes to entity category 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 category changes:

1. Choose **Edit** → **Revert Categories**. A confirmation prompt is displayed.
2. Click Yes.

64.13. 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.

64.13.1. 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 64-32).

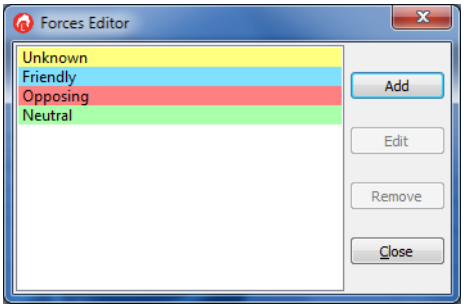


Figure 64-32. Forces Editor

2. Click Add. The Force Hostility Editor opens (Figure 64-33). The new force is assigned a force ID. You cannot edit the force ID.

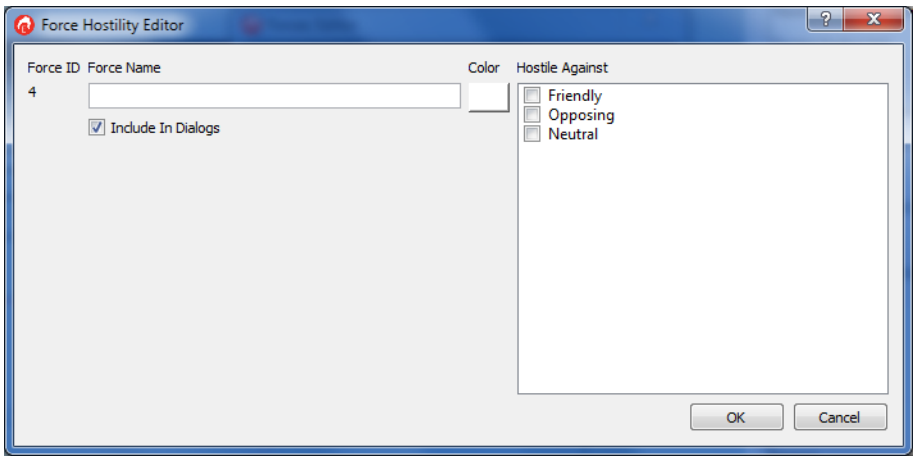


Figure 64-33. Force Hostility Editor

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 9-10.
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.

64.13.2. 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 64-32).
2. Select the force that you want to edit.
3. Click Edit. The Force Hostility Editor opens (Figure 64-33).
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.

64.13.3. 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 64-32).
2. Select the force that you want to remove.
3. Click Remove. The force is removed from the list.
4. Click Close.

64.14. 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.



65. 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 65-2
- Creating a Unit for Entity-Level Scenarios 65-2
- Specifying an Aggregate As Option 65-3
- Editing Unit Composition 65-4
  - Adding Subordinates 65-5
  - Removing Subordinates 65-5
  - Replacing Subordinates 65-6
  - Changing the Order of Subordinates 65-6
  - Editing a Subordinate Unit 65-7
- Editing Formations 65-8
  - Adding Formations 65-9
  - Removing Formations 65-10
  - Renaming Formations 65-10
  - Specifying the Default Formation 65-10
  - Expanding and Collapsing the Display of Formations 65-10
  - Displaying Bounding Boxes 65-11
  - Copying Formations 65-12
  - Laying Out a Formation Automatically 65-13
  - Changing a Formation's Layout by Hand 65-13

65.1. 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 63, *The Simulation Object Editor and SMSs* and Chapter 64, *Editing Simulation Object Models*.

The two major differences in configuring the different types of units is that units in entity-level scenarios have 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.

65.2. 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 Entity," on page 64-31.
2. Add subordinates.
3. Optionally, configure formations.

65.3. 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 Allow as Aggregate As Option check box.

65.4. 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 65-1 illustrates the Simulation Object Editor with a unit selected. The Unit Composition group box is highlighted.

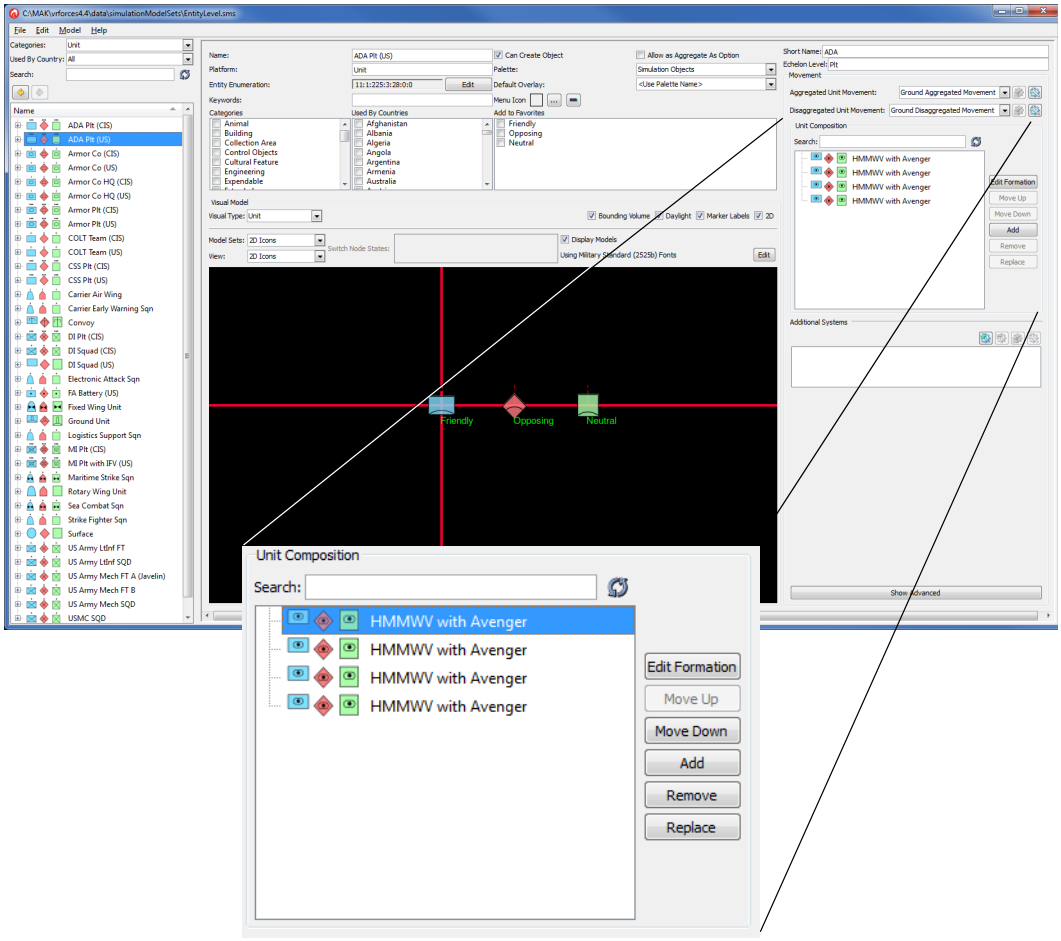


Figure 65-1. Unit Composition

65.4.1. Adding Subordinates

When you create a new unit without copying an existing unit, you must specify its subordinates. You can also add subordinates to existing units.

To add a subordinate to a unit:

1. In the Unit Composition group box (Figure 65-1), click Add. The Add to Hierarchy dialog box opens (Figure 65-2).

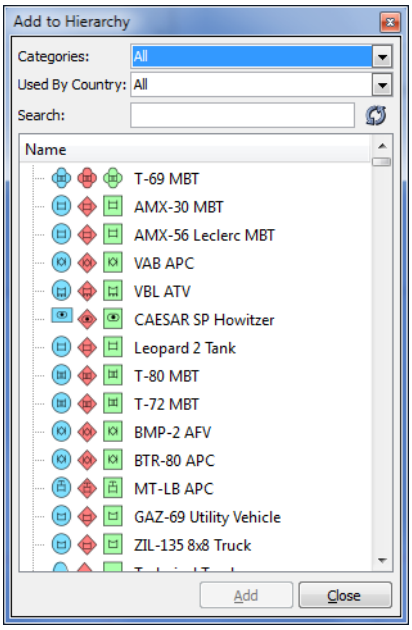


Figure 65-2. Add to Hierarchy dialog box

2. In the list of simulation objects, select the simulation object that you want to add.
3. 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.
4. Continue to add as many subordinates as desired.
5. Click Close.

65.4.2. Removing Subordinates

To remove a subordinate from a unit:

1. In the Unit Composition window (Figure 65-1), select the subordinate that you want to remove.
2. In the Unit Composition group box, click Remove. The subordinate is removed from the unit.

65.4.3. 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 65-1), 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 65-2).
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.

65.4.4. 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 65-1), select the subordinate that you want to move.
2. In the Unit Composition group box, click Move Up or Move Down.

65.4.5. 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.)

Navigating the Subordinate Hierarchy

If you jump to a subordinate in the Simulation Object Editor, you can quickly back up to its superior unit by clicking the back arrow above the object list (Figure 65-3).

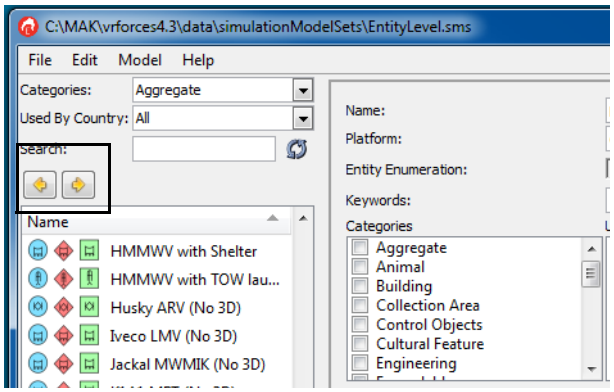


Figure 65-3. Forward and Back arrows

65.5. 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 70-8.

65.5.1. Adding Formations

To add a formation:

1. In the Unit Composition group box (Figure 65-1), click Edit Formation. The Formation Editor opens (Figure 65-4).

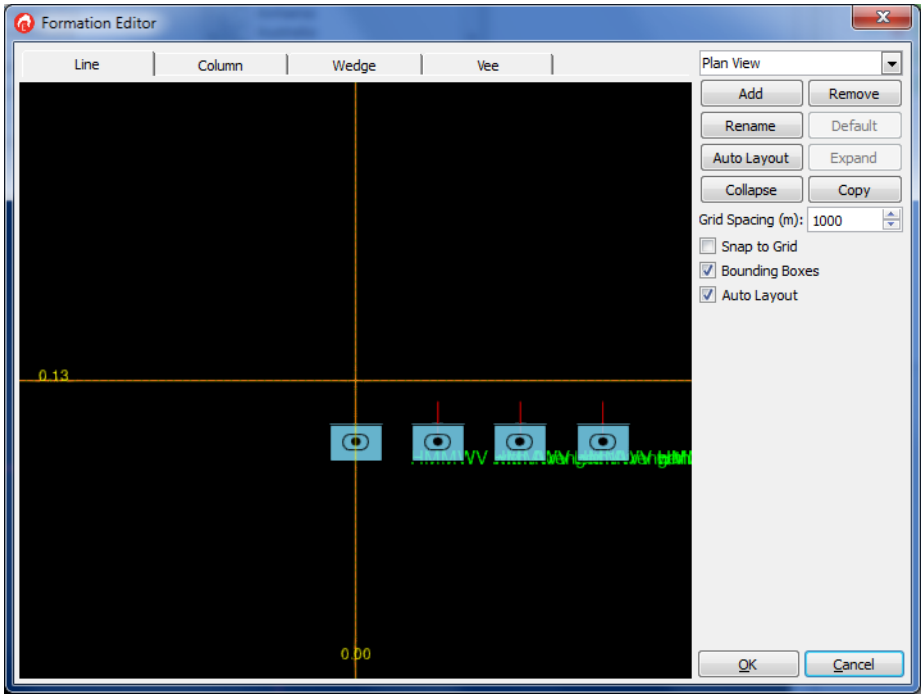


Figure 65-4. Formation Editor

2. In the Formation Editor, click Add. The Add Formation dialog box opens (Figure 65-5).

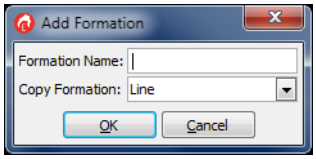


Figure 65-5. Add Formation dialog box

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 65-13.

65.5.2. Removing Formations

To remove a formation:

1. In the Unit Composition group box (Figure 65-1), click Edit Formation. The Formation Editor opens (Figure 65-4).
2. In the Formation Editor, select the tab for the formation that you want to remove.
3. Click Remove. The tab is removed.

65.5.3. Renaming Formations

To rename a formation:

1. In the Unit Composition group box (Figure 65-1), click Edit Formation. The Formation Editor opens (Figure 65-4).
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.

65.5.4. 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 65-1), click Edit Formation. The Formation Editor opens (Figure 65-4).
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.

65.5.5. 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.

65.5.6. 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 65-6 illustrates a unit with the display of bounding boxes off and on.

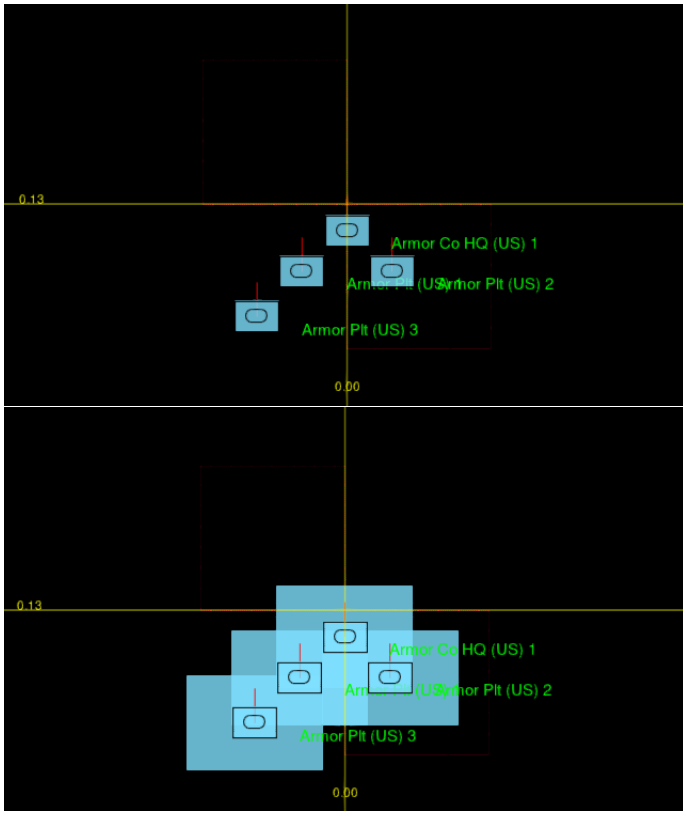


Figure 65-6. Unit bounding boxes in Formation Editor

➤ To display or hide bounding boxes, in the Formation Editor (Figure 65-4), select or clear the Bounding Boxes check box.

65.5.7. 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 65-4).
3. In the Formation Editor, click Copy. The Select Formations dialog box opens (Figure 65-7).

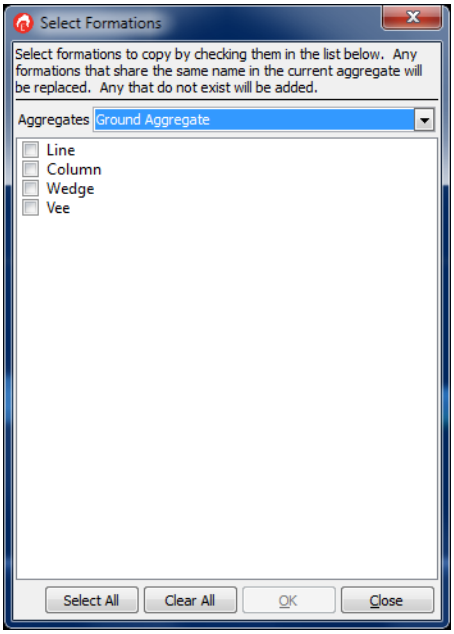


Figure 65-7. Select Formations dialog box

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.

65.5.8. 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 65-4).
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.

65.5.9. Changing a Formation's Layout by Hand

You can lay out a formation by hand. The grid line in the Formation Editor indicates the center of the formation (Figure 65-8). This is where the unit icon is placed when the unit is collapsed or aggregated.

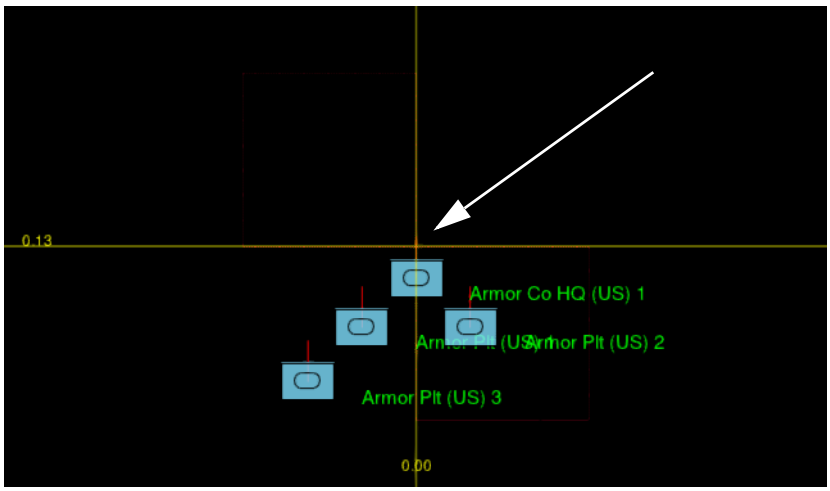


Figure 65-8. Formation Editor crosshair

To change a formation's layout:

1. In the Unit Composition group box, click Edit Formation. The Formation Editor opens (Figure 65-4).
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.



66. 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 66-3
Configuring Simulation Objects for Aggregate-Level Modeling 66-3
Aggregating Units into Higher Echelon Units 66-4
Configuring Ammunition, Weapons, and Equipment 66-5
Defining Ammunition, Weapons, and Equipment Items 66-8
Assemblies 66-10
Creating Assemblies 66-11
Editing Assemblies 66-12
Adding Assemblies to a Unit 66-13
Removing Assemblies 66-13
Creating Roll Up Rules 66-14
Rolling Up Assemblies 66-15
Adding Activities 66-16
Setting Object Parameters for Aggregate-Level Modeling 66-17
Physical Parameters 66-18
Movement Parameters 66-19
Sensor Parameters 66-21
Attack Parameters 66-22
Defense Parameters 66-23
Electronic Warfare (EW) Parameters 66-23
Personnel and Equipment Parameters 66-24
Supplies Parameters 66-24
Configuring Combat Engineering Objects 66-25

Editing Reader/Writer Key/Value Pairs 66-26

66.1. 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.

66.2. 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 categories 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 66-1).

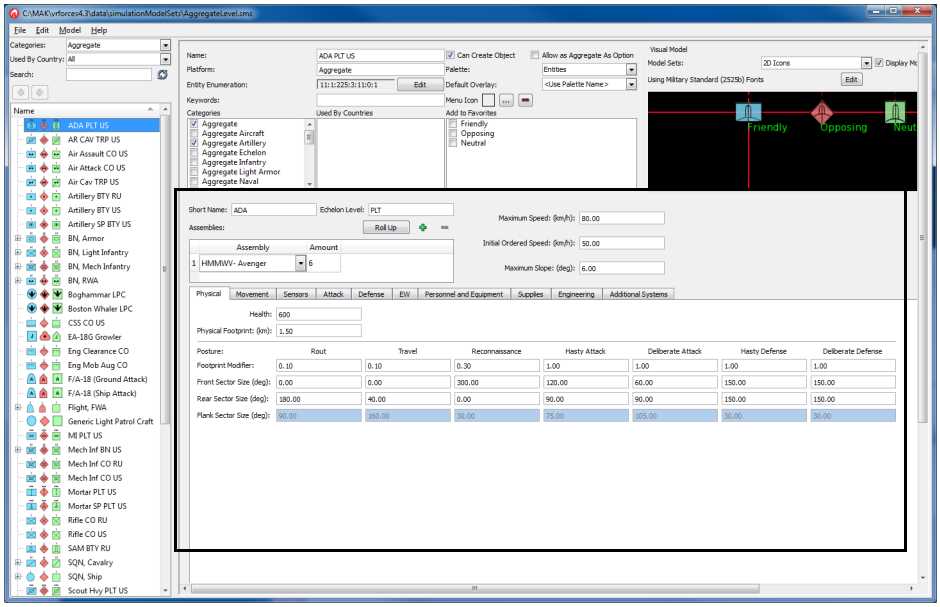


Figure 66-1. Unit in *AggregateLevel.sms*

The parameters for simulation objects in *AggregateLevel.sms* are organized into the following groups:

- 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.
- 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.
- Defense. Specifies vulnerability to different types of attack and modifiers to vulnerability.
- 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.
- 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 "Setting Object Parameters for Aggregate-Level Modeling," on page 66-17.

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.

66.2.1. 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.

66.3. 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 66-2).

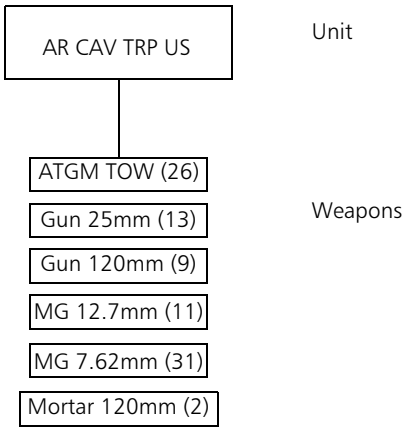


Figure 66-2. Anatomy of a unit without using assemblies

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 66-3 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 66-4).

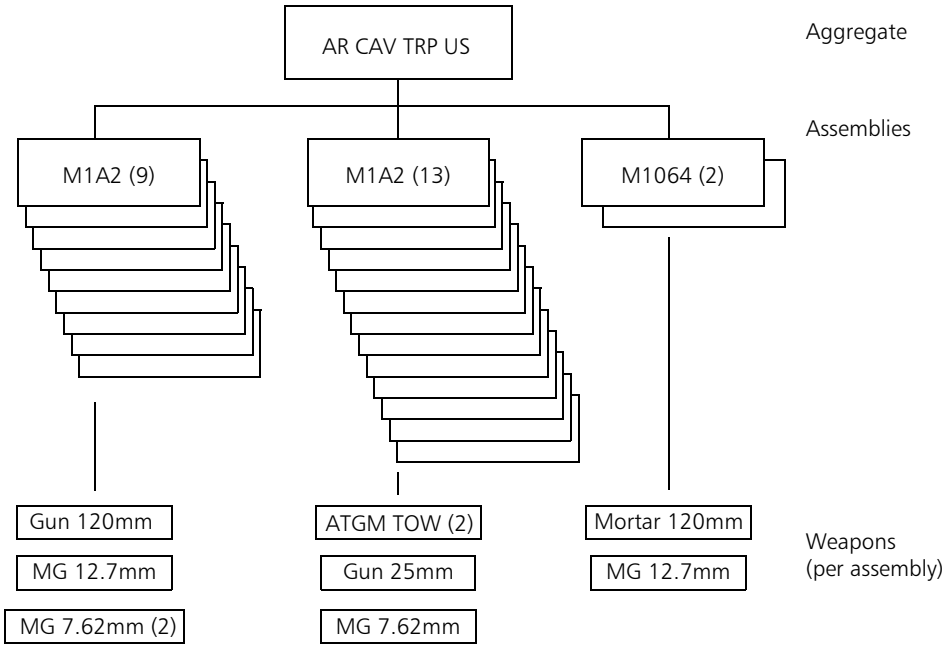


Figure 66-3. Anatomy of a unit using assemblies

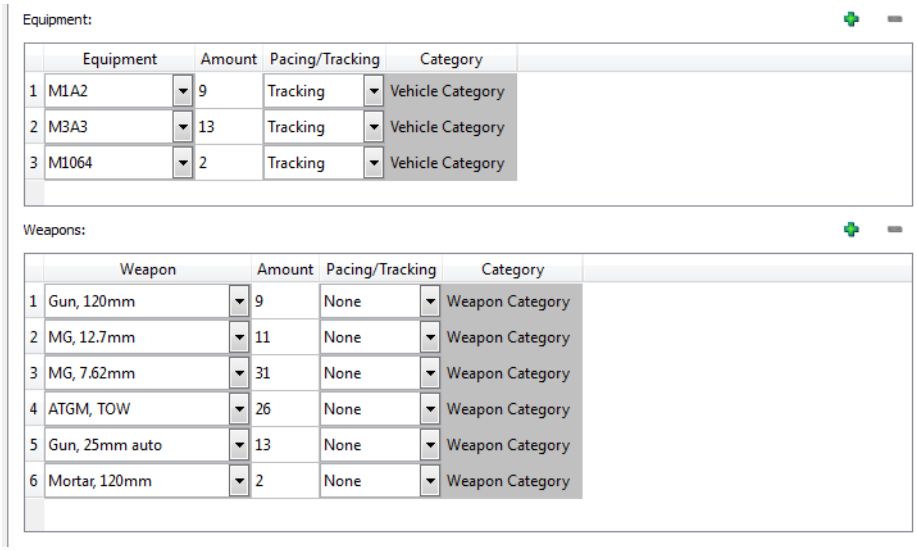


Figure 66-4. AR CAV TRP US weapons rolled up on Personnel and Equipment tab

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 66-10.

66.3.1. 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 66-5).

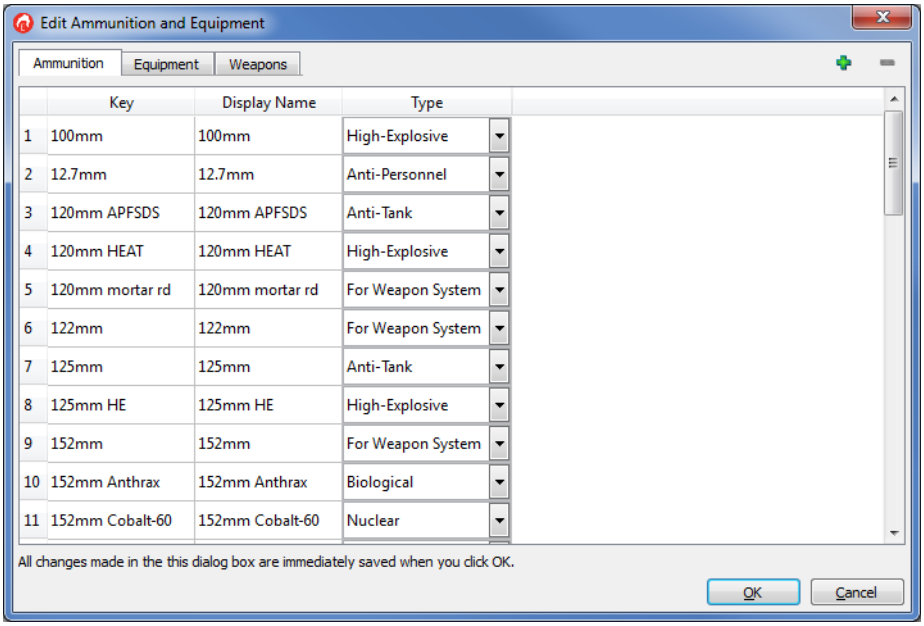


Figure 66-5. Edit Ammunition and Equipment dialog box

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.

66.4. 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.

66.4.1. 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 66-6).

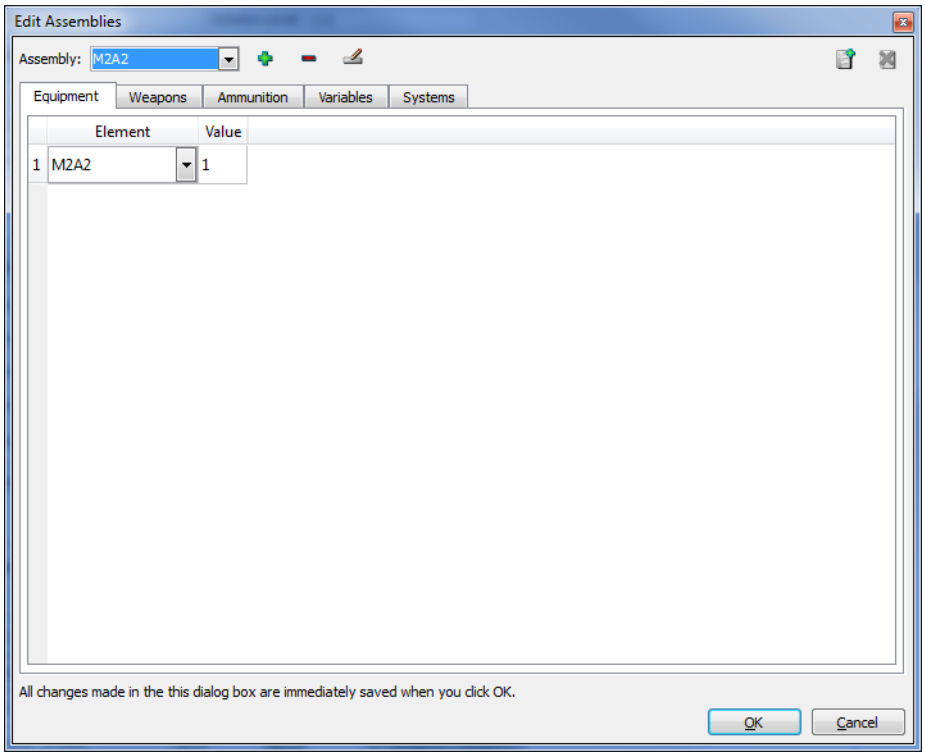


Figure 66-6. Edit Assemblies dialog box

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.

66.4.2. 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 66-6).
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.

66.4.3. 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 66-7). A new row is added to the list.

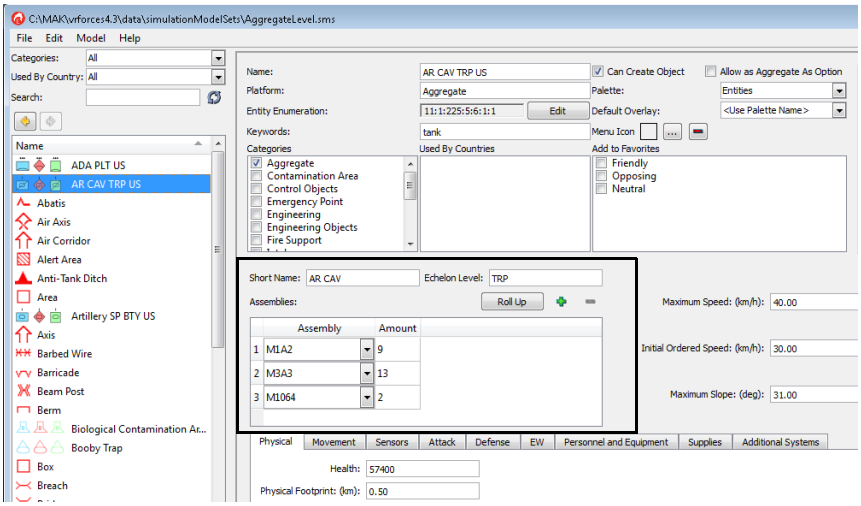


Figure 66-7. Assemblies table

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.

66.4.4. 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 (−).

66.5. 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 66-8).

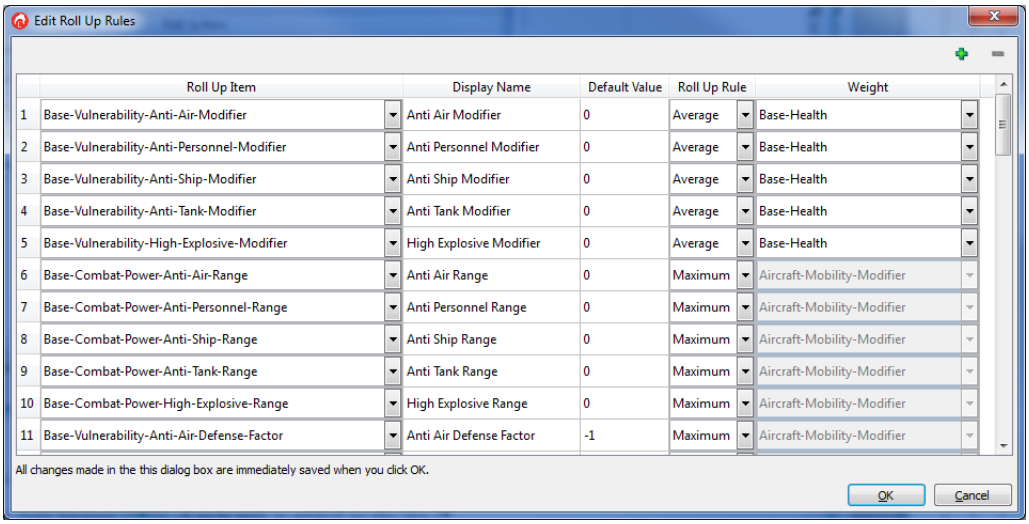


Figure 66-8. Edit Roll Up Rules dialog box

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.

66.6. 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.

66.7. 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 66-9).

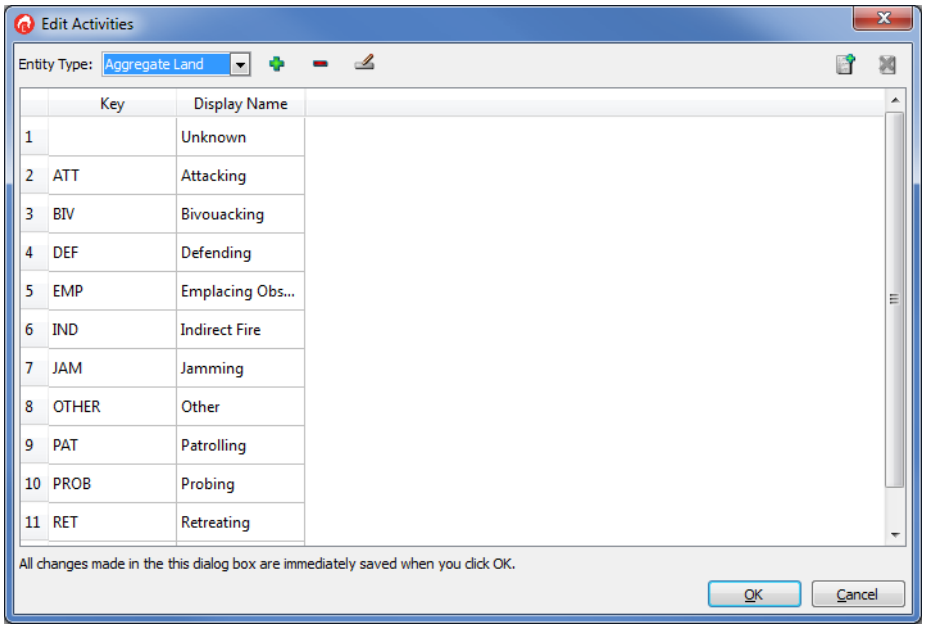


Figure 66-9. Edit Activities dialog box

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.

66.8. Setting 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.

66.8.1. 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.

Table 66-1: Physical parameters

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.
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°, within which rear-facing sensors and interactions take place.
Flank Sector Size	The number of degrees, centered on the heading +/- 90°, within which side-facing sensors and interactions take place.

66.8.2. 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.

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 "Configuring Embarkation," on page 64-33.

Table 66-2 describes the movement parameters.

Table 66-2: 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 66-3 describes movement system properties. Systems do not all have the same properties.

Table 66-3: 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 this 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 66-10).

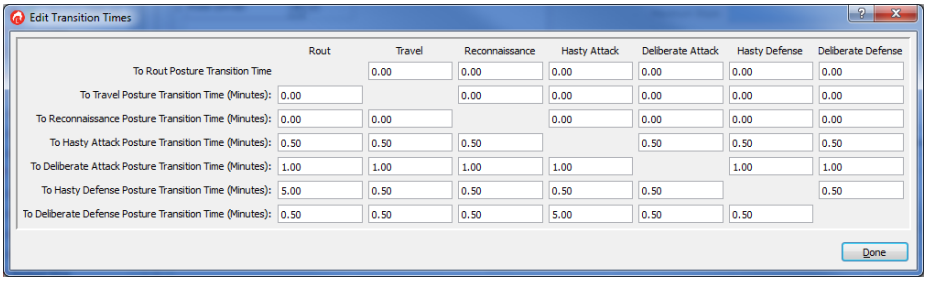


Figure 66-10. Edit Posture Transition Times dialog box

3. Change the times, in minutes, as desired. Changes take effect immediately.
4. Click Done.

66.8.3. 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 64-22.)

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.

66.8.4. 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.

Table 66-4: 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.
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.

Table 66-4: Attack parameters

Parameter	Description
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.

66.8.5. Defense Parameters

The Defense tab lets you specify modifiers to a simulation object's vulnerability. Vulnerability is specified within each weapon domain.

Table 66-5: 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.

66.8.6. 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.

66.8.7. 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 66-6: 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.

66.8.8. Supplies Parameters

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 66-26.) 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.

66.9. 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 66-7 describes the parameters specific to CEOs. Some parameters do not apply to all CEOs.

Table 66-7: 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 66-7: 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.

66.10. 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 66-8 describes the effect of adding new keyword:value pairs.

Table 66-8: 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.



67. 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 67-2
Copy DI-Guy Files to the Data Directory 67-2
Add Model Definitions and Element Definitions 67-3
DI-Guy Configuration Files 67-10
Adding New Characters to the Character Digest 67-11
Adding a New DI-Guy Animation 67-11
How VR-Forces Maps DI-Guy Models 67-13
Add a New Entity in the Simulation Object Editor 67-13

67.1. Adding New DI-Guy Models to VR-Forces

VR-Forces supports DI-Guy characters and includes many lifeform characters. However, MÄK 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. Add the new character to *character_digest.xml*. If the character has animations, add them to *character_animations_table.mtl*.
4. Create a new VR-Forces character in the Simulation Object Editor.

67.1.1. 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 SISO Enumerations mapped to element definitions or by recognizing custom DI-Guy PDUs.

- If you want to map your custom characters to element definitions, follow the procedures in this chapter.
- 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.



If you want to be able to create custom DI-Guy characters in a scenario, then you must add new simulation models in the Simulation Model Editor. This will also automatically create the complementary model and element definitions and object type mappings.

67.1.2. Add Model Definitions and Element Definitions

To visualize a DI-Guy character in VR-Forces, you must add a model definition and element definition for each character.

Create a DI-Guy Model Definition

To create a model definition for a DI-Guy:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page.
3. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens.
4. Type a name for the new model definition. It must end with the letters DIG.
5. In the Schema list, select DiGuyCharacter (Figure 67-1).

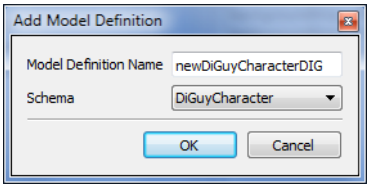


Figure 67-1. Add Model Definition dialog box

6. Click OK. The new model definition is added. It is highlighted in red because it is missing parameters (Figure 67-2).

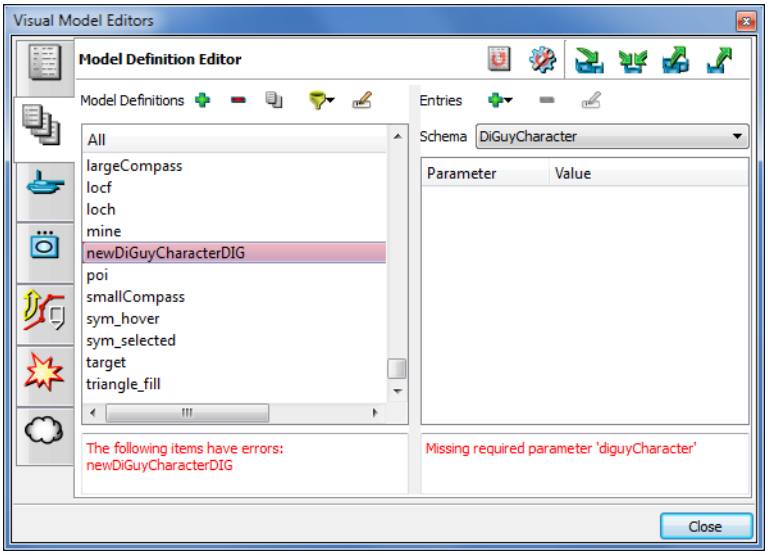


Figure 67-2. New model definition

7. Click the Add Parameter button (). A list of parameters is displayed (Figure 67-3).

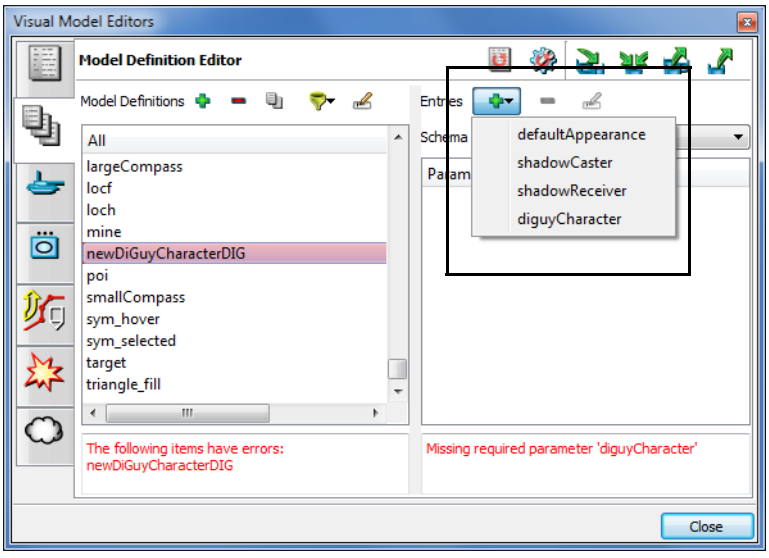


Figure 67-3. Add parameter list

8. Select defaultAppearance. It is added to the parameter list.
9. Click the value for the parameter. The field becomes editable.
10. Type the default appearance name for this character (Figure 67-4).
11. Add a diguyCharacter parameter.

12. Click the value for the parameter. The field becomes editable.
13. Type the character name for this character.
14. Add the shadowCaster and shadowReceiver parameters (Figure 67-4).

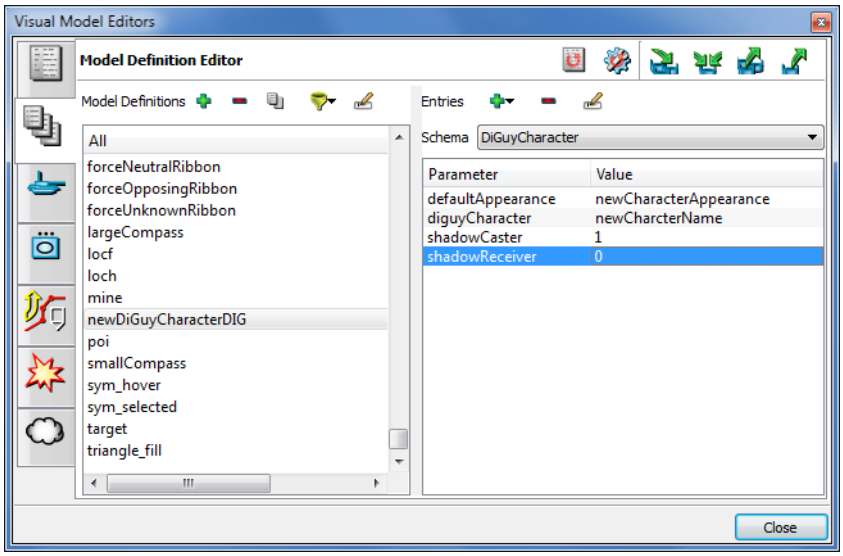


Figure 67-4. DI-Guy character parameters

Add an Element Definition

To create an element definition:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page.
3. Expand the element definition hierarchy to the Life Form level and select the type of life form you are adding (Figure 67-5).

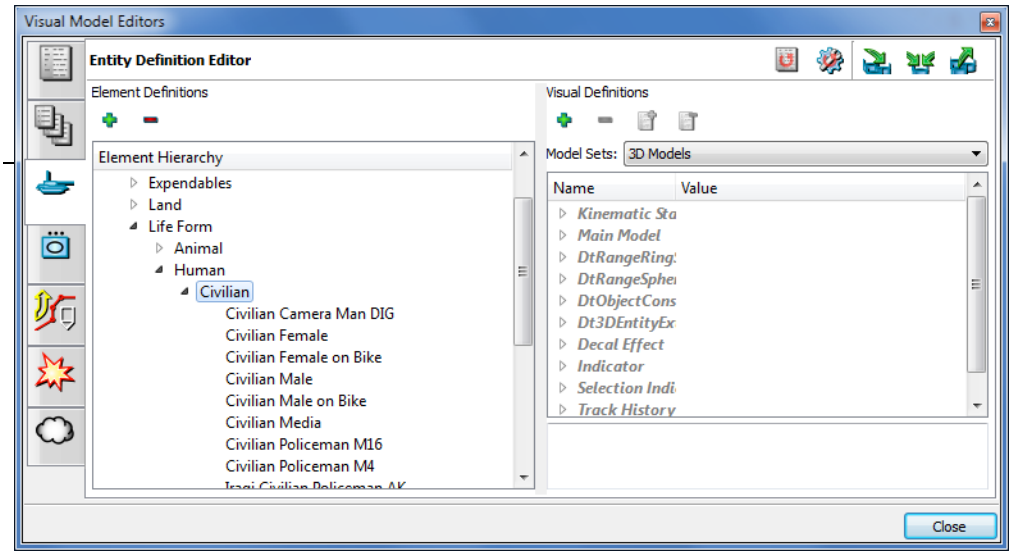


Figure 67-5. Entity Definition Editor page

4. Click the Add button (✚). The Create Element Definition dialog box opens.
5. Type a name for the element definition. (You cannot edit the name after you create it, so try to avoid typing errors.)
6. Click OK. The new element is added to the list (Figure 67-6).

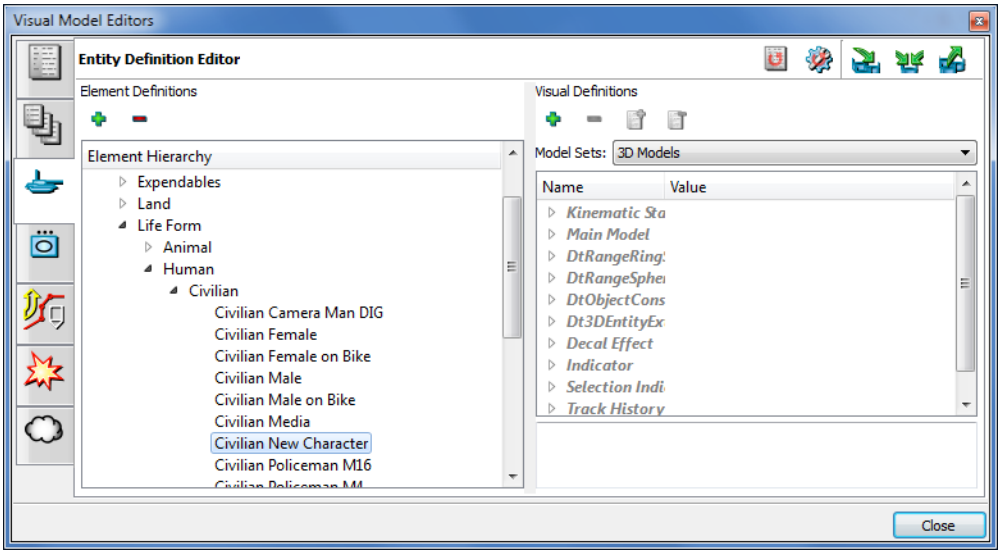


Figure 67-6. New entity definition added

7. In the Visual Definitions list, select Main Model (Figure 67-7).

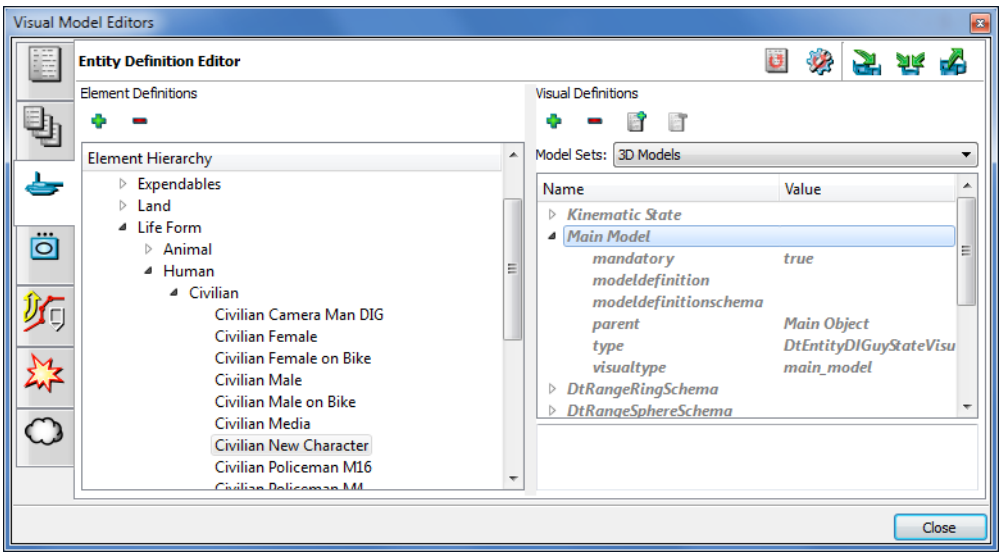


Figure 67-7. Main Model visual definition

8. Click the Value for the **modeldefinition** attribute. The Choose Model Definition dialog box opens.
9. Click the Add button (+). A list is added to the window (Figure 67-8).

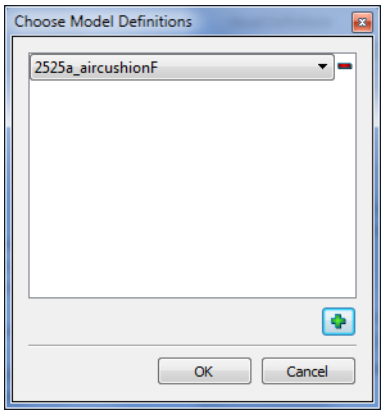


Figure 67-8. Choose Model Definition dialog box

10. Select the model definition that you created for this character (Figure 67-9).

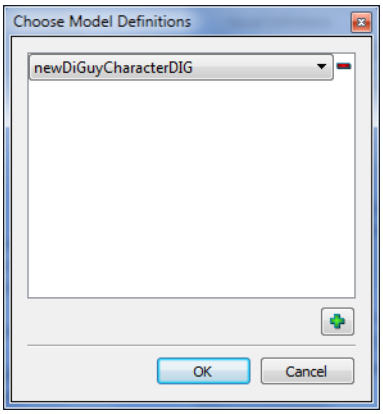


Figure 67-9. New model definition selected

11. Click OK. The model definition is updated (Figure 67-10).

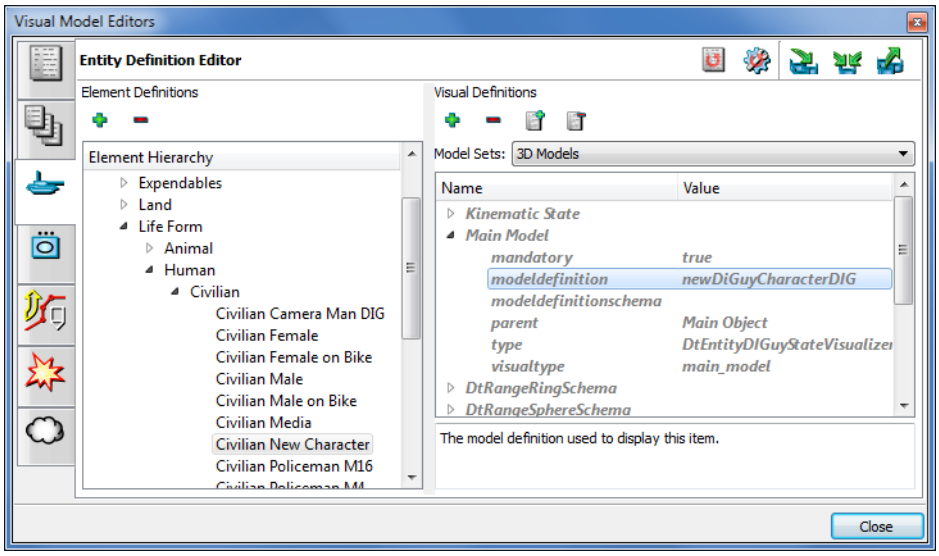


Figure 67-10. Completed entity definition

12. Select the value column for the **modeldefinitionschema** attribute. The Choose Supported Model definition Schema dialog box opens.
13. Click the Add button (). A list is added to the window.
14. Select DiGuyCharacter.
15. Click OK.
16. Close the dialog box.

67.2. 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 26-14, "DI-Guy Appearance," on page 30-15, and "DI-Guy Hand Item," on page 30-15.

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 26-14"Editing a Lifeform's DI-Guy Character, Appearance, and Animation," on page 64-14.) The DI-Guy mappings are stored in the entity's *.entity* file.

The DI-Guy characters, appearances, and animations available to lifeforms are controlled by two configuration files:

- *./appData/settings/vrfSim/character_animations_table.mtl* lists the DI-Guy characters available for VR-Forces entities and the appearances available for each character. 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 that are already in *character_digest.xml*. 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.

- *./appData/settings/vrfSim/character_digest.xml* contains information that VR-Forces needs to synchronize a lifeform's simulated location with the movements generated by DI-Guy animations. For example, if a model's animation causes the model to step to the left, VR-Forces needs to move the simulated location of the entity the same distance to the left.

67.2.1. Adding New Characters to the Character Digest

If you create a new character, you have two options for adding it to the character digest. You can use the DI-Guy SDK to generate a new version of *character_digest.xml*, or you can edit the file by hand. For details about using the SDK to update this file, please see *$(DIGUY)\programming_examples\DIGuyDigest* in your DI-Guy installation.

To add a new character manually:

1. Open *character_digest.xml* in a text editor.
2. Copy the entry for a character that is similar to your new character.
3. Change the attribute values to valid values for your new character.
4. Save the file.

67.2.2. 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 the list of animations for this character in *./appData/settings/vrfSim/character_digest.xml*. For example, if the new animation is *char_chicken_action_table.cfg*, add an entry to the chicken character. The following example assumes a new animation called lay_egg. The chicken will sit, lay the egg, then stand up.

```
<character_type name="chicken">
  <class>animal</class>
  <quality_bias>5</quality_bias>
  <character_type_map>
    <type_map_field name="character_class" value="animal" />
    <type_map_field name="category" value="domestic" />
    <type_map_field name="subcategory" value="chicken" />
  </character_type_map>
  <default_appearance name="chicken_1" />
  <default_skinned_appearance name="" />
  <appearances>
    <appearance name="chicken_1" />
    <appearance name="chicken_2" />
    <appearance name="chicken_3" />
  </appearances>
  <actions>
    <action name="lay_egg" > <---Needs to match name in action table
 and motion file
      <duration>1.500000</duration>
      <speed>0.000000</speed>
      <direction>none</direction>
      <posture>upright</posture>
      <primary_variant>normal</primary_variant>
    </action>
    <action name="stand" >
      <duration>0.500000</duration>
      <speed>0.000000</speed>
      <direction>none</direction>
      <posture>upright</posture>
      <primary_variant>normal</primary_variant>
    </action>
...
```

5. 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")
    )
...
```

67.3. 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.

67.4. Add a New Entity in the Simulation Object Editor

Add a new entity in the Simulation Object Editor, as described in "Creating a New Entity," on page 64-31. Give the new entity the same name as the element definition you created in the Visual Model Editor dialog box. This will ensure that the entity enumeration is mapped to the correct element definition.

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*. The list of appearances is drawn from a character's entry in *./appData/settings/vrfSim/character_digest.xml*. Therefore, these files must have been updated before you can specify the 3D model in the Simulation Object Editor.



68. 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 68-2
VR-Forces Classes and Object Parameters 68-3
Object Types 68-4
Parameter Types 68-8
Local and Remote Subcomponents 68-9
Component Descriptors and Parameters 68-13
Common Elements of Component Descriptors 68-13
Configuring the Priority of Components 68-16
System Definition Files 68-16
Creating a System Definition File 68-17
Configuring System Connections 68-18
Adding Variable Bindings to Platform and System Files 68-21
Adding a Variable Binding 68-22
Making Variables Editable in the Simulation Object Editor 68-23
Adding Platform Variable Bindings to the Simulation Object Editor . 68-23
Adding System Definition File Variables to the Simulation Object Editor ... 68-25
Configuring Units 68-25
Configuring Tactical Graphics 68-26
Pathname Interpretation in Simulation Model Set Files 68-26
Updating Object Parameter Database Files for New Releases 68-27

68.1. 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 68-1 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.

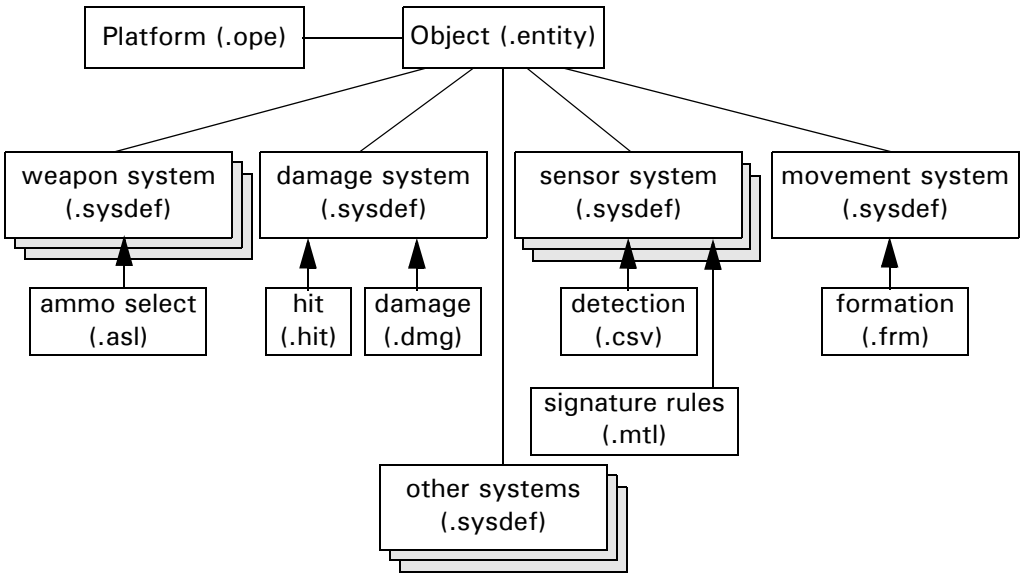


Figure 68-1. Object definition

68.1.1. 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.

68.1.2. 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 68-2 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 (*Reference for Enumeration for Simulation Interoperability*).

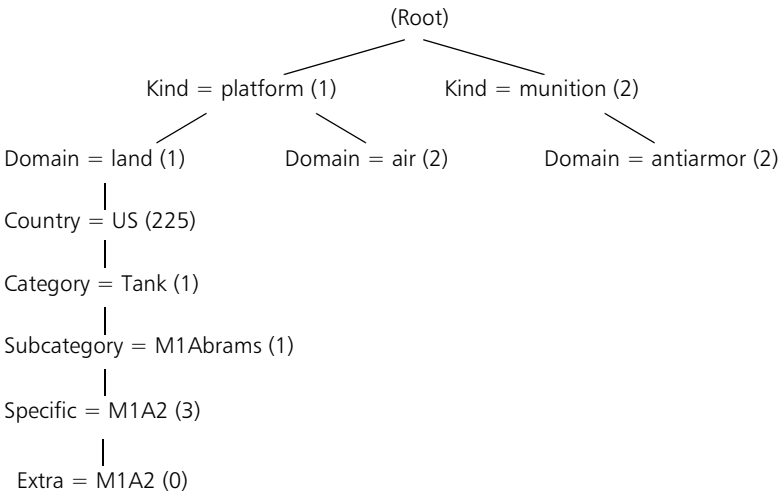


Figure 68-2. Object type enumeration tree

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 68-3). 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).

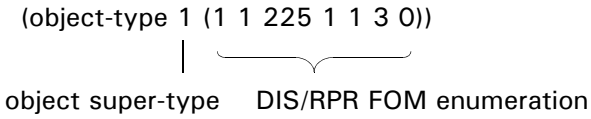


Figure 68-3. VR-Forces object-type parameter

Table 68-1 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 B-9.

Table 68-1: 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 68-7.)
- 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 *DtVrfObjects*.

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 68-7.) 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 68-4, 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.

68.1.3. 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 68-4 illustrates the object parameter type hierarchy.

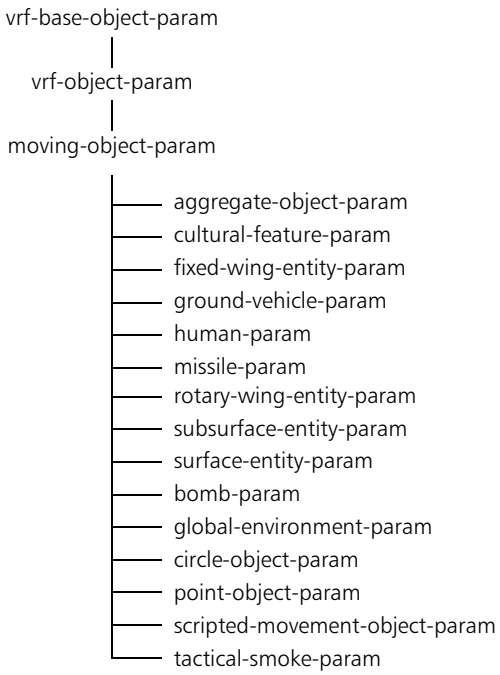


Figure 68-4. Object parameter hierarchy

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 68-12.



The **object-type** is specified in an object's *.entity* file. The parameter-type is specified in the platform (*.ope*) file.

68.1.4. 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 68-2 maps the subcomponent classes to the parameters that specify them.

Table 68-2: 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 68-3 lists the permissible entries for each parameter listed in Table 68-2.

Table 68-3: Permissible subcomponent parameters

Parameter	Permissible values
state-repository	◆ vrf-object-base-state-repository ◆ vrf-overlay-object-state-repository ◆ vrf-object-state-repository ◆ vrf-moving-object-state-repository ◆ vrf-aggregate-state-repository ◆ fixed-wing-entity-state-repository ◆ ground-entity-state-repository ◆ human-state-repository ◆ cultural-feature-state-repository ◆ missile-entity-state-repository ◆ rotary-wing-entity-state-repository ◆ surface-entity-state-repository ◆ subsurface-entity-state-repository
net-interface	◆ aggregate-local-entity-net-interface ◆ individual-local-entity-net-interface ◆ local-environmental-net-interface ◆ pseudo-aggregate-local-entity-net-interface ◆ vrf-aggregate-remote-entity-net-interface ◆ vrf-individual-remote-entity-net-interface ◆ vrf-remote-environmental-net-interface ◆ aggregate-remote-entity-net-interface ◆ individual-remote-entity-net-interface ◆ remote-environmental-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")
  (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")
)
(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")
  (net-interface-min-tick-period  -1.000000)
  (net-interface-min-tick-period-variance  -1.000000)
  (task-manager  "")
  (component-manager  "")
  (plan-manager  "")
)
```

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 68-4 lists parameter types and the state repository type required.

Table 68-4: State repository and parameter types

State Repository Type	Parameter Type
ground-entity-state-repository	ground-vehicle-param
fixed-wing-entity-state-repository	fixed-wing-entity-param
individual-lifeform-state-repository	individual-lifeform-param
rotary-wing-entity-state-repository	rotary-wing-entity-param
human-state-repository	human-param
cultural-feature-state-repository	cultural-feature-param
missile-entity-state-repository	missile-param
surface-entity-state-repository	surface-entity-param
subsurface-entity-state-repository	subsurface-entity-param
vrf-moving-object-state-repository	moving-object-param
vrf-overlay-object-state-repository	vrf-object-param
vrf-object-state-repository	

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 B, *Object Parameters* for tables that list the network interfaces used for the simulation objects provided with VR-Forces.

68.2. 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 15-7.) 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 68.1). 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 68-13 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 component descriptions and a list of port groups, please see the online help for the OPD Editor. 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 68-16.

68.2.1. 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 68.1, 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 68.1, 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 68.1, 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 68.1, line 46) 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 68.1 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 or the online help for the OPD Editor for the names of port and port groups used by a given component.

The components that you can use are described in the online help for the OPD Editor.

Example 68.1 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 68.1

```
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       (road-center-offset  2.000000)
15     )
16     (patrol-between-points
17       (component-descriptor-type "ground-move-to-descriptor")
18       (component-type  "ground-auto-patrol-between-controller")
19       (min-tick-period  -1.000000)
20       (min-tick-period-variance  -1.000000)
21       (tick-period-uses-real-time False)
22       (process-state-repository-name  "")
23       (process-state-repository-type  "")
24       (is-enabled True)
25       (near-distance  25.000000)
26       (at-distance  1.000000)
27       (approach-speed  2.000000)
28       (road-center-offset  2.000000)
29     )
30   ) ;; end controllers
31   (actuators
32     (powertrain
33       (component-descriptor-type "automotive-component-descriptor")
34       (component-type  "automotive-actuator")
35       (min-tick-period  -1.000000)
36       (min-tick-period-variance  -1.000000)
37       (tick-period-uses-real-time False)
38       (process-state-repository-name  "")
39       (process-state-repository-type  "")
40       (is-enabled True)
41       (art-part-list )
42       (check-terrain-collisions False)
43       (use-high-fidelity-ground-clamping True)
44     )
45   ) ;; end actuators
46   (connections
47     (connect move-along:automotive-control powertrain:automotive-
         control)
48     (connect patrol-between-points:automotive-control
         powertrain:automotive-control)
49   ) ;; 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.

68.2.2. 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).

68.3. 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 OPD Editor 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.

68.3.1. Creating a System Definition File

If you want to create a new system definition file, MÄK recommends that you copy an existing file and use it as a template for adding the components that you want to configure into a system. You can use the OPD Editor 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 "MÄK Technologies Lisp (MTL) Files," on page A-9.) 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.

68.3.2. 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 68-13.) 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 68-21.)

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 68-5 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).

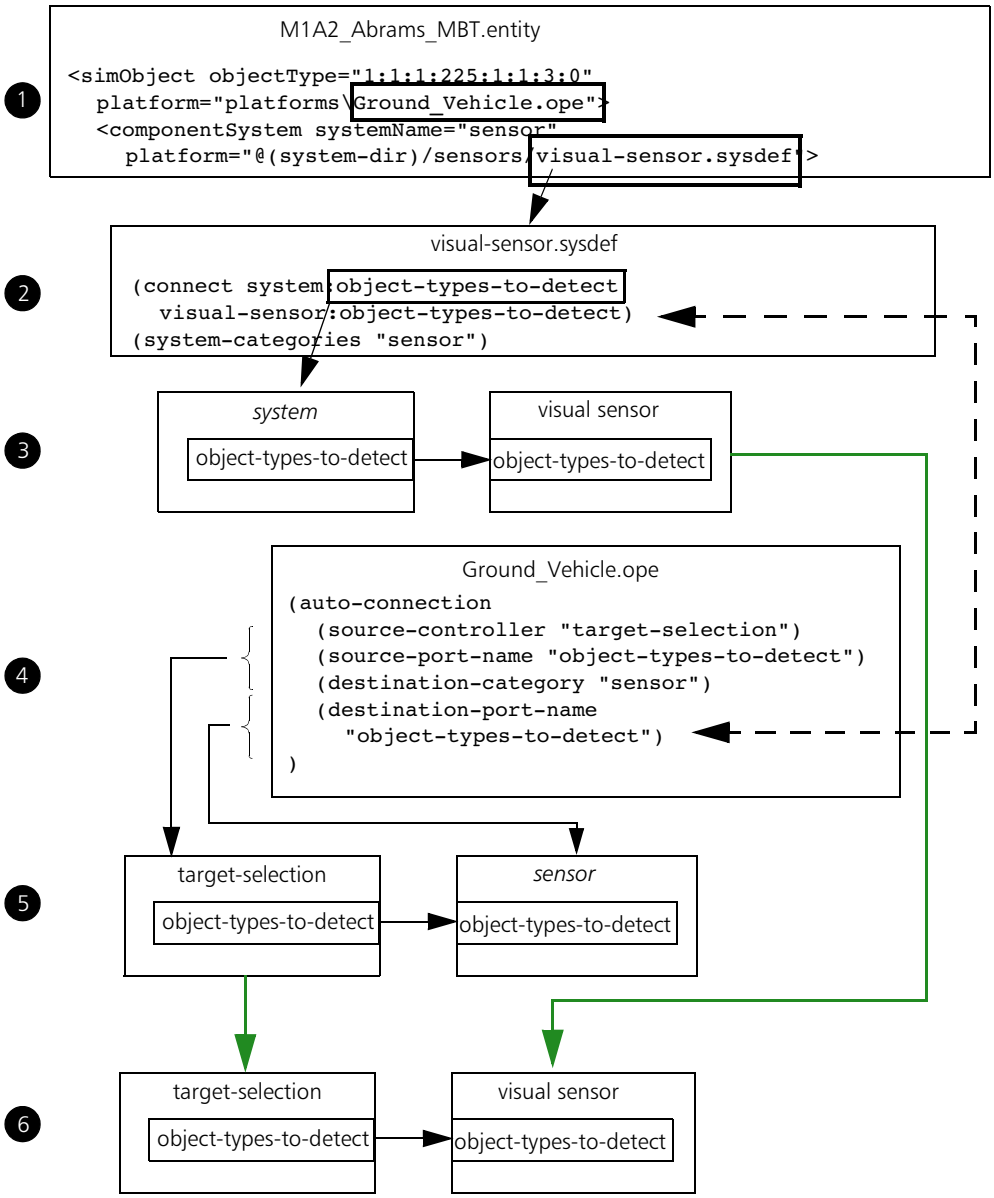


Figure 68-5. External component connection

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.

68.4. Adding Variable Bindings to Platform and System Files

Platform files and most system definition files (SDF) 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 SDF 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 SDF 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 SDF.

68.4.1. Adding a Variable Binding

The procedure for adding a variable binding is the same for platform files and SDF files.

To add a variable binding to a platform file or SDF:

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 or in the OPD Editor.

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 SDF, 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 SDFs get applied to all simulation objects that use them. For details, please see “Applying Platform Updates,” on page 63-17.

If you do not add an attribute to an *.entity* file that references the modified platform or SDF, a default value will be used, which may result in unexpected behavior.

68.4.2. Making Variables Editable in the Simulation Object Editor

When you add new variable bindings, you can make them editable in the Simulation Object Editor. Variables in platform files are edited in the Attributes panel (the right-most panel) of the Simulation Object Editor. They include variables such as speed, mass, and bounding volume. Variables in SDFs 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.

68.4.3. Adding Platform Variable Bindings to the Simulation Object Editor

Variable bindings for platform files are edited in the Attributes Panel. The layout of this panel 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.



You do not need a developers license to use Qt Designer. You will need to download the version of Qt used by your version of VR-Forces. Please see *VR-Forces 4.4.1 Release Notes* for details.

In "Adding Variable Bindings to Platform and System Files," on page 68-21, 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.

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.

68.4.4. Adding System Definition File Variables to the Simulation Object Editor

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 SDF. 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 68-21 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)
      )
...
```

68.5. 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.

68.6. 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.

68.7. 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"/>
```

68.8. Updating Object Parameter Database Files for New Releases

New releases of VR-Forces frequently include new components and new parameters for existing components. If you are using a customized SMS you will probably want to update it to take advantage of the new components and parameters. Unfortunately, there is no automatic way to migrate an object parameter database. Here are some guidelines and options for migrating your SMS:

- If you have created new simulation objects using the VR-Forces-installed system definition files, copy their *.entity* files into the *./data/simulationModel-Sets/<model_set>/vrfSim* directory. They will pick up all changes to systems, platforms, and other configuration files.



Copying *.entity* files does not copy changes made to element definitions and entity type mappings. You will have to update element definitions and entity type mappings using the Simulation Object Editor or the Visual Models Editor and Entity Type Mappings dialog box.

- If you have added new systems using the VR-Forces-installed components, copy the system definition files and any other configuration files, such as damage files or detection files, to the appropriate directories in the simulation model set.
- If you have added new simulation objects or systems that rely on new components created with the VR-Forces API, you will have to rebuild the components using the new version of VR-Forces and copy all required files as described in the previous paragraphs.
- Use multiple simulation model sets. VR-Forces can load multiple simulation model sets (for details, please see "Specifying Multiple Simulation Model Sets," on page 7-8). If your customizations do not overlap the default SMS or are relatively easy to separate from the default SMS, you can create a separate SMS and load it along with the SMS supplied with VR-Forces.
- Include the VR-Forces-supplied SMSs in your customized SMS. For information about including SMSs in other SMSs, please see "Including Simulation Model Sets in other Simulation Model Sets," on page 63-6.



69. Using the OPD Editor

This chapter describes how to use the OPD Editor to edit simulation model sets (SMS).

The Object Parameter Database Editor 69-2
Starting the OPD Editor 69-2
Loading a Simulation Model Set 69-3
Editing Platforms and Systems 69-4
- Creating a New Platform or System from an Existing One 69-5
- Editing Individual Parameters 69-6
- Deleting Parameters and Components 69-7
- Restoring Parameters 69-7

Adding New Components and Parameters 69-8
- Adding New Components 69-8
- Changing A Component's Priority 69-9
- Creating A Connection 69-10
- Adding New Resource Parameters 69-11

Editing Variables 69-11
Regenerating Platform Files 69-12
Saving Changes to the Object Parameter Database 69-12

69.1. The Object Parameter Database Editor

The Object Parameter Database Editor (OPD Editor) is a graphical user interface for editing platforms and systems. When you use the OPD Editor instead of a text editor, you do not have to worry about getting MTL or XML syntax correct in your files. It also quickly gives you access to all the platforms and systems without needing to know which particular files they use.

If you want to edit or add simulation objects, use the Simulation Object Editor.

69.2. Starting the OPD Editor

➤ To start the OPD Editor, choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.4.1** → **Tools** → **OPD Editor**. The OPD Editor opens (Figure 69-1).

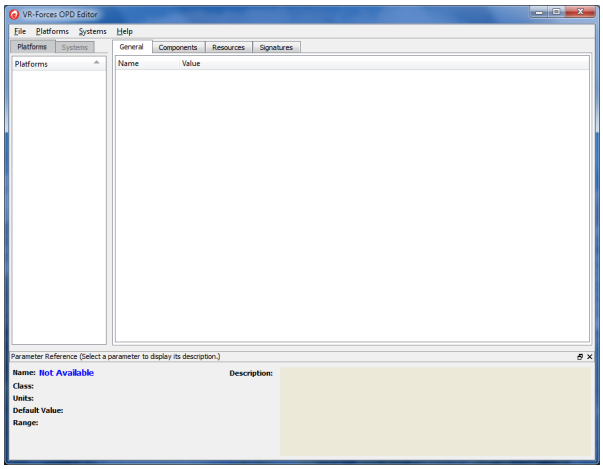


Figure 69-1. OPD Editor

To start the OPD Editor from the Windows or Linux command line:

1. Change directory to *./bin64*.
2. Run *vrfOpdEditor*.

69.3. Loading a Simulation Model Set

To edit a simulation model set, you must load it into the editor.

To load a simulation model set:

1. Choose **File** → **Open Simulation Model Set**. An Open dialog box opens. By default it opens to *./data/simulationModelSets*.
2. Select the SMS you want to load, for example, *EntityLevel.sms*.
3. Click Open. The SMS is loaded. Platforms are listed on the Platforms tab, systems on the Systems tab.

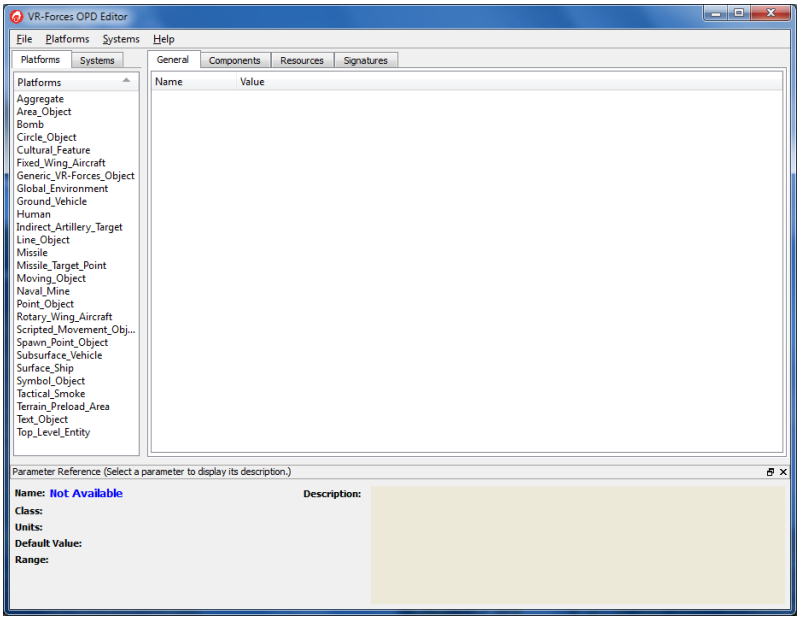


Figure 69-2. OPD Editor with SMS file loaded

69.4. Editing Platforms and Systems

Platforms and systems are similar in organization and construction. They simply configure objects at different levels. They have a set of general parameters, components, and possibly resources, signatures, subordinate objects, and a soil list. These parameters are organized into tabs. Within each tab, the parameters and their values are listed.

This section explains how to edit an object's parameters, how to delete parameters, and how to restore parameters that you have changed or deleted.

Parameter definitions are documented by component in the online help for the OPD Editor in the Object Parameter Descriptions folder. Individual parameter definitions are displayed in the Parameter Reference panel of the OPD Editor. The definitions are also in the header files and class documentation.

69.4.1. Creating a New Platform or System from an Existing One

You can add new platforms and systems to the object parameter database using an existing one as a template.

To add a new platform based on an existing platform:

1. Select the platform on which you want to base the new one.
2. Choose **Platforms** → **Create Platform From Existing**. The Create New Platform dialog box opens.
3. Type a name for the new platform.
4. Click OK.
5. Edit the parameters.
6. Save the object parameter database.

To add a new system based on an existing system:

1. Select the system on which you want to base the new one.
2. Choose **Systems** → **Create System From Existing**. The Create New System dialog box opens.
3. Type a new name for the system definition file or click Browse and select an existing file name.
4. Type a name for the new system.
5. Optionally, specify configuration files to copy:
   a. Click the >>> button. The dialog box expands to display the Copy Files group box.
   b. Select the file types that you want to copy.
6. Click OK.
7. Edit the parameters.
8. Save the object parameter database.

69.4.2. Editing Individual Parameters

The various parameters in the object parameter database are all edited essentially the same way. A parameter's value is a number or string that you type, a value that you select from a list, or a filename that you edit by hand or select from a file chooser. When you edit a parameter, the name of the parameter and all its parents are displayed in bold (Figure 69-3).

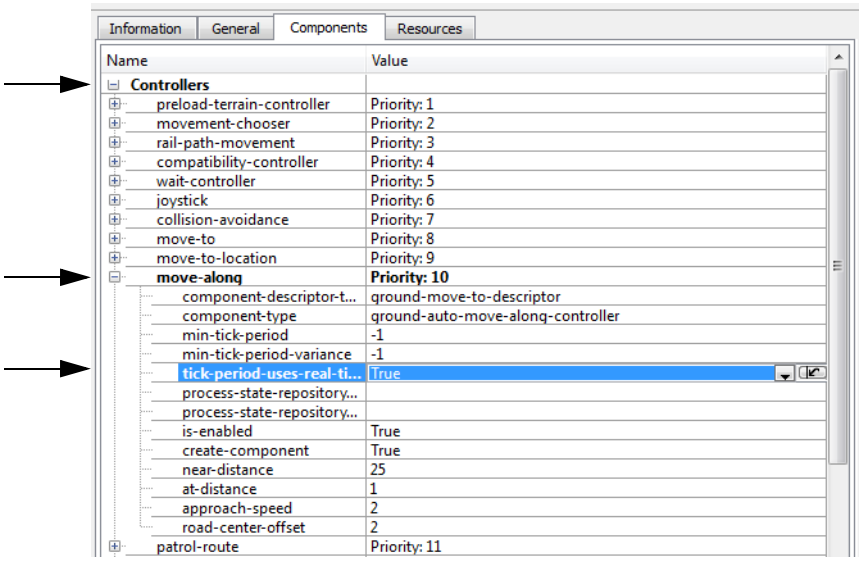


Figure 69-3. Parameter hierarchy after edits

To edit a parameter:

1. Select the parameter by clicking in the Name or Value column. The Value field becomes editable.
2. Enter the new value. In most cases you enter a new value by typing it. If a parameter value has a File Chooser button (Figure 69-4), click the button to open a File Chooser dialog box. If the value has an Edit button, click the button to open a window that allows you to edit the file.

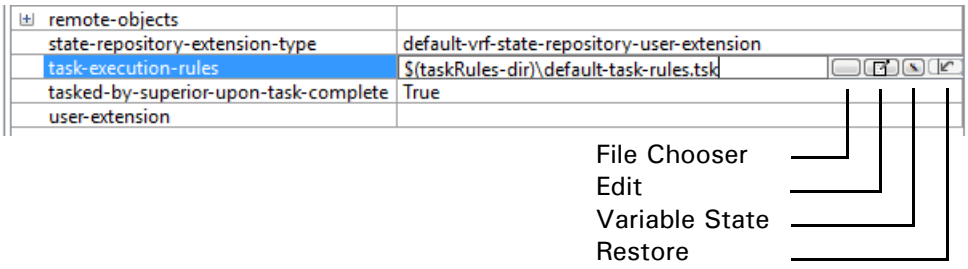


Figure 69-4. Parameter value buttons

69.4.3. Deleting Parameters and Components

The parameters on the General tab cannot be deleted. Parameters on the other tabs can be deleted.

To delete a parameter or component:

1. Select the parameter or component in the Value list.
2. Click the Delete button at the right end of the parameter entry (Figure 69-5).



Figure 69-5. Delete parameter button

69.4.4. Restoring Parameters

If you change a value or delete a parameter or component, you can restore the original value. You can restore at the individual parameter level and at any level in a given hierarchy of entries (Figure 69-3). For example, you can restore a change to the **at-distance** parameter for the **move-to** controller, all edits made to the **move-to** controller, or all edits made to all Controllers. However, since Actuators are not part of the Controllers hierarchy, restoring a change at the Actuators level would not affect changes to Controllers.

To restore a parameter or component:

1. Select the parameter to restore. If you are restoring a deleted parameter or component, select the parent component.
2. Click the Restore button (Figure 69-4).
3. To undo all changes to a platform or system, choose **Platforms** → **Restore Platform** or **Systems** → **Restore System**.

69.5. Adding New Components and Parameters

You cannot add new parameters to the General tab. You can add new components, resources, subordinates, and signatures if an object supports them.



For most platforms, the Components tab does not list all of the components that apply to the entity. This is because many components are referenced through the systems that are attached to an instance of a platform. If you do not see a component that you are looking for in a platform's component list, look at the component list for the relevant system in the Systems list.

69.5.1. Adding New Components



Before you add a component to a platform, consider whether or not the component should be added as part of a new or existing system, rather than being added directly to the platform.

To add a component to a platform:

1. Select the Components tab.
2. Select the component type (Actuator, Controller, Sensor) you want to add.
3. Click the Add button (Figure 69-6). The Create New Component dialog box opens. The Component list lists all components that are available for this object.

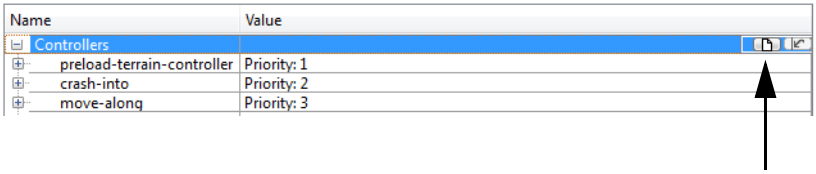


Figure 69-6. Add button

4. From the Component list, select the component you want to add.
5. Click OK. The component is added to the object.

69.5.2. Changing A Component's Priority

When you change a component's priority, the other components are reprioritized as needed. All affected components are listed in bold.

To change a component's priority:

1. Select the component you want to edit.
2. Click the up or down arrow on the value line (Figure 69-7).



Figure 69-7. Change Priority buttons

69.5.3. Creating A Connection

When you add a component, you must also add connections between the component and other components it interacts with.

To add a connection:

1. On the Components tab, select Connections.
2. Click the Add button (Figure 69-6). The Create Connection dialog box opens (Figure 69-8).

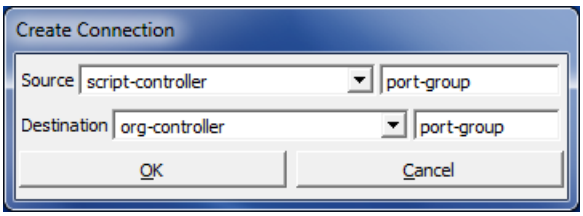


Figure 69-8. Create Connection dialog box

3. Choose a source component from the Source list.
4. Enter the port group for the source component. You must know the name of the port group you need to use. For a list of components and port groups, see the online help.
5. Choose the destination component from the Destination list.
6. Enter the port group for the destination component. You must know the name of the port group you need to use.
7. Click OK. The connection is added to the Connections list.

69.5.4. Adding New Resource Parameters

To add a resource parameter to an object:

1. Select the Resources tab.
2. Select Resources.
3. Click the Add button (Figure 69-6). The Create Resource dialog box opens.

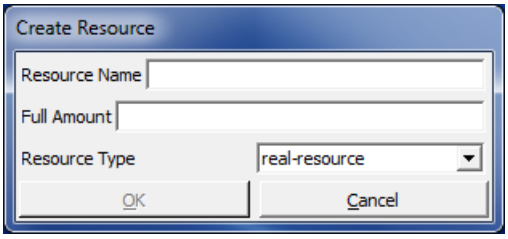


Figure 69-9. Create Resource dialog box

4. In the Resource Name box, enter the name of the resource you want to add. Resource names must start with a letter, not a number. You are responsible for knowing that this is a valid resource supported by your implementation of VR-Forces.
5. In the Full Amount box, enter the amount of the resource to assign to this object.
6. Select a resource type from the Resource Type list.
7. Click OK. The resource is added to the object.

69.6. Editing Variables

The object parameter database uses variables to specify the location of supporting files, such as ammo selection files, system definition files, and formation files. You can edit these variables. (For information about variables, please see "Adding Variable Bindings to Platform and System Files," on page 68-21.)

To edit variables:

1. Select the platform whose variables you want to edit.
2. Choose **Platforms** → **Edit Variables**. The Edit Variables dialog box opens.
3. Change the variable values as desired.
4. Click OK.

69.7. Regenerating Platform Files

If a developer adds new parameters to a component that need to be set in a platform file, the parameters need to be added to the platform file. The OPD Editor can add new parameters to all platform files in one operation.

➤ To regenerate platform files, choose **Platforms** → **Regenerate Platform Files**.

69.8. Saving Changes to the Object Parameter Database

➤ To save an edited object parameter database, choose **File** → **Save**.



70. SMS Configuration Files

This chapter describes configuration files that affect simulation object behavior and the effects of munitions.

Detection Tables 70-2
Probability and Selection Tables for Munitions 70-3
Ammo Select Tables 70-3
Hit Probability Tables 70-4
Damage Probability Tables 70-6
Configuring an Entity to Take Damage from a Munition 70-7
Configuring Formations 70-8
Formation Files 70-9
Creating a User-Defined Formation 70-10
Configuring Emitters 70-11
Configuring Radios 70-13
Publishing Radio Transmitters 70-13
Configuring Task Execution Rules 70-14
Configuring Action Categories 70-15
Configuring Obstruction Avoidance 70-15
Configuring Object Types 70-15
Configuring Vector Feature Types 70-16
Configuring Embarkation 70-16
Configuring the Effects of Suppressive Fire 70-17
Configuring VR-Forces Components on Remote Entities 70-18

70.1. 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 70-1, 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	|	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 70-1. 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 9-13.

70.2. Probability and Selection Tables for Munitions

When the entities in an entity-level scenario shoot at each other, VR-Forces uses an ammo select table to select the ammunition to be used. It uses a set of tables to calculate the probability of:

- Hitting a target (hit probability table).
- Being damaged when hit by hostile munitions (damage probability table).

70.2.1. 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 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 wild-cards.
- **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.
  - **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:

```
(ammo-select-table
   (default ;; matches any ground vehicle
      (target-type 1 1 -1 -1 -1 -1 -1)
      (ammo
         (M830-HEAT-120mm ;; these names need to match resource names
            (munition-type  2 2 225 2 13 3 0)
            (warhead  0)
         )
         (M829A1-AP-120mm ;; these names need to match resource names
            (munition-type  2 2 225 2 13 2 0)
            (warhead  0)
         )
      )
   )
   (battle-tank ;; matches any tank
      (target-type 1 1 -1 1 -1 -1 -1)
      (ammo
```

```
            (M829A1-AP-120mm ;; these names need to match resource names
               (munition-type  2 2 225 2 13 2 0)
               (warhead  0)
            )
            (M830-HEAT-120mm ;; these names need to match resource names
               (munition-type  2 2 225 2 13 3 0)
               (warhead  0)
            )
         )
      )
   )
```

70.2.2. 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)
        )
      )
    )
  )
)
```

70.2.3. Damage Probability Tables

The damage actuator uses damage probability tables specified by its component descriptor to determine direct and indirect fire damage based on several factors. 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.

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.

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), and 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 70-4. The damage actuator provided by MÄK looks for a probability entry with the name **catastrophic-kill**, **mobility-kill**, or **firepower-kill**.



The range for missiles is the range from the detonation location to the target, not from the entity that fired the missile.

The following damage probability table fragment has two surface entries, front and rear. The first uses a coefficients range determinant, the second surface entry uses a **range-list** determinant.

```
(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)
        )
      )
    )
  )
  (rear
    (angle-of-incidence
```

```
        (angle  0.523600)
        (range-determinant
          (range-list
            (range 2000.000000
              (catastrophic-kill  0.800000)
            )
            (range 3000.000000
              (catastrophic-kill  0.600000)
            )
            (range 4000.000000
              (catastrophic-kill  0.300000)
            )
          )
        )
      )
      (angle-of-incidence
        (angle  1.047200)
        (range-determinant
          (range-list
            (range 2000.000000
              (catastrophic-kill  0.800000)
            )
            (range 3000.000000
              (catastrophic-kill  0.600000)
            )
            (range 4000.000000
              (catastrophic-kill  0.300000)
            )
          )
        )
      )
    )
  )
```

The structure of a collateral damage probability file is similar to the direct damage file. However the file is processed differently. VR-Forces calculates a range around the munition impact within which other simulation objects can be damaged.

70.2.4. Configuring an Entity to Take Damage from a Munition

If you want an entity to take damage from a particular type of munition, it must have an ammo-entry block for that type of munition in its damage-component. For example, the individual lifeform damage actuator is not configured with an entry for M107-155mm, which means it would not take damage from artillery. If you want to make individual lifeforms vulnerable to this type of munition, you must add an entry for this munition type to the entity's damage model.

70.3. 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>
```

70.3.1. 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.

70.3.2. 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.

70.4. 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/simulationModel-Sets/<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
      ...
    )
  )
)
```

70.5. 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)
  )
)
```

70.5.1. 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.

70.6. Configuring Task Execution Rules

Each simulation object OPE file contains an entry for a task execution rule configuration file, named **task-execution-rules**. (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.

70.6.1. Configuring Action Categories

In scripted tasks, action categories serve the same function as task rules for enforcing concurrency. Action categories are configured in *./data/simulationModel-Sets/<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/simulationModel-Sets/<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.

70.7. 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.

70.7.1. 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
)
```

70.7.2. 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/featureconfig.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")
  )
)
```

70.8. 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 "Configuring Embarkation," on page 64-33.

70.9. Configuring the Effects of Suppressive Fire

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.

70.10. Configuring VR-Forces Components on 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 32-22.

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.



71. Configuring Sensors

This chapter explains how to configure sensors and sensor signatures.

Configuring Sensors 71-2
Adding a New Sensor to a Simulation Object 71-2
Editing a Sensor System Definition File 71-4
Specifying a Sensor Signature 71-5
Adding Detection Types or Why Doesn't My Sensor Detect Anything? 71-9
Adding a New Sensor Type 71-9
Illumination and Time of Day 71-12

71.1. 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/simulationModelSets/<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.

71.1.1. 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 MÄK support auto-connections), its connections are handled automatically. (For details, please see "Configuring System Connections," on page 68-18.)

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)
```

71.1.2. Editing a Sensor System Definition File

The preferred method for editing a sensor system definition file, as with all system definition files, is to edit it in the OPD Editor. You can also edit the file by hand. 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.

71.1.3. 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 that 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 71-12.

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. You can edit the values of sensor-signatures and add new sensor signatures in the OPD Editor.

Table 71-1 lists the sensor signatures that VR-Forces uses for general categories of simulation objects in *EntityLevel.sms*. (These categories do not match the categories 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 71-1: Entity-level sensor signatures (Sheet 1 of 3)

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
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

Table 71-1: Entity-level sensor signatures (Sheet 2 of 3)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
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
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

Table 71-1: Entity-level sensor signatures (Sheet 3 of 3)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
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 camouflage)	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
Military Small vehicle (Jeep/Technical)	1	1	2	N/A	N/A	N/A
Military Micro (human sized or smaller)	0.5	0.25	1	N/A	N/A	N/A

71.1.4. Adding Detection Types or Why Doesn't My Sensor Detect Anything?

Sometimes VR-Forces users add a sensor to a simulation object and then wonder why it doesn't detect anything. This problem occurs when a simulation object does not have a weapon. In this case, the sensor's `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))
)
```

71.1.5. 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 *physicalWorldParams.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 71-2. 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)
)
```

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.

71.1.6. 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 71-5 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.



72. Configuring Joystick and Keyboard Controls

This chapter explains how to configure joysticks and the keyboard to control entity movement.

Configuring Joysticks and Keyboard Control 72-2
Creating a Joystick Configuration 72-3
Mapping the Joystick Stick 72-4
Mapping Joystick Buttons and Keyboard Keys 72-6
Configuring “Switch Between” Options 72-9
Specifying a Favorite Joystick Configuration 72-11

72.1. 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 sysdef file, the Joystick Configuration page (Figure 72-1) has an Automotive Driving function with controls for brake, change-gear, steering, and throttle.

72.1.1. 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 72-1).

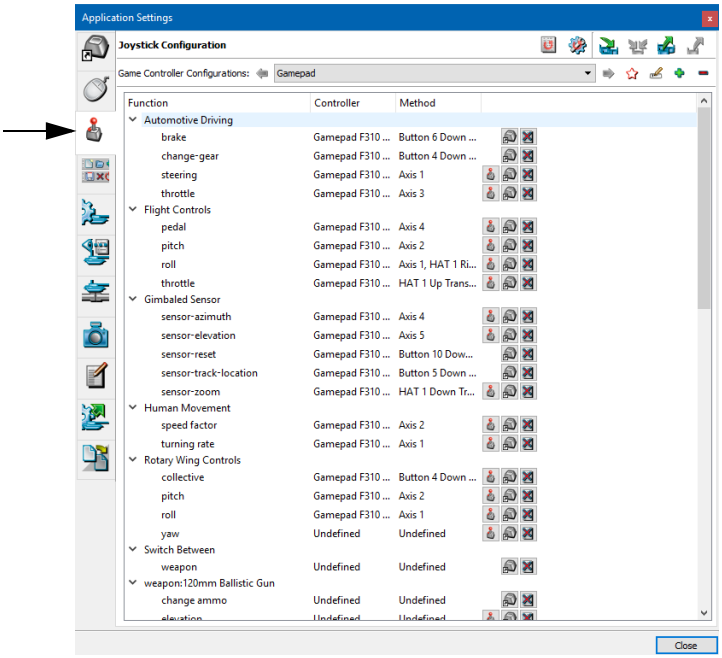


Figure 72-1. Joystick Configuration page

4. Click the Add button (✚). The New Configuration dialog box opens.
5. Type a name for the configuration.
6. Click OK. The window redisplays to show the functions that this device can control.
7. Edit the joystick and keyboard mappings as described in "Mapping the Joystick Stick," on page 72-4 and "Mapping Joystick Buttons and Keyboard Keys," on page 72-6.

72.1.2. 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 72-3.
2. On the Joystick Configuration page (Figure 72-2), click the Joystick button ( ) for the function you want to map. A joystick configuration dialog box opens (Figure 72-3).

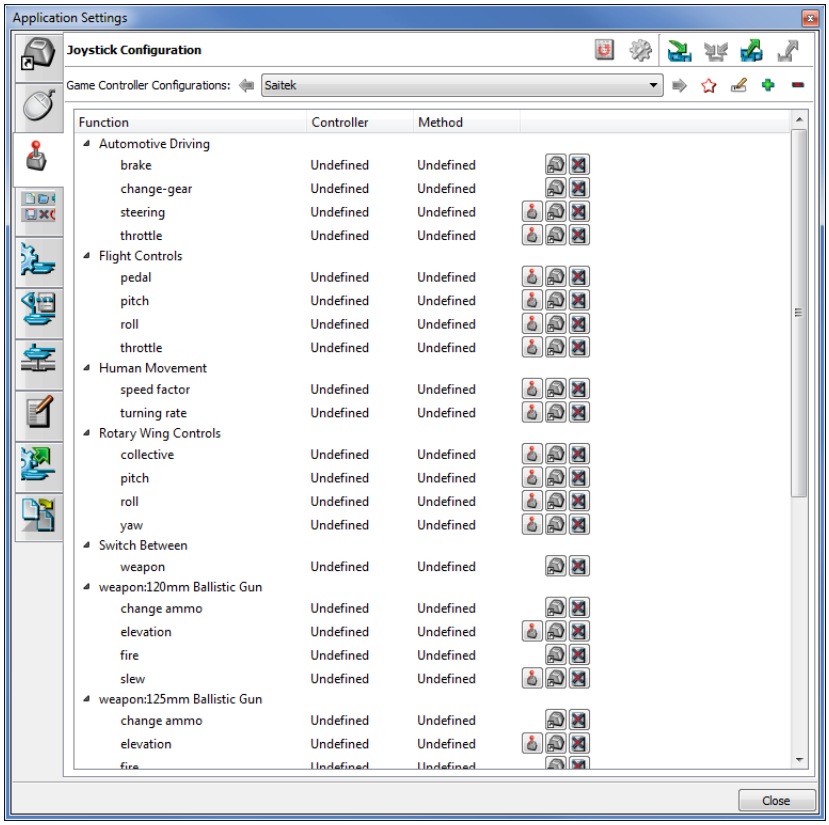


Figure 72-2. Joystick Configuration page

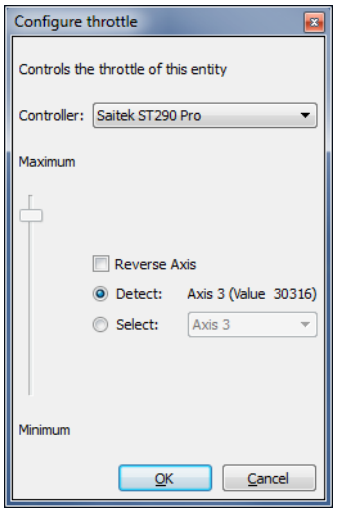


Figure 72-3. Joystick configuration dialog box

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.

72.1.3. 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 72-3.
2. On the Joystick Configuration page (Figure 72-2), 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 72-4), or just be a toggle (Figure 72-5).

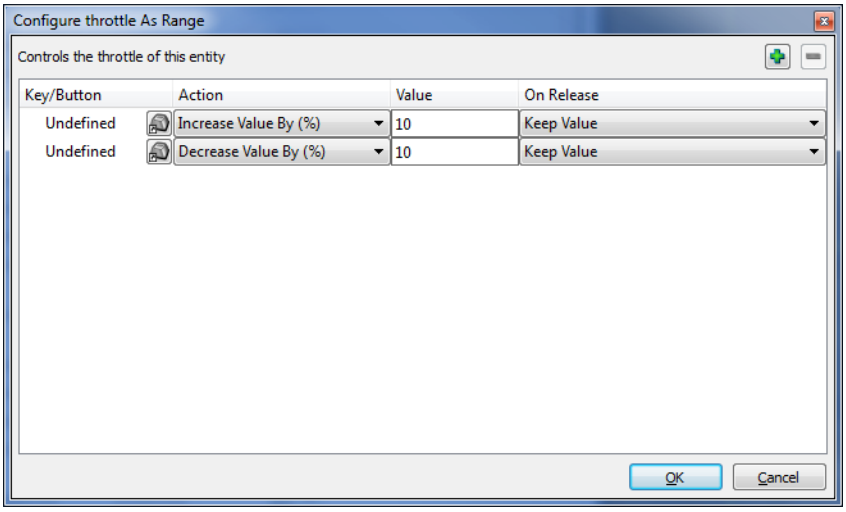


Figure 72-4. Key mapping configuration dialog box

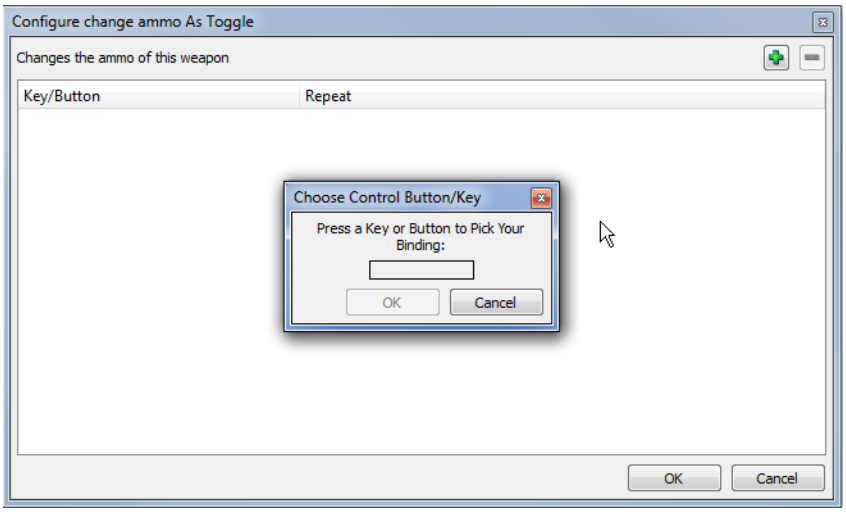


Figure 72-5. Key mapping configuration as toggle dialog box

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 72-6).

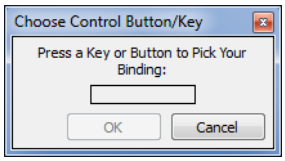
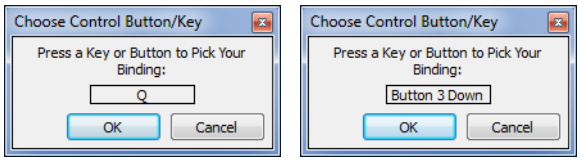


Figure 72-6. Choose Control Button/Key dialog box

   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 72-7). (If a key binding is not available, the OK button does not become activated.)



Key Joystick button

Figure 72-7. 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 72-8.

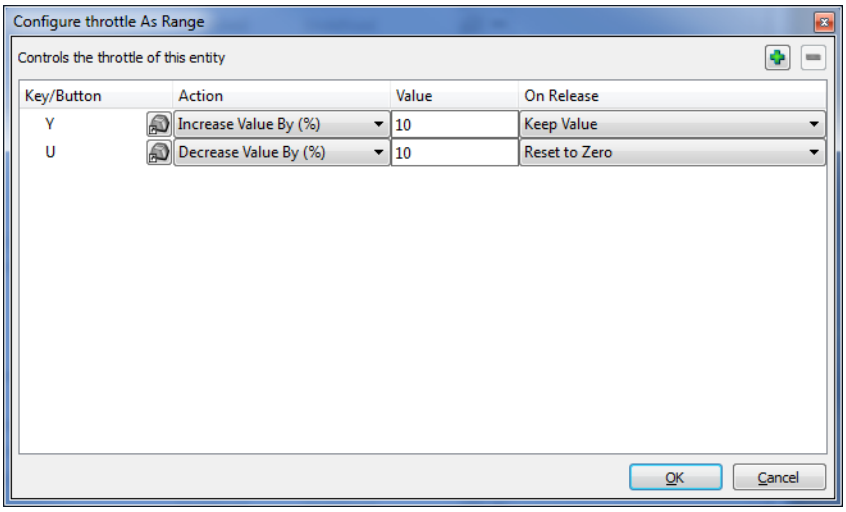


Figure 72-8. Key mapping configuration dialog box

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.

72.1.4. 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 72-6. 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 *120mm-gun.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  "120mm Gun")
  (system-description  "Turreted 120mm gun, typical of tanks such as
  the M1A2. Fires M829A1-AP and M830-HEAT rounds. Targets ground
  vehicles.")
  (allowed-state-repository-types "ground-vehicle-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  "")
      (default-value  "weapon:120mm Ballistic Gun")
    )
    (string-parameter-data
      (parameter-name  "ballistic-gun-group")
      (variable-type  "DtRwString")
      (display-label  "Ballistic Gun Joystick Group Name")
      (display-units  "")
      (source-units  "")
      (default-value  "weapon:120mm Ballistic Gun")
    )
...
```

Other weapon systems are configured similarly. On the Joystick Configuration page (Figure 72-9), the Switch Between group lists weapon as an option. The weapons configured for switching all show the `switch_option:sysdef syntax`.

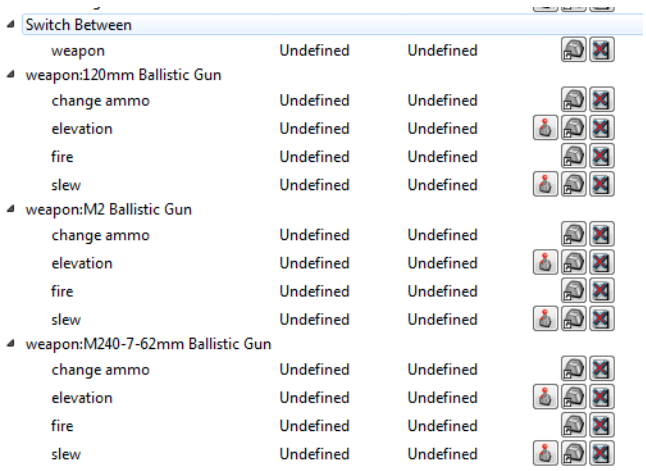


Figure 72-9. Switch between option

For details about using the switch between option, please see "Switching Between Systems," on page 19-20.

72.1.5. 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 72-1).
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.



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.



73. 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 73-2
- Window Types 73-2

Managing Display Engine Configurations 73-3
- Adding a Window 73-3
- Adding a Channel to a Window 73-5
- Removing a Window 73-6
- Removing a Channel 73-6
- Saving a Display Engine Configuration 73-6
- Loading a Display Engine Configuration 73-6

Changing a Window's Attributes 73-7

Changing a Channel's Attributes 73-8
- Setting the Clipping Planes 73-10
- Specifying the Projection Resize Policy Attribute 73-12
- Changing a Channel's Frustum (Field of View) 73-15
- Changing the Viewport 73-16
- Configuring Water Visibility 73-17

Configuring Multichannel Displays 73-18
- Changing the Camera's Position and Orientation Offset 73-20

Stereoscopic Displays 73-21
- Configuring Anaglyphic Stereo 73-22
- Configuring Polarized Stereo 73-23

73.1. 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.

73.1.1. 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 46-2.)
- 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.

73.2. 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.

73.2.1. 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 73-1).

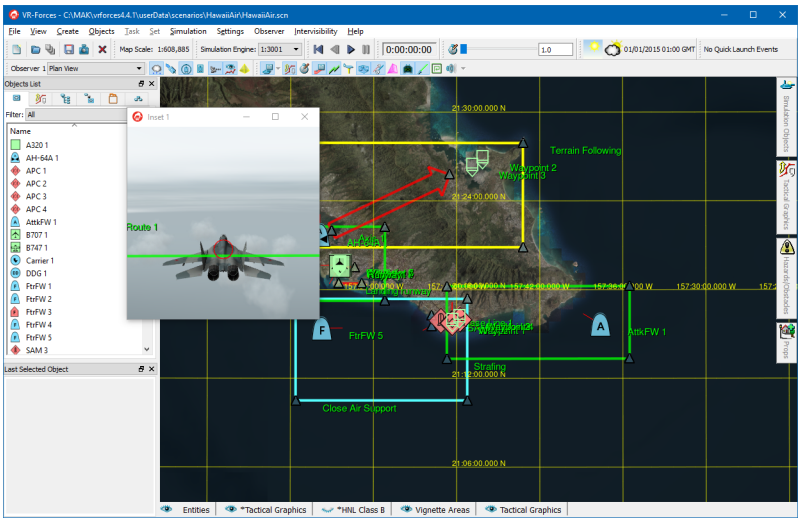


Figure 73-1. New window

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 73-2).

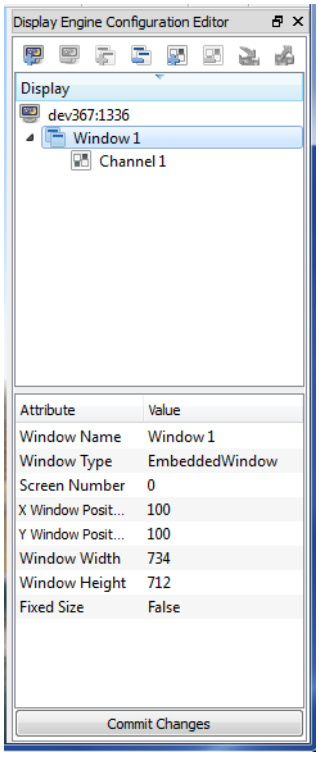


Figure 73-2. Display Engine Configuration Editor Panel

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 73-3).

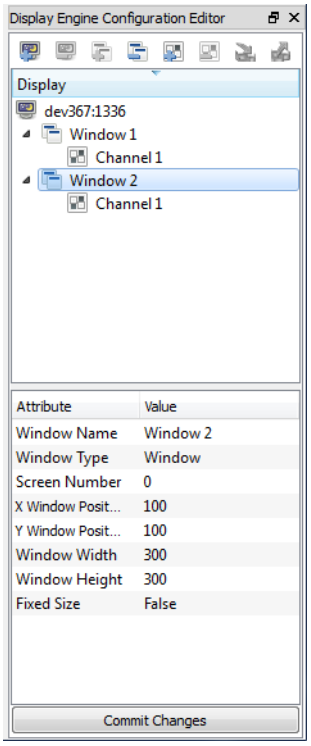


Figure 73-3. New window

73.2.2. 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 (Figure 73-10). For details about editing channel attributes, please see "Changing a Channel's Attributes," on page 73-8.

73.2.3. 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.

73.2.4. 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.

73.2.5. 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.

73.2.6. 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.

73.3. 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 73-2).
- 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 73-2).
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.

73.4. 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.
- 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".

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 73-4).

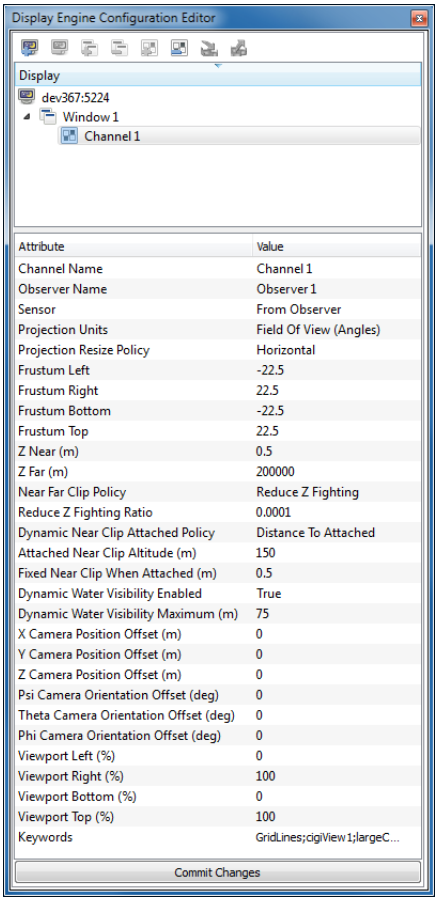


Figure 73-4. Channel attributes

2. Click the value of the attribute that you want to change.
3. To change a numeric value, type or select a new value.

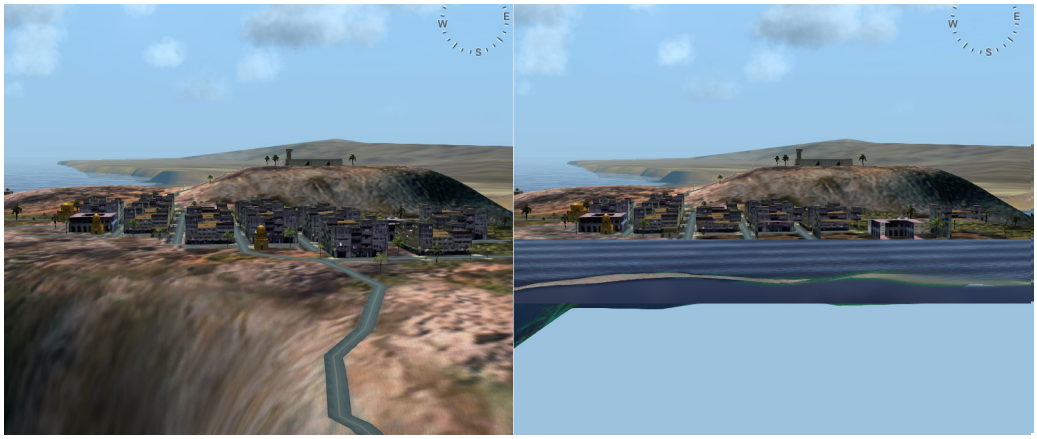
   To change the Observer Name, Sensor, Project Units, Projections Resize Policy, or Parent Channel, select a value from the list.

   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.

73.4.1. Setting the Clipping Planes

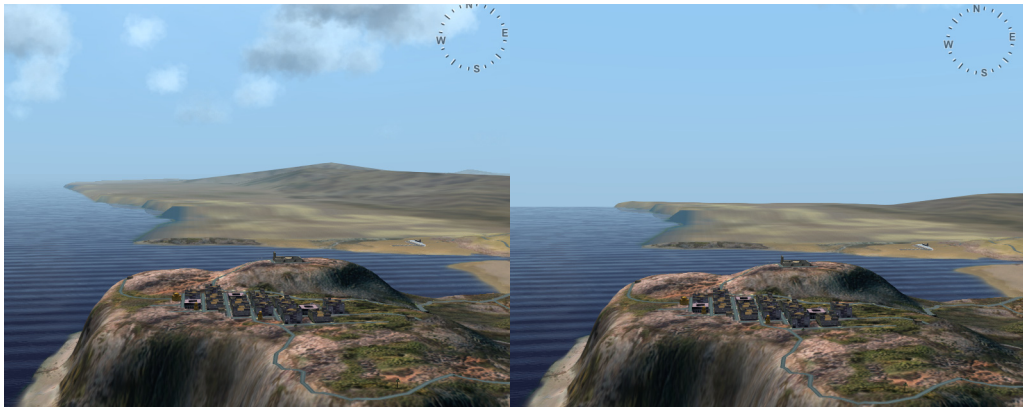
To improve performance, the graphics engine does not render anything that falls outside of a specified range of distances from the observer. This range is determined by the near and far clipping planes. Clipping planes are set on a per-channel basis.

Figure 73-5 illustrates the effect of changing the near clipping plane (the Z Near attribute). Figure 73-6 illustrates the effect of changing the far clipping plane (the Z Far attributes).



Z Near = 1 Z Near = 500

Figure 73-5. Near clipping



Z Far = 10000000 Z Far = 10000

Figure 73-6. 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.

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.

73.4.2. 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. Maintain the horizontal aspect ratio when the window is resized.
- Vertical. Maintain the vertical aspect ratio when the window is resized.

Figure 73-8 illustrates the effect of different resize policies, as follows:

- In the window with the fixed resize policy, all of the visual data in the initial view is still in the resized view, although the vertical dimensions are distorted.
- In the window with the horizontal resize policy, the field of view is widened to maintain the correct aspect ratio of the visual data.
- In the window with the vertical resize policy, the field of view is shortened.

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.

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 73-7).

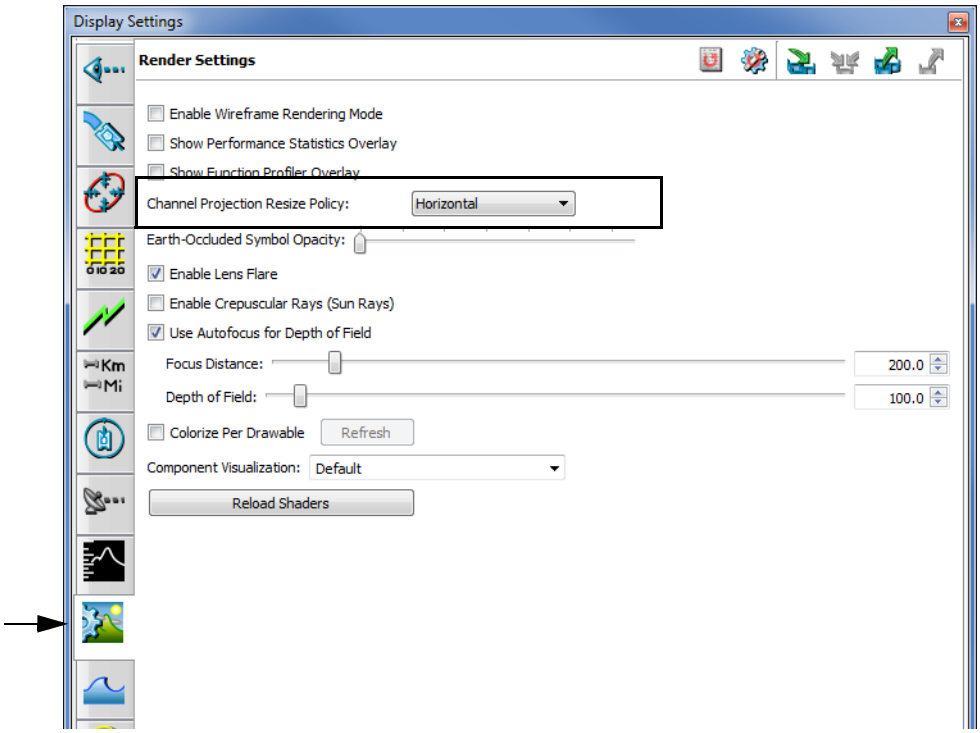
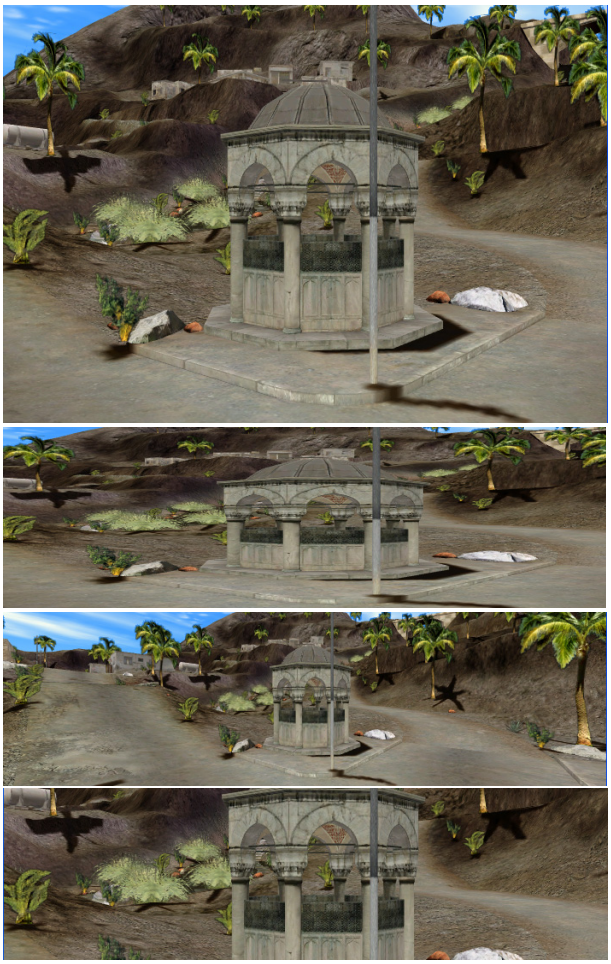


Figure 73-7. Display Settings, Render Settings page

3. Select an option on the Channel Projection Resize Policy list.



Initial view

Resized with Fixed resize policy

Resized with Horizontal resize policy

Resized with Vertical resize policy

Figure 73-8. Effect of different projection resize policies

73.4.3. 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.

Figure 73-9 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.

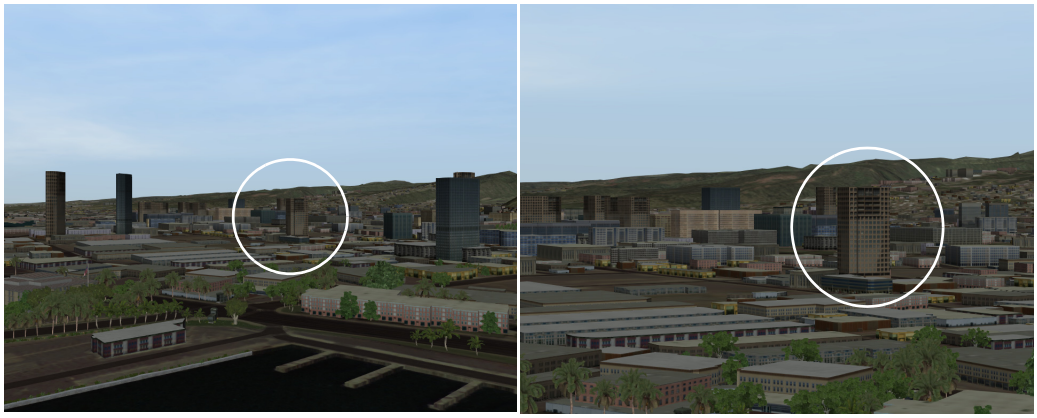


Figure 73-9. Field of view

➤ To change a channel's field of view, edit the frustum values in its attributes list.

73.4.4. 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 73-10 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.

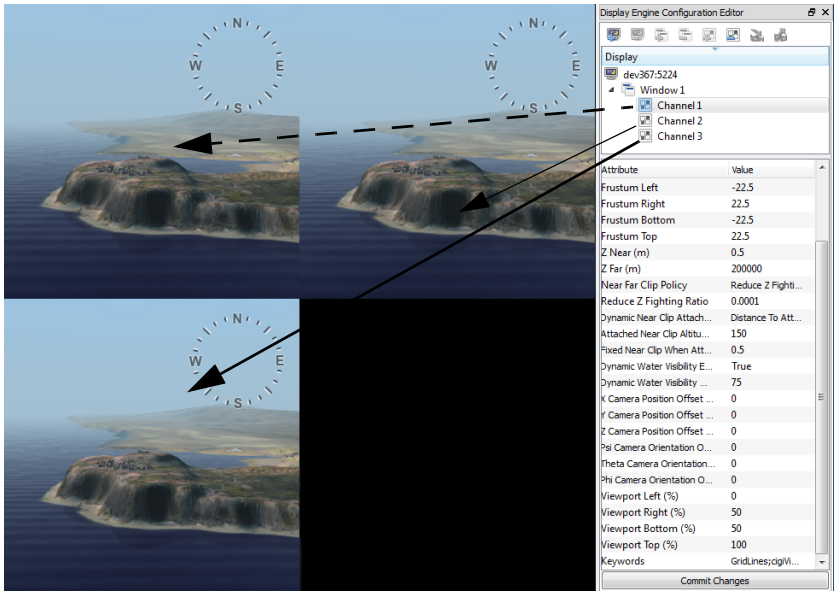


Figure 73-10. Window with three channels and three viewports

Viewports are specified as a percentage of the window from left to right and top to bottom. Figure 73-11 shows the percentage allocation for the viewports in (Figure 73-10).

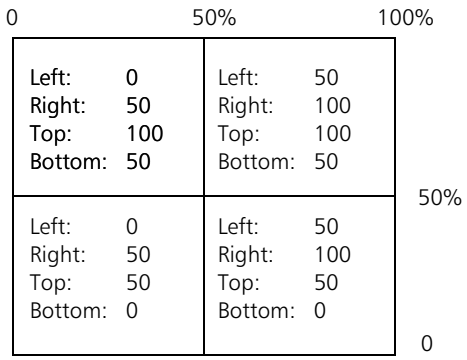


Figure 73-11. Viewport configuration



When you change viewports, the window might not completely refresh. To see the changes, resize the window.

73.4.5. 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 is used to increase transparency automatically as the observer's altitude increases. Dynamic Water Visibility Maximum specifies the maximum depth to which transparency is added (Figure 73-12).

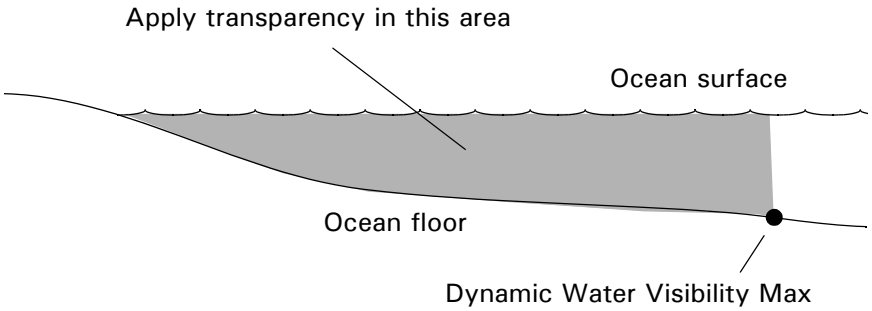


Figure 73-12. Dynamic water visibility

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 here 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 50-16.)

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.

73.5. 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 73-13).

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.

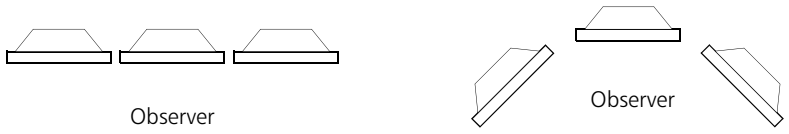


Figure 73-13. Multi-channel displays (top-down view)

By default, all windows and channels show the same view of the terrain (Figure 73-15, 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 73-14 illustrates the default frustum for Channel 1. It has a 60° field of view, from -30° to +30°.

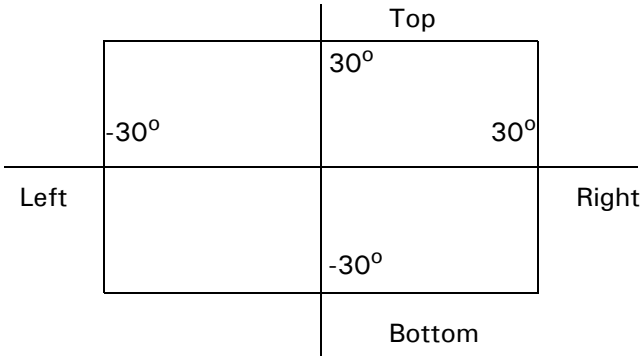


Figure 73-14. Default frustum for Channel 1

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 73-15. The field of view is the same: -30° to +30°. However, now each channel shows a 20° segment of the total field of view.

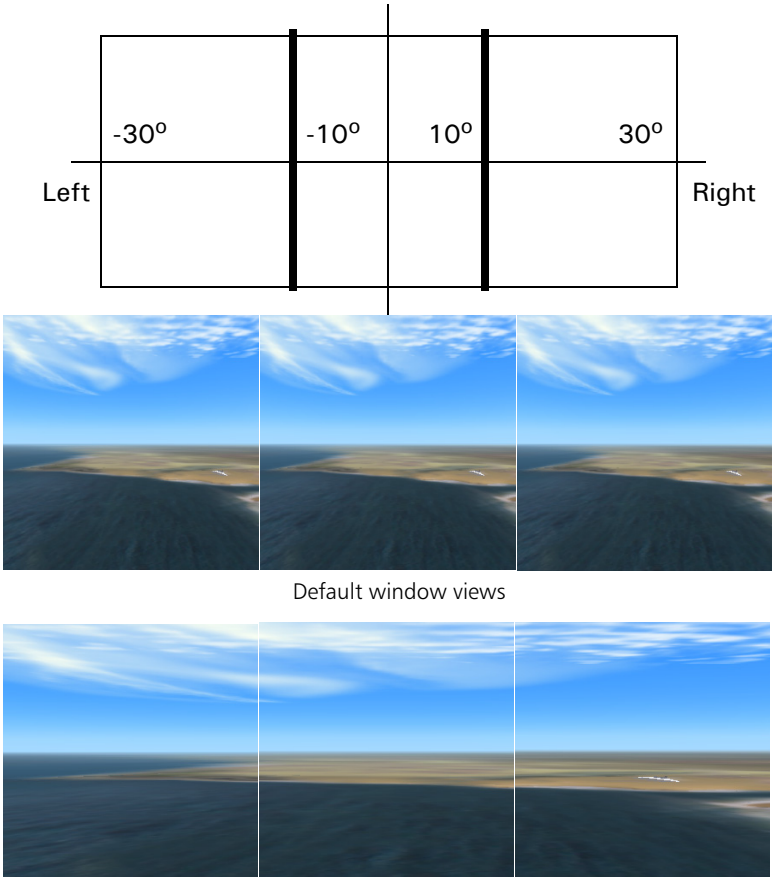


Figure 73-15. Frustum values for three channels

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.

73.5.1. 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 73-16. (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).

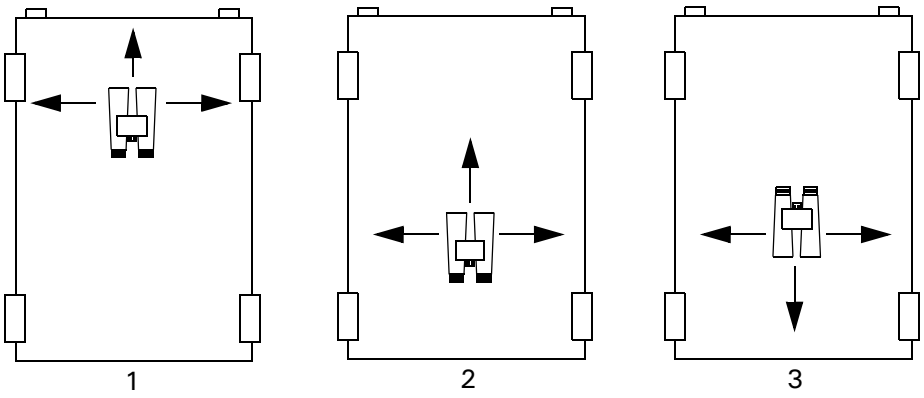


Figure 73-16. Position and orientation offsets

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).

73.6. 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.

73.6.1. 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 73-1.

Table 73-1: Environment variables for anaglyphic stereo

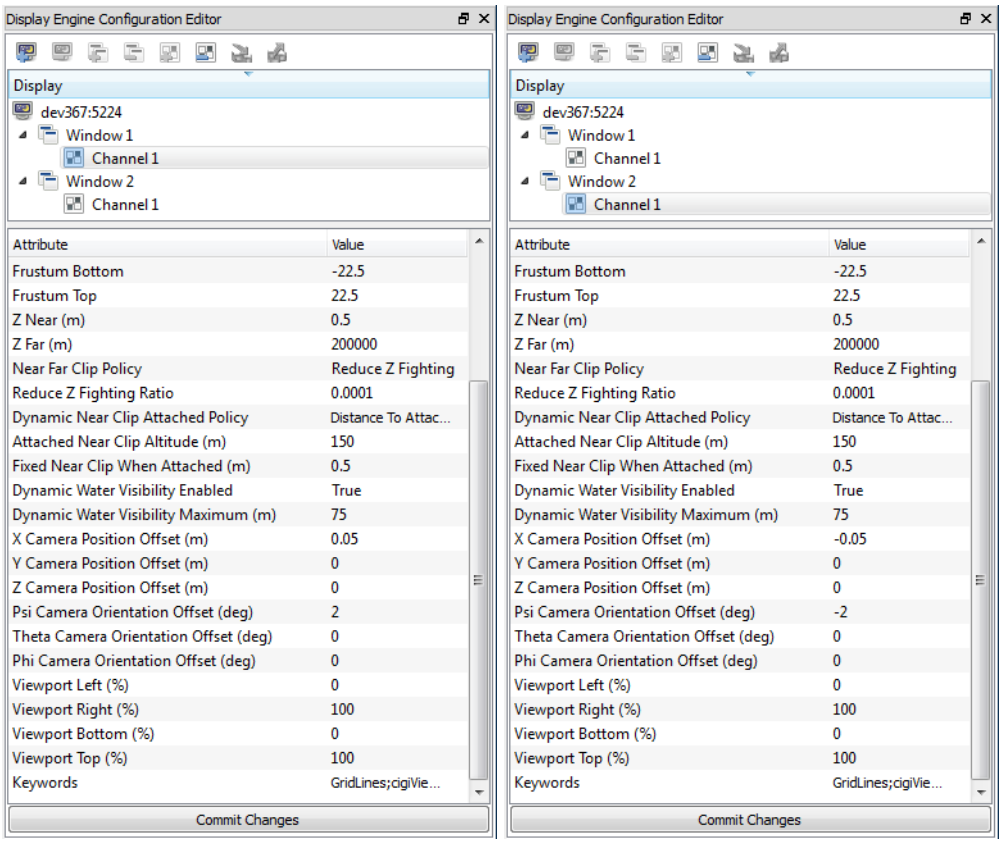
Environment Variable	Description	Default
OSG_STEREO	Turn stereo on.	ON
OSG_STEREO_MODE	Use anaglyphic stereo when in stereo.	ANAGLYPHIC
OSG_SCREEN_DISTANCE	Specifies the distance, in meters, that the viewer is from the screen.	0.50
OSG_SCREEN_HEIGHT	Specifies the height of the image, in meters, on the 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/projects/osg/wiki/Support/UserGuides/StereoSettings.

73.6.2. 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 73-17 shows an example configuration.



Left channel Right channel

Figure 73-17. 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 (http://www.cl.cam.ac.uk/~nad10/pubs/EI5291A-05.pdf).
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.



74. Common GUI Controls and Behaviors

This chapter provides details about using the various panels and toolbars in the front-end.

Dockable Panels and Toolbars 74-2
- Docking and Undocking a Panel 74-2
- Docking and Undocking a Toolbar 74-3
- Displaying and Hiding Panels and Toolbars 74-3
- Restoring the Layout of Toolbars and Panels 74-4

Using Tear-Off Menus 74-5

Viewing the Window in Full Screen Mode 74-6

Using Context-Sensitive Menus 74-6

Common VR-Forces Buttons 74-7

Adding Text to the Title Bar 74-8

Specifying Display Units 74-9

74.1. Dockable Panels and Toolbars

VR-Forces has floating, dockable panels for many of its important functions (Figure 74-1). It also has toolbars that let you quickly access the commands available on the menus.

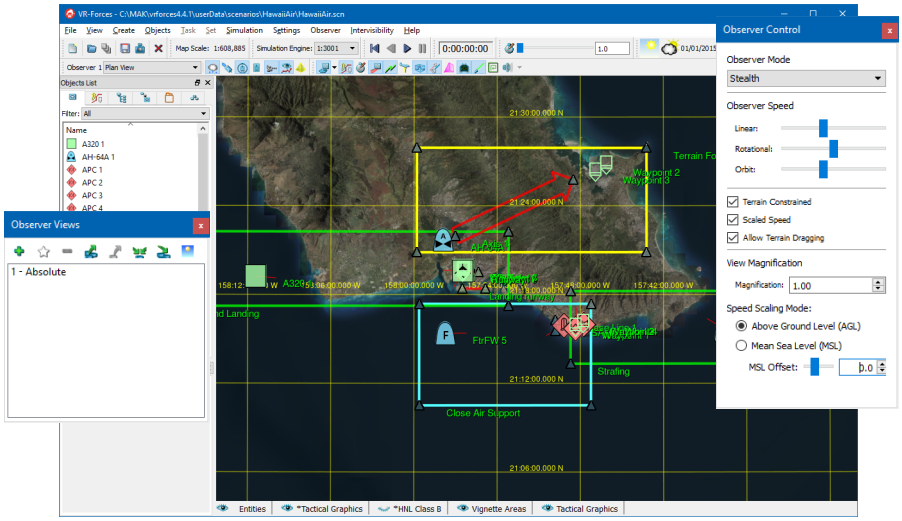


Figure 74-1. VR-Forces window and undocked panels

74.1.1. 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.

74.1.2. Docking and Undocking a Toolbar

To undock a toolbar:

1. Place the cursor over the handle.



Figure 74-2. 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 74-3).

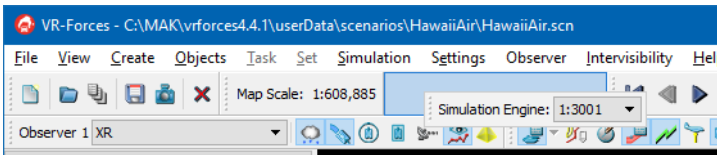


Figure 74-3. Docking a toolbar

74.1.3. 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.

74.1.4. Restoring the Layout of Toolbars and Panels

VR-Forces saves the layout of toolbars and panels when you shut down. (It does not save the size and location of the main window.) You can restore the factory layout or the last saved layout.

To restore the layout of toolbars and panels:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the GUI Layout page (Figure 74-4).

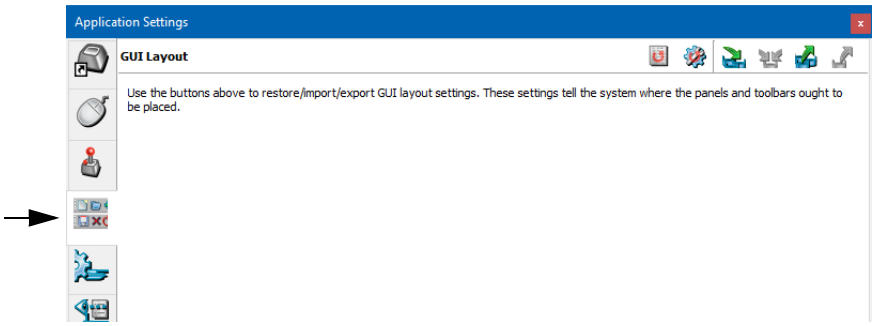


Figure 74-4. GUI Layout page

3. Click the settings button for the operation you want to perform. (For a summary of common settings buttons, please see "Managing VR-Forces Settings," on page 4-23.)

74.2. 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 74-5).
3. Right-click. The menu is displayed as a window that you can leave open and reposition.

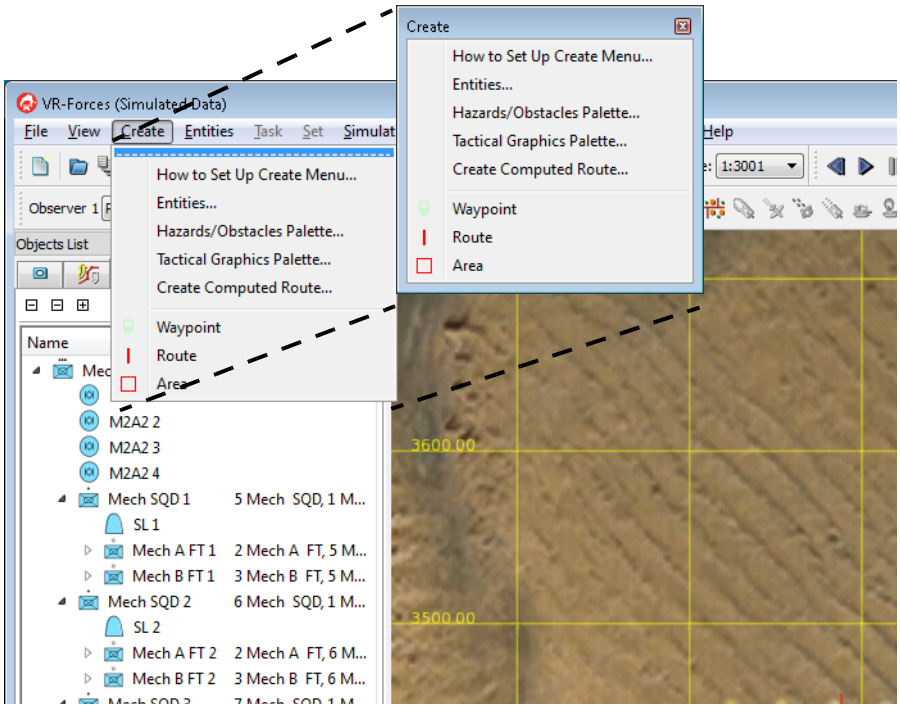


Figure 74-5. Tearing off a menu

74.3. 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`**.

74.4. Using Context-Sensitive Menus

VR-Forces has context sensitive menus (also called 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.

74.5. 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.

74.6. 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 Session Options page (Figure 74-6).

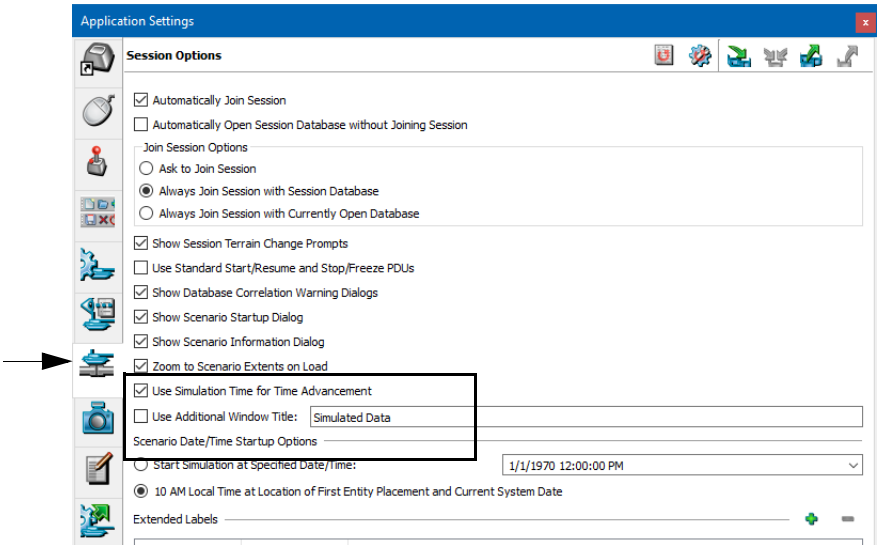


Figure 74-6. Session Options page

3. Select the Use Additional Window title check box.
4. Type the text you want to display in the text box.

74.7. 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 following coordinate system and unit display information is independent of (and therefore not affected by) the settings described in this section:

- The coordinate system information for the observer location in the Observer Information Panel.
- Grid line coordinates, units, and precision.

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 74-1 describes the coordinate systems that you can specify.

Table 74-1: 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.

Table 74-1: Coordinate systems supported

Coordinate system	Description
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.
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 74-7).

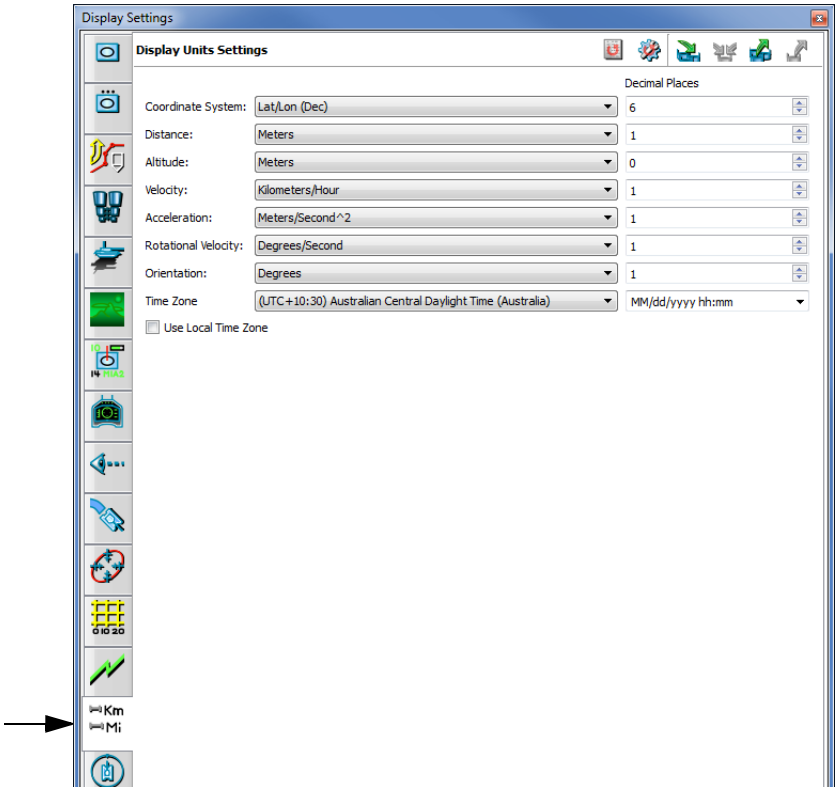


Figure 74-7. Display Units Settings page

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.



75. 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 75-2
Configuring Menus and Menu Commands 75-2
Configuring Dialog Box Pages 75-3
Configuring Context-Sensitive Menus 75-5
Configuring Toolbars 75-6
    Creating Toolbars 75-6
    Specifying an Icon for a Toolbar Item 75-7
Locking Main GUI Elements 75-7

75.1. Introduction

VR-Forces displays menus, menu commands, toolbars, dialog box pages, and popup 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.

75.2. 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 `menuname` and `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.

75.3. 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="DtInterfaceLayoutEditorPage"/>
    <pageitem pageitemname="DtPerformanceOptionsPage"/>
    <pageitem pageitemname="DtSpotReportOptionsPage"/>
    <pageitem pageitemname="DtSessionOptionsPage"/>
    <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="DtEntityStateFrozen"/>
        <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.

75.4. 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:

```
<rightclickmenus name="right_click_menus">
  <rightclickmenu rightclickmenuname="Multiple_Entity">
    <popupmenu menuname="DtVrfRightClickContextMenu">
      <menu menuname="DtRightClickTaskMenu"/>
      <menu menuname="DtRightClickSetMenu"/>
      <separator separatorname="2"/>
      <menuitem itemname="DtVrfEntityAbandonPlanAction"/>
      <menuitem itemname="DtVrfEntityRestartPlanAction"/>
      <menuitem itemname="DtVrfEntityPlanAction"/>
      <menuitem itemname="DtVrfManageReactiveTasksActionId"/>
      <menuitem itemname="DtVrfEntitySkipTaskAction"/>
      <separator separatorname="110"/>
      <menuitem itemname="DtVrfAggregateAsAction"/>
      <menuitem itemname="DtExpandAggregateMenuItem"/>
      <menuitem itemname="DtExpandAllAggregateMenuItem"/>
      <menuitem itemname="DtCollapseAggregateMenuItem"/>
      <menuitem itemname="DtCollapseAllAggregateMenuItem"/>
      <menuitem itemname="DtObjectSettingsAction"/>
      ...
      <menuitem itemname="DtZoomToExtentsAction"/>
      <separator separatorname="DtInformationItemSeparator"/>
      <menuitem itemname="DtVrfVrlinkObjectInformationItem"/>
    </popupmenu>
  </rightclickmenu>
<rightclickmenus name="right_click_menus">
```

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.

75.5. 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>
```

75.5.1. 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="Place_IEDToolbarItem"/>
</toolbar>
```

75.5.2. 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 Scripted Tasks dialog box. For details, please see "Specifying the Task Menu Icon," on page 28-11. 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*.

75.6. 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.



76. Creating and Editing Key Mappings

This chapter explains how to edit key mappings and create new key mappings.

Introduction 76-2
The Key Mapping Editor 76-3
Binary Key Mappings 76-4
Editing a Key Map 76-4
Adding a Key Mapping 76-5
Changing a Key Mapping 76-7
Deleting a Key Mapping 76-7
Using Combined Key Mappings 76-8
Filtering the Function List 76-9
Creating a Key Map 76-9
Deleting a Key Map 76-9

76.1. 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.

76.2. The Key Mapping Editor

The Key Mapping Editor (Figure 76-1) 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.

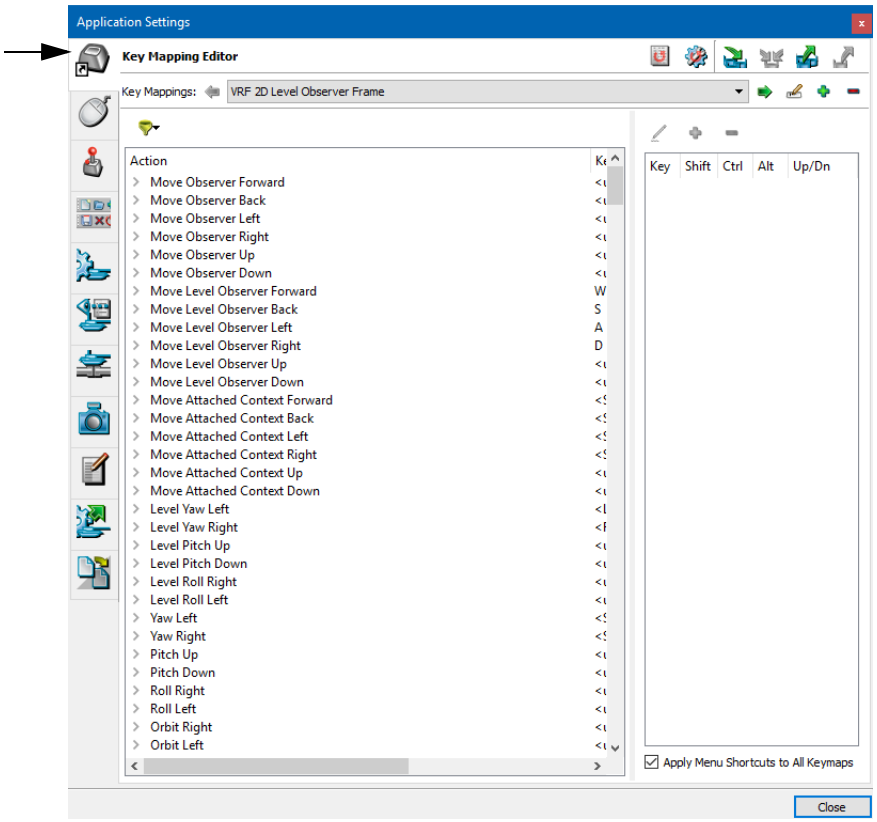


Figure 76-1. Key Mapping Editor

76.2.1. 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 76-1.

76.3. 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 76-1).
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 76-5.
   - "Changing a Key Mapping," on page 76-7.
   - "Deleting a Key Mapping," on page 76-7.
5. Optionally, export the key map to a file.
6. Click OK.

76.3.1. 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 most 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 76-9.
2. Select the function to which you want to add a key mapping.
3. Click the Add button (+) above the Key Map window (Figure 76-2). The Key Entry dialog box opens (Figure 76-3).

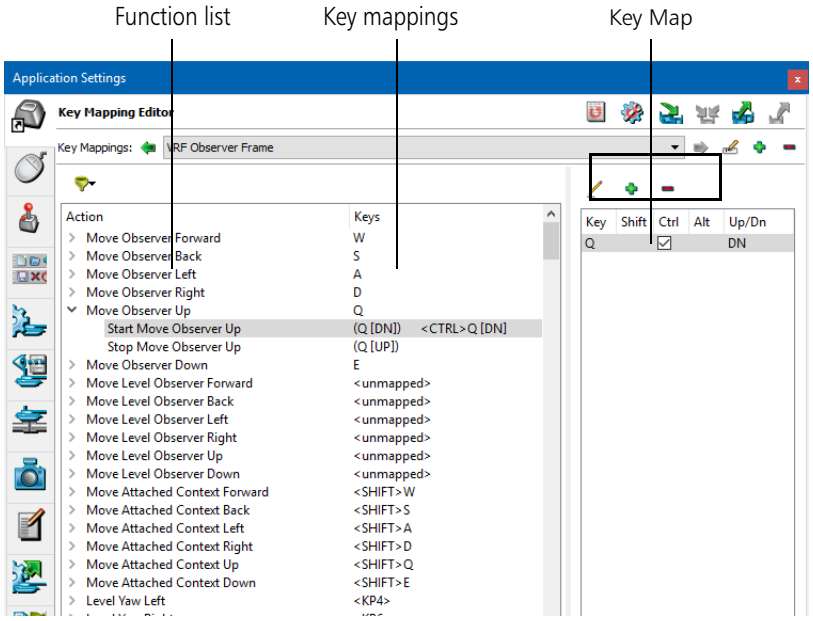


Figure 76-2. Key Mapping Editor

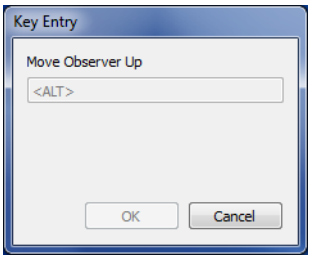


Figure 76-3. Key Entry dialog box

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 76-8.) The mapping is added to the function and is listed in the Keys column of the function list and in the Key Map window.

76.3.2. 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 76-2).
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 76-4). It shows the current key mapping.

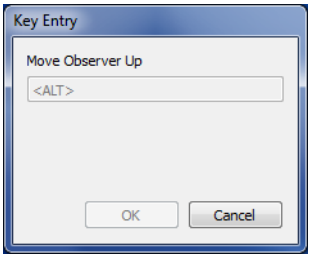


Figure 76-4. Key Entry dialog box

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.

76.3.3. 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 76-2).
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.

76.4. 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 **Space Bar** 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 76-5). It lists the functions that already use this key mapping.

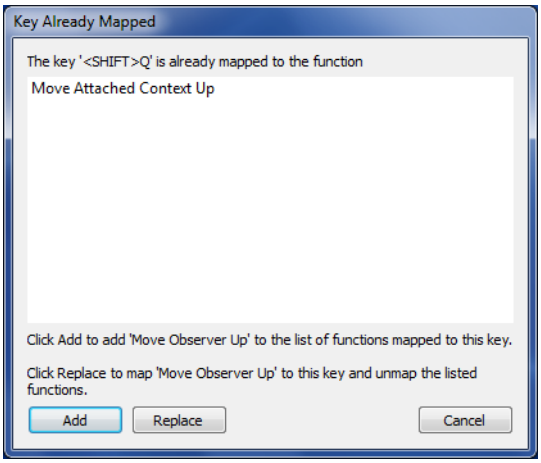


Figure 76-5. Key Already Mapped dialog box

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.

76.5. 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.

76.6. 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 76-1).
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 76-5.

76.7. 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 76-1).
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. Appendixes

The appendixes contain supplementary information.



A. VR-Forces Configuration Files

This chapter introduces the configuration files used by VR-Forces and describes the syntax of VR-Forces MTL files.

Introduction A-2
The vrfSim.mtl Configuration File A-3
Configuring Communication Models A-8
MÄK Technologies Lisp (MTL) Files A-9
Using Environment Variables in an MTL File A-9
Specifying Lists of Lists A-10
Using Conditional Statements A-10

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 MÄK Technologies LISP (MTL) syntax. The major back-end configuration file is *vrfSim.mtl*. It is described in this chapter. Other files are described throughout this manual in the appropriate section. MTL syntax is described in "MÄK Technologies Lisp (MTL) Files," on page A-9.

A.2. The vrfSim.mtl Configuration File

The file *./appData/settings/vrfSim/vrfSim.mtl* configures back-ends. Table A-1 describes the parameters in *vrfSim.mtl*. The Description column includes the command-line argument equivalent for parameters that have them.

Table A-1: *vrfSim.mtl* parameters (Sheet 1 of 6)

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 order of battle 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 A-1: *vrfSim.mtl* parameters (Sheet 2 of 6)

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.
diGuyCharacterDataFile	Specifies the DI-Guy character data file to load if useDIGuy is 1.
diGuyAnimationsFile	Specifies the DI-Guy animations file to load if useDIGuy is 1.
disableParallelPreprocess	Enables (0) or disables (1) use of parallel processing for preprocessing objects. If disabled, VR-Forces uses serial preprocessing. Default: 0.
doNotUseConsole	Disables (1) or enables (0) writing of output to the console. Default: 0.
DtNotifyLevel	Sets the notification level (0-4) for information and warning messages. Default: 2. Command line: `-n`
dynamicFeaturesDisabled	Enables (0) or disables (1) dynamic features. Dynamic features are features that get integrated into the vector network for path planning.
enableDebugDrawer	Enables (1) or disables (0) the simulation terrain debug drawer. Default: 1.
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.
frameRateStatisticsLogFile	Specifies a log file for frame rate statistics.
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.

Table A-1: *vrfSim.mtl* parameters (Sheet 3 of 6)

Parameter	Description
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 MÄK 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.
msdlSIDCMappingFile	The file VR-Forces uses to map between SIDC codes and object type enumerations when it imports MSDL files.
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.
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`
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.
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`

Table A-1: *vrfSim.mtl* parameters (Sheet 4 of 6)

Parameter	Description
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.
statusUpdateHeartbeat	Specifies how often, in seconds, the back-end status heartbeat is sent.
targetFrameRate	Specifies the mean frame rate 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 6-17. Default: 22.
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.
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 5-16
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.

Table A-1: *vrfSim.mtl* parameters (Sheet 5 of 6)

Parameter	Description
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 5-19.
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.
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`

Table A-1: *vrfSim.mtl* parameters (Sheet 6 of 6)

Parameter	Description
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.

A.3. Configuring Communication Models

Communication models are configured in *./data/simulationModelSets/<model-set>/commModelParams.mtl*. Communication models determine how radio messages travel from one simulation object to another. They determine which simulation objects are connected to each other and how long it takes a message to get from one to another.

Communication model classes are derived from *DtCommModel* and are generated from the communication model factory based on the string types specified in the *commModelParams.mtl* file. A communication model entry contains parameters for the type of communication model descriptor and communication model class being used.

For more information about communication models, please see "How Simulation Objects Communicate," on page 15-6 and Chapter 13, *Using a Communications Effects Server*.

A.4. MÄK Technologies Lisp (MTL) Files

Many of the configuration files for MÄK products are ASCII text files that use MÄK 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 MÄK 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.4.1. 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.4.2. 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.4.3. 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. Object Parameters

This appendix describes the high-level parameters in the object parameter database. The OPD Editor has context-sensitive help for most parameters and also describes them in the online help system. It also has reference material for local and remote subcomponents and object type enumerations.

The Structure of a Simulation Object Parameter B-2
The Parameter Type Hierarchy B-3
Vrf Base Object Parameters B-3
Vrf Object Parameters B-7
Moving Object Parameters B-8
Parameter Types for Entity Platforms B-8
Simulation Object Parameters B-8
Object Type Enumerations B-9
Object Super-type B-9
AggregateKind B-9
ObjectKind B-10
Unit Category B-10
Unit Specific B-11
Point Object Category B-12
LinearObjectKind B-12
LinearObjectCategory B-12
Interaction B-12

B.1. The Structure of a Simulation Object Parameter

Many of the files in the object parameter database uses MÄK Technologies LISP (MTL). (For details about MTL, please see "MÄK Technologies Lisp (MTL) Files," on page A-9.) 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.
- 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.

B.2. The Parameter Type Hierarchy

Figure 68-4, in "Parameter Types," on page 68-8, illustrates the hierarchy of parameter types. In this section, we describe the parameters associated with each **parameter-type**.

B.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). Please see the General Parameters topic in the OPD Editor online help for parameter descriptions. In addition to parameters, each object has an object type and parameter type, as described in Table B-1.

Table B-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 68-4.
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 B-2 lists the subcomponent parameters and their possible values.

Table B-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 B-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 B-3 lists the subcomponent parameters and their possible values.

Table B-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 B-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.)

B.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**.

B.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.

B.2.4. Parameter Types for Entity Platforms

Table B-4 lists the parameter types used by each entity type.

Table B-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

B.3. Simulation Object Parameters

For descriptions of simulation object parameters, please see the descriptions of individual components in the OPD Editor online help.

B.4. 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.

B.4.1. 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

B.4.2. 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

B.4.3. 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

B.4.4. 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		

B.4.5. 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

B.4.6. Point Object Category

Subcategories for use with the Kind field = *DtPointObjectKind*.

Other	0
ControlPoint	1
TargetPoint	2
Marker	3

B.4.7. LinearObjectKind

Subcategories for use with the Kind field = *DtLinearObjectKind*.

Other	0
PhaseLine	1
Route	2

B.4.8. LinearObjectCategory

Subcategories for use with the Kind field = *DtArealObjectKind*.

Other	0
ControlArea	1
Obstacle	2

ArealObjectCategory

B.4.9. Interaction

Kinds for Supertype interaction.

Other	0
Fire	1
Detonate	2



C. 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..... C-2

C.1. VR-Forces Systems

Table C-1 lists all of the systems provided with the *EntityLevel.sms* and provides information about them. Table C-2 lists each system and the entity types that use it. Table C-3 lists systems and the scripted tasks configured in them for *EntityLevel.sms*. Table C-4 lists the systems for *AggregateLevel.sms*. Table C-5 lists the system and entity types for *AggregateLevel.sms*. Table C-6 lists systems and the scripted tasks configured in them for *AggregateLevel.sms*.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
120mm Gun	weapon	ground	Turreted 120mm gun, typical of tanks such as the M1A2. Fires M829A1-AP and M830-HEAT rounds. Targets ground vehicles.
125mm Gun	weapon	ground	Turreted 125mm gun, typical of tanks such as the T-80. Fires both BM-9-AP and BM-14m-HEAT rounds. Targets ground vehicles.
ADA Quad 14.5mm Machine Gun	weapon	ground	ADA Quad Machine Gun, multi-barreled machine gun used for ADA. Found on ZPU-4. 14.5mm rounds. Targets Air entities.
25mm Gun	weapon	ground	Turreted 25mm gun. Fires M791-AP 25mm rounds. Targets ground vehicles.
30mm Gun	weapon	ground	Turreted 30mm gun. Fires AT-30mm rounds. Targets ground vehicles.
Active RADAR Sensor	sensor	all	Allows an entity to detect other objects through RADAR. Also publishes emitter beams that represent the emissions from the sensor.
Active SONAR Sensor	sensor	all	Allows an entity to detect other objects through SONAR.
Air Aggregate	movement	aggregate	Basic movement capabilities for any air-based unit, fixed wing or rotary wing.
Air Cushion	movement	ground	Air cushion movement; allows entity to cross both land and water.
Airborne Targeting RADAR	sensor	fixed-wing, rotary-wing	RADAR used by aircraft to detect targets. Publishes an emitter beam.
Anti-Submarine Missile (Vertically Launched)	weapon	surface	Missile launched from deck of ship, flies out to submarine target and drops a homing torpedo.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
Building Default Damage	damage	cultural-feature	Default damage system for a building.
CIS Air Defense Missile Package	weapon	fixed-wing	Missile launcher with loadout for CIS air defense entity. Fires Aphid AA-8 missiles against air targets and Kedge AS-14 missiles against ground targets.
CIS Attack Helicopter Missile Package	weapon	rotary-wing	Missile launcher with loadout for CIS attack helicopter. Fires Archer AA-11 missiles against air targets and Spiral AT-6 missiles against ground targets.
CIS Attack/Strike Missile Package	weapon	fixed-wing	Missile launcher with loadout for CIS attack/strike entity. Fires Aphid AA-8 missiles against air targets and Kerry AS-7 missiles against ground targets.
CIS Counter Measures Launcher	other	all	Allows an entity to launch counter measures for self defense. Configured with CIS flares and chaff.
CIS Fighter Bomber Bomb Bay	weapon	fixed-wing, rotary-wing	Releases bombs in response to Release Bomb on Target, Location and Laser Spot tasks. Configured to launch bombs manufactured in the Commonwealth of Independent States.
CIS Heavy Bomber Bomb Bay	weapon	fixed-wing, rotary-wing	Releases bombs in response to Release Bomb on Target, Location and Laser Spot tasks. Configured to launch bombs manufactured in the Commonwealth of Independent States.
Counter Measures Launcher	other	all	Allows an entity to launch counter measures for self defense.
Cruise Missile	movement	fixed-wing	Cruise missile flight dynamics. Fast and maneuverable, but with limited tasking capabilities.
DDG Embedded Support Craft (LAMPS/RHIB)	other	all	System that allows an entity to deploy and recover Embedded Entities appropriate to a ship with supporting LAMPS units.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
DI Laser Designator	other	all	A laser designator, which can aim a laser at targets. For use with a laser guided missile delivery system. This laser designator can be explicitly tasked to designate particular targets, or set to autonomous lasing mode. In autonomous lasing mode, the laser will automatically designate hostile ground vehicles.
Exocet Cruise Missile Launcher (Forward Launched)	weapon	fixed-wing	Launcher for Exocet anti-ship cruise missiles. Launches missiles forward of the entity, appropriate for aircraft.
Exocet Cruise Missile Launcher (Vertically Launched)	weapon	surface, ground	Launcher for Exocet anti-ship cruise missiles. Launches missiles at an upward angle, appropriate for ships.
Explosive Device	weapon	all	Explosive device system. Allows 3 types of detonations -- instant, time-delayed, and proximity.
Fall From Sky Dynamics	movement	all	Movement dynamics for based on gravity and drag of object.
Maverick Missile Launcher	weapon	fixed-wing	Fixed Maverick Missile Launcher. Fires Air-To-Ground missiles at ground targets. Fixed to the entity (no turret) and always fires forward.
Sidewinder Missile Launcher	weapon	fixed-wing, rotary-wing	Fixed Sidewinder Missile Launcher. Fires Air-To-Air missiles at air targets. Fixed to the entity (no turret) and always fires forward.
Fixed Wing Default Armor	damage	fixed-wing	Basic armor for a fixed-wing aircraft.
Fighter Jet	movement	fixed-wing	Fighter jet, such as the A-10.
Heavy Plane	movement	fixed-wing	Heavy plane flight dynamics, such as a cargo plane.
High Maneuverability Fighter	movement	fixed-wing	Highly maneuverable fighter jet, such as the F-16.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
Frictionless Gravity Dynamics	movement	all	Movement dynamics solely based on gravity.
GAU-8 Avenger	weapon	fixed-wing	GAU-8 30mm Gatling gun, found on the U.S. A-10 ground attack aircraft. Used for attacking armor and other vehicles.
Ground Aggregated Movement	aggre-gated	aggregate	Basic movement capabilities for any ground-based aggregated unit, both vehicle and DI.
Ground Convoy Movement	disaggre-gated	aggregate	Basic movement capabilities for any ground-based convoy unit.
Ground Disaggre-gated Move-ment	disaggre-gated	aggregate	Basic movement capabilities for any ground-based disaggregated unit, both vehicle and DI.
Heavy Armor	damage	ground	Heavy armored ground vehicle, such as a tank.
Light Armor damage model	damage	ground	Component that computes damage for lightly armored ground vehicles. Maps munition types to damage tables.
Tracks	movement	ground	Movement model for ground entities with tracks.
Tracks/Amp hibious	movement	ground	Amphibious tracked vehicles that can move over land and water.
No Armor	damage	ground, vrf object	Unarmored ground vehicle, such as a civilian vehicle.
Wheels (off road)	movement	ground	Wheeled vehicle designed for off road use, such as a HMMWV. Can switch into road following mode.
Wheels (road)	movement	ground	Wheeled vehicle designed for on-road use, such as a civilian car.
GSh-301 Cannon	weapon	fixed-wing	GSh-301 30mm cannon used on Russian attack aircraft. Used for attacking ground and air targets.
AK-47	weapon	human	Handheld AK-47 rifle. Targets human entities.
AT4	weapon	human	Handheld AT4 Anti-Tank weapon. Targets ground vehicles and helicopters.
M16 rifle	weapon	human	Handheld M16 rifle. Targets human enti-ties.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
M240B Machine Gun	weapon	human	Hand carried M240B 7.62mm Machine Gun. Targets lifeforms.
M249 SAW	weapon	human	M249 Squad Automatic Weapon, 5.56mm fully automatic rifle. Targets lifeforms.
M60 Machine Gun	weapon	human	Hand carried M60 7.62mm Machine Gun. Targets lifeforms.
RPG Launcher	weapon	human	Handheld RPG Launcher. Targets ground vehicles and helicopters.
Handheld UAV	movement	fixed-wing	Handheld / hand-launched UAV such as the RQ-11 Raven.
Homing Torpedo Capability (Forward Launched)	weapon	subsurface, surface, rotary-wing, fixed-wing	Homing torpedo capability. Launched from forward tubes.
Human Disaggregated Movement	disaggregated	aggregate	Basic movement capabilities for any human disaggregated unit, both vehicle and DI.
Suicide Vest	weapon	human	Explosive device system that is worn by a lifeform.
Human Default	movement	human	Default movement dynamics for human entities. Manages movement and posture.
IFF Transponder	other	all	Sends IFF data, when turned on. IFF is off by default.
IR Sensor	sensor	all	Allows an entity to detect other objects through Infrared.
Laser Designator	other	all	A laser designator, which can aim a laser at targets. For use with a laser guided missile delivery system. This laser designator can be explicitly tasked to designate particular targets, or set to autonomous lasing mode. In autonomous lasing mode, the laser will automatically designate hostile ground vehicles.
Laser Guided Hellfire Missile Launcher	weapon	fixed-wing, rotary-wing	Launches Hellfire missiles at targets that are lased with the laser code matching the one set in the launcher. Can only fire on targets designated with a laser.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
Laser Guided Missile Dynamics	movement	missile	Movement dynamics for a laser guided missile. A missile using this type will require a laser on its target.
Default Armor	damage	human	Basic damage model for lifeform entities.
Limit Entity Existence	other	all	Used to control the life span of the object this system is attached to. Once it's life span is up the object is removed from the simulation.
M2HB Machine Gun	weapon	ground	Turreted M2 .50 Caliber Machine Gun, typical of the secondary gun on tanks such as the M1A2. M2 12.7mm rounds. Targets lifeforms.
M230 Chain Gun	weapon	ground, rotary-wing	M230 chain gun, mounted on the AH-64 Apache. Fires High Explosive Dual Purpose rounds that can penetrate 2 inches of armor, and that fragment to kill soft targets.
M240 Machine Gun	weapon	ground, rotary-wing	Vehicle mounted M240 7.62mm Machine Gun. Typical secondary gun on tanks such as the M1A2. Targets lifeforms and unarmored vehicles.
M250 smoke grenade launcher	other	all	Allows an entity to launch thermal obscuring white phosphorous (WP) smoke.
M252 81mm mortar	weapon	human	
M270 MLRS Launcher	weapon	ground	Turreted indirect fire weapon. Fires M26 rockets on command. Targets must be at least 2km away.
M284 155mm Cannon	weapon	ground	Turreted indirect fire weapon. Fires 155mm M107 rounds on command. Targets must be at least 2km away.
M60 Machine Gun	weapon	ground, rotary-wing	M60 7.62mm Machine Gun, typical vehicle machine gun. Replaced by M240. Targets lifeforms and unarmored vehicles.
M61 Vulcan	weapon	fixed-wing	M61 20mm Gatling gun, found on the U.S. attack aircraft. Used for attacking ground and air targets.
MAD Sensor	sensor	all	Allows an entity to detect Subsurface entities using Magnetic Anomaly Detection.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
Missile Default Armor	damage	missile, fixed-wing, scripted-movement	Basic armor for a missile.
Guided Missile Dynamics	movement	missile	Movement dynamics for a basic guided missile.
Missile Warhead	weapon	fixed-wing, subsurface, surface	Missile warhead. It explodes as soon as it hits another entity, the ground, or the surface of the water.
MK 45 Naval Gun	weapon	surface	Turreted indirect fire weapon. Fires 127mm rounds on command.
Naval Depth Charge Deployment	weapon	fixed-wing	Depth charge release system for naval depth charges.
Naval Mine Deployment	weapon	fixed-wing, subsurface, surface	Mine release system for naval (under-water) mines.
Naval Mine Dynamics	movement	all	Movement dynamics based on gravity and drag of object. Will also allow the object to stop at a certain altitude.
Naval Mine Explosive Device	weapon	bomb	Naval Mine device system. Allows 3 types of detonations -- instant, time-delayed, and proximity.
Naval Mine Sweep	other	surface	Execute task to sweep and disable naval mines.
Passive RADAR Sensor	sensor	all	Allows an entity to detect other objects through RADAR. No emitters are published for this sensor.
Passive SONAR Sensor	sensor	all	Allows an entity to detect other objects through SONAR.
Patriot Missile Launcher	weapon	ground	Turreted Patriot Missile Launcher. Targets air vehicles.
Periscope	other	subsurface	System that will allow modeling of a periscope.
Rotary Wing Attack	movement	rotary-wing	Attack Helicopter, such as the A-64.
Rotary Wing Default Armor	damage	rotary-wing	Basic armor for a rotary wing aircraft.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
Cargo	movement	rotary-wing	General Cargo helicopter, such as the CH-53 Super Stallion.
Dipping SONAR Sensor	sensor	rotary-wing	Allows an entity to detect other objects through SONAR.
Utility	movement	rotary-wing	General utility helicopter, such as the UH-60 Blackhawk.
VTUAV	movement	rotary-wing	General Vertical Takeoff Unmanned Aerial Vehicle such as the Dragon Warrior.
SA-9 SAM Missile Launcher	weapon	ground	Turreted surface-to-air missile launcher which fires SA-9 missiles. Targets air vehicles.
Scripted Movement	scripted-movement	scripted-movement	System that will allow for scripted movements.
Small boat	movement	surface	Movement dynamics for small boats.
Bomb Dynamics	movement	bomb	Movement dynamics for a guided smart bomb.
SONAR Sensor	sensor	all	Allows an entity to detect other objects through SONAR.
Sonobuoy Deployer	other	all	System that allows an entity to deploy sonobuoys.
Space Shuttle	movement	fixed-wing	Based on the heavy plane flight dynamics.
Spot Report Generator	other	all	Allows the entity to send spot reports through its radio when its sensors detect other entities in the scenario.
Spot Report Receiver	other	all	Allows an entity to receive spot reports from other entities over the radio, and add these contacts to its list of known entities. By default, spot reports expire after 5 minutes.
Stinger Missile Launcher	weapon	ground	Turreted Stinger Missile Launcher. Targets air vehicles between 1km and 8km away, and fires guided stinger missiles at them.
Subsurface Entity Default	damage	subsurface	Basic damage model for subsurface entities.
Subsurface	movement	subsurface	Movement dynamics for a subsurface vessel.

Table C-1: System descriptions, EntityLevel.sms

System	Category	Entity Types	Description
Surface Entity Default	damage	surface	Basic damage model for a surface vessel.
Surface Disaggregated Movement	disaggregated	aggregate	Basic movement capabilities for any surface-entity-based disaggregated unit.
Large Ship	movement	surface	Movement dynamics for a large ship.
Surface Multiple Hit Damage	damage	surface	Hit point based surface damage model.
Small Surface Ships	damage	surface	Damage model for small boats such as inflatables, sailboats, or small patrol boats.
Tactical Radar Jammer	other	all	Multi-mode ECM tactical jamming system.
Torpedo Warhead	weapon	subsurface, surface	Torpedo warhead. It explodes as soon as it hits another entity, the ground, but not the surface of the water. Can also explode proximal to entity.
TOW Missile Launcher	weapon	ground	Turreted TOW Missile Launcher. Targets ground vehicles.
US Fighter Bomber Bomb Bay	weapon	fixed-wing, rotary-wing	Releases bombs in response to Release Bomb on Target, Location and Laser Spot tasks. Configured to launch bombs manufactured in the United States of America.
US Heavy Bomber Bomb Bay	weapon	fixed-wing, rotary-wing	Releases bombs in response to Release Bomb on Target, Location and Laser Spot tasks. Configured to launch bombs manufactured in the United States of America.
Vertical SAM Missile Launcher	weapon	ground, surface	Surface-to-air missile launcher which fires SM-2 missiles. Targets cruise missiles.
Visual Sensor	sensor	all	Allows an entity to detect other objects through visible light.

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
120mm Gun	AMX-30 MBT, AMX-56 Leclerc MBT, Arjun MBT (No 3D), C1 Ariete MBT (No 3D), FV4030/4 Challenger MBT (No 3D), FV4034 Challenger 2 MBT (No 3D), K1A1 MBT (No 3D), Leopard 2 Tank, M1A2 Abrams MBT, Merkava III MBT (No 3D), Merkava IV MBT (No 3D), PT-91 Twardy MBT (No 3D), Type 10 MBT (No 3D), Type 90 Kyu-maru MBT (No 3D)
125mm Gun	MBT 2000 (No 3D), T-69 MBT, T-72 MBT, T-80 MBT, T-84 MBT (No 3D), T90A MBT (No 3D), Type 99 MBT (No 3D)
ADA Quad 14.5mm Machine Gun	ZPU-4 AA Gun, ZSU-23-4 Shilka
25mm Gun	AAVC7A1 Landing Vehicle, FV101 Scorpion CVR, M2A2 Bradley IFV, M3A2 Bradley CFV, MBT 2000 (No 3D), T-69 MBT, T-72 MBT, T-80 MBT, T-84 MBT (No 3D), T90A MBT (No 3D), Type 99 MBT (No 3D), XM7 FIST-Bradley
30mm Gun	BMP-2 AFV, FV 510 Warrior (No 3D), FV107 Scimitar (No 3D)
Active RADAR Sensor	Aircraft Carrier, Arleigh Burke-class Destroyer, Bay-class DLS (No 3D), Sentinel R1 (No 3D), Dabur-class Patrol Boat (No 3D), Durand de la Penne-class Destroyer (No 3D), Echo-class Survey Ship (No 3D), Fridtjof Nansen-class (No 3D), Guided Missile Destroyer, Guided Missile Frigate, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Henry J Kaiser-class Oiler (No 3D), Horizon-class Destroyer (No 3D), Hunt-class Mine hunter (No 3D), Invincible-class Carrier (No 3D), Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), L-class Frigate (No 3D), M-class Frigate (No 3D), Kingston-class Patrol vessel (No 3D), L 14 Albion (No 3D), LPH01 Ocean (No 3D), Harpers Ferry-class LSD, MIM-104 Patriot Launcher, Osprey-class Mine CMS (No 3D), Nimitz-class Carrier, Oksoy-class Mine hunter (No 3D), Queen Elizabeth-class Carrier (No 3D), River-class Patrol Vessel (No 3D), Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Saar 5-class Corvette (No 3D), Sandown-class Mine hunter (No 3D), Skjold-class Patrol (No 3D), Sovremenny-class Destroyer, Super Dvora Mark II (No 3D), Tiger-class Fast Attack Craft (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D)
Active SONAR Sensor	Alta-class Minehunter (No 3D), Fridtjof Nansen-class (No 3D), L-class Frigate (No 3D), M-class Frigate (No 3D), Oksoy-class Minehunter (No 3D), Sonobuoy (Passive)
Air Aggregate	
Air Cushion	LCAC

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Airborne Targeting RADAR	A-10 Thunderbolt, A-4 Skyhawk (No 3D), AV-8B Harrier II, B-2 Spirit, BAE CT-155 Hawk (No 3D), C-130 Hercules, C-17 Globemaster III (No 3D), C-5 Galaxy (No 3D), Cypher VTUAV, Dragon Warrior VTUAV, E-2 Hawkeye, E-3 Sentry, EA-6B Prowler, Eurofighter Typhoon, F-117A Nighthawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, Hermes 450 (No 3D), Maveric UAS (No 3D), MiG-27 Flogger, MiG-29 Fulcrum, Mirage 2000, Mirage F1, MQ-9 Reaper UAV, P-3 Orion, MQ-1 Predator, RQ-11 Raven UAV, RQ-4 Global Hawk UAV (No 3D), RQ-7 Shadow UAV (No 3D), ScanEagle UAV (No 3D), SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker, Tornado ADV
Anti-Submarine Missile (Vertically Launched)	Arleigh Burke-class Destroyer, Type 23 Frigate (No 3D)
Building Default Damage	POL Storage Tank, Tower Radio
CIS Air Defense Missile Package	MiG-29 Fulcrum, SU-27 Flanker
CIS Attack Helicopter Missile Package	KA-50 Hokum, Mi-24 Hind, Mi-28 Havoc
CIS Attack/Strike Missile Package	MiG-27 Flogger, SU-25 Frogfoot, Su-37 Flanker
CIS Counter Measures Launcher	KA-50 Hokum, Mi-24 Hind, Mi-28 Havoc, MiG-27 Flogger, MiG-29 Fulcrum, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker
CIS Fighter Bomber Bomb Bay	MiG-27 Flogger, MiG-29 Fulcrum, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker
CIS Heavy Bomber Bomb Bay	

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Counter Measures Launcher	A-10 Thunderbolt, A-4 Skyhawk (No 3D), AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), AS-365 Dauphin (No 3D), AS-565 Panther (No 3D), AV-8B Harrier II, BAE CT-155 Hawk (No 3D), Bell 412 (No 3D), Bell 206 JetRanger (No 3D), Bell 406 (No 3D), C-130 Hercules, C-17 Globemaster III (No 3D), C-5 Galaxy (No 3D), CH-148 Cyclone (No 3D), CH-46E Sea Knight, CH-47 Chinook (No 3D), CH-53E Super Stallion, E-2 Hawkeye, EA-6B Prowler, EC145 (No 3D), EH101 Merlin (No 3D), Eurofighter Typhoon, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, HH-65 Dolphin (No 3D), KA-50 Hokum, MD 500 (No 3D), MH-47 Chinook (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MH-6 Little Bird (No 3D), Mi-24 Hind, Mi-28 Havoc, Mi-2 Hoplite, MiG-27 Flogger, MiG-29 Fulcrum, Mirage 2000, Mirage F1, OH-58 Kiowa, P-3 Orion, RAH-66 Comanche, SA 330 Puma (No 3D), SH-3 Sea King (No 3D), SH-60 Seahawk, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker, Tornado ADV, UH-1N Twin Huey (No 3D), UH-60 Blackhawk, UH-72 Lakota (No 3D), V-22 Osprey (No 3D), Westland Lynx (No 3D), WS-61 Sea King (No 3D)
Cruise Missile	Exocet Cruise Missile, RUR-5 ASROC (No 3D)
DDG Embedded Support Craft (LAMPS/RHIB)	Arleigh Burke-class Destroyer
DI Laser Designator	DI Lasing (CIS), DI Lasing (US)
Exocet Cruise Missile Launcher (Forward Launched)	MiG-27 Flogger, Mirage 2000, SU-25 Frogfoot, Su-37 Flanker
Exocet Cruise Missile Launcher (Vertically Launched)	Arleigh Burke-class Destroyer, Durand de la Penne-class Destroyer (No 3D), Guided Missile Destroyer, Guided Missile Frigate, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Horizon-class Destroyer (No 3D), Iroquois-class Destroyer (No 3D), KD-III-class Destroyer (No 3D), Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Saar 5-class Corvette (No 3D), Skjold-class Patrol (No 3D), Sovremenny-class Destroyer, Tiger-class Fast Attack Craft (No 3D), Type 42 Destroyer (No 3D), Type 22 Frigate (No 3D), Udaloy-class Destroyer (No 3D)
Explosive Device	Car Bomb, Duffle Bag, Naval Mine, Quickstrike Mk 65, Roadside IED, Sea Lance ASW, Suicide Bomber
Fall From Sky Dynamics	CIS Chaff, CIS Flare, US Chaff, US Flare

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Maverick Missile Launcher	A-10 Thunderbolt, A-4 Skyhawk (No 3D), AV-8B Harrier II, E-2 Hawkeye, EA-6B Prowler, Eurofighter Typhoon, F-117A Nighthawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, Mirage F1, P-3 Orion, MQ-1 Predator, Tornado ADV
Sidewinder Missile Launcher	A-10 Thunderbolt, A-4 Skyhawk (No 3D), AV-8B Harrier II, BAE CT-155 Hawk (No 3D), Eurofighter Typhoon, F-117A Nighthawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, Mirage F1, Tornado ADV
Fixed Wing Default Armor	A-10 Thunderbolt, A-4 Skyhawk (No 3D), Airbus 310, Airbus 320, Airbus 380 (No 3D), AT-802 (No 3D), AV-8B Harrier II, B-2 Spirit, Beechcraft 36 Bonanza (No 3D), Beechcraft C-12 Super King (No 3D), Beechcraft T-6 (No 3D), BN-2 Islander (No 3D), Boeing 707, Boeing 737 British Midland, Boeing 747-400, Boeing 757 (No 3D), Boeing 767 (No 3D), Boeing 777 (No 3D), Boeing 787 (No 3D), Bombardier CL-600 (No 3D), Bombardier DHC-8 (No 3D), Sentinel R1 (No 3D), Bombardier CRJ100 Conrad Air, Bombardier CRJ200 Air France, C-130 Hercules, C-17 Globemaster III (No 3D), C-5 Galaxy (No 3D), Dornier DO 228-200 LGW, E-2 Hawkeye, E-3 Sentry, EA-6B Prowler, Eurofighter Typhoon, F-117A Nighthawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, Gulfstream G550 (No 3D), Hermes 450 (No 3D), IAI 1124 Sea Scan (No 3D), Lockheed L-1011 (No 3D), Maveric UAS (No 3D), MiG-27 Flogger, MiG-29 Fulcrum, Mirage 2000, Mirage F1, MQ-9 Reaper UAV, P-3 Orion, MQ-1 Predator, RQ-11 Raven UAV, RQ-4 Global Hawk UAV (No 3D), RQ-7 Shadow UAV (No 3D), ScanEagle UAV (No 3D), Space Shuttle, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker, Tornado ADV
Fighter Jet	A-10 Thunderbolt, AV-8B Harrier II, B-2 Spirit, F-117A Nighthawk, MiG-27 Flogger, SU-25 Frogfoot, Su-37 Flanker
Heavy Plane	Airbus 310, Airbus 320, Airbus 380 (No 3D), AT-802 (No 3D), Beechcraft 36 Bonanza (No 3D), Beechcraft C-12 Super King (No 3D), Beechcraft T-6 (No 3D), BN-2 Islander (No 3D), Boeing 707, Boeing 737 British Midland, Boeing 747-400, Boeing 757 (No 3D), Boeing 767 (No 3D), Boeing 777 (No 3D), Boeing 787 (No 3D), Bombardier CL-600 (No 3D), Bombardier DHC-8 (No 3D), Sentinel R1 (No 3D), Bombardier CRJ100 Conrad Air, Bombardier CRJ200 Air France, C-130 Hercules, C-17 Globemaster III (No 3D), C-5 Galaxy (No 3D), Dornier DO 228-200 LGW, E-2 Hawkeye, E-3 Sentry, EA-6B Prowler, Gulfstream G550 (No 3D), IAI 1124 Sea Scan (No 3D), Lockheed L-1011 (No 3D), P-3 Orion

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
High Maneuver-ability Fighter	A-4 Skyhawk (No 3D), BAE CT-155 Hawk (No 3D), Canadair CT-114 (No 3D), Eurofighter Typhoon, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, Hermes 450 (No 3D), MiG-29 Fulcrum, Mirage 2000, Mirage F1, MQ-9 Reaper UAV, MQ-1 Predator, RQ-4 Global Hawk UAV (No 3D), SU-27 Flanker, Tornado ADV
Frictionless Gravity Dynamics	M107 155mm, M26 Rocket, M374A2 81mm
GAU-8 Avenger	A-10 Thunderbolt
Ground Aggre-gated Movement	ADA Plt (CIS), ADA Plt (US), Armor Co (CIS), Armor Co (US), Armor Co HQ (CIS), Armor Co HQ (US), Armor Plt (CIS), Armor Plt (US), COLT Team (CIS), COLT Team (US), CSS Plt (CIS), CSS Plt (US), DI Plt (CIS), DI Squad (CIS), DI Squad (US), FA Battery (US), Ground Aggregate, MI Plt (CIS), MI Plt with IFV (US), USMC FT, USMC SQD, US Army LtInf FT, US Army LtInf SQD, US Army Mech FT A (Javelin), US Army Mech FT B, US Army Mech SQD
Ground Convoy Movement	Convoy
Ground Disaggre-gated Movement	ADA Plt (CIS), ADA Plt (US), Armor Co (CIS), Armor Co (US), Armor Co HQ (CIS), Armor Co HQ (US), Armor Plt (CIS), Armor Plt (US), CSS Plt (CIS), CSS Plt (US), DI Squad (US), FA Battery (US), Ground Aggregate, MI Plt (CIS), MI Plt with IFV (US)
Heavy Armor	AMX-30 MBT, AMX-56 Leclerc MBT, Arjun MBT (No 3D), Badger AEV (No 3D), Beaver AVLB (No 3D), Buffalo MRAP III (No 3D), Buffel ARV (No 3D), C1 Ariete MBT (No 3D), FV4030/4 Challenger MBT (No 3D), FV4034 Challenger 2 MBT (No 3D), K1A1 MBT (No 3D), Leopard 2 Tank, M1A2 Abrams MBT, MBT 2000 (No 3D), Merkava III MBT (No 3D), Merkava IV MBT (No 3D), MRAP-ATV (No 3D), MRAP-CAT II Cougar (No 3D), MRAP-CAT I MaxxPro (No 3D), PT-91 Twardy MBT (No 3D), T-69 MBT, T-72 MBT, T-80 MBT, T-84 MBT (No 3D), T90A MBT (No 3D), Taurus ARV (No 3D), Type 10 MBT (No 3D), Type 90 Kyu-maru MBT (No 3D), Type 99 MBT (No 3D), VM 90 LSVW (No 3D)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Light Armor damage model	Aardvark JSFU (No 3D), AAVC7A1 Landing Vehicle, Achzarit APC (No 3D), AS-90 Artillery, Bison APC (No 3D), BMP-2 AFV, BTR-80 APC, CAESAR SP Howitzer, Cougar FSV (No 3D), Coyote APC (No 3D), FV 510 Warrior (No 3D), FV101 Scorpion CVR, FV107 Scimitar (No 3D), FV432 APC (No 3D), Grizzly APC (No 3D), Husky ARV (No 3D), L-118 Howitzer (No 3D), LAV III APC (No 3D), LG1 Howitzer (No 3D), M-71 Howitzer (No 3D), M109 Howitzer, M1126 Stryker ICV, M113 APC, M252 Mortar (No 3D), M2A2 Bradley IFV, M3A2 Bradley CFV, M577A2 Command Post, M777 Howitzer (No 3D), M88 Medium Recovery Vehicle, M993 MLRS, M9 ACE, Mamba APC (No 3D), MT-LB APC, Namer APC (No 3D), Puma CEV (No 3D), Rapier SAM (No 3D), SA-15 Gauntlet SAM, SA-19 Grison, SA-6 Gainful SAM, SA-9 Gaskin SAM System, VAB APC, VBL ATV, XM7 FIST-Bradley, ZPU-4 AA Gun, ZSU-23-4 Shilka
Tracks	Aardvark JSFU (No 3D), Achzarit APC (No 3D), AMX-30 MBT, AMX-56 Leclerc MBT, Arjun MBT (No 3D), AS-90 Artillery, Badger AEV (No 3D), Beaver AVLB (No 3D), Bison APC (No 3D), BMP-2 AFV, BTR-80 APC, Buffel ARV (No 3D), C1 Ariete MBT (No 3D), Cougar FSV (No 3D), Coyote APC (No 3D), FV 510 Warrior (No 3D), FV101 Scorpion CVR, FV107 Scimitar (No 3D), FV4030/4 Challenger MBT (No 3D), FV4034 Challenger 2 MBT (No 3D), FV432 APC (No 3D), Grizzly APC (No 3D), Husky ARV (No 3D), K1A1 MBT (No 3D), LAV III APC (No 3D), Leopard 2 Tank, LG1 Howitzer (No 3D), M-71 Howitzer (No 3D), M109 Howitzer, M1A2 Abrams MBT, M252 Mortar (No 3D), M2A2 Bradley IFV, M3A2 Bradley CFV, M577A2 Command Post, M777 Howitzer (No 3D), M88 Medium Recovery Vehicle, M993 MLRS, M9 ACE, Mamba APC (No 3D), MBT 2000 (No 3D), Merkava III MBT (No 3D), Merkava IV MBT (No 3D), MT-LB APC, Namer APC (No 3D), PT-91 Twardy MBT (No 3D), Puma CEV (No 3D), Rapier SAM (No 3D), SA-15 Gauntlet SAM, SA-19 Grison, SA-6 Gainful SAM, T-69 MBT, T-72 MBT, T-80 MBT, T-84 MBT (No 3D), T90A MBT (No 3D), Taurus ARV (No 3D), Type 10 MBT (No 3D), Type 90 Kyu-maru MBT (No 3D), Type 99 MBT (No 3D), VAB APC, VBL ATV, XM7 FIST-Bradley, ZSU-23-4 Shilka
Tracks/Amphibious	AAVC7A1 Landing Vehicle

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
No Armor	Actros AHSVS (No 3D), Ambulance, BV206 SUSV (No 3D), Car Bomb, Civilian vehicle, Defense Satellite, Duffle Bag, Female On Bike, Fire Engine, Fuel Tanker Trailer, G-Wagen (No 3D), GAZ-69 Utility Vehicle, Goat, HMMWV Utility Vehicle, HMMWV with Avenger, HMMWV with M2, HMMWV with Shelter, HMMWV with TOW launcher, Horse, Horse with Rider, Iveco LMV (No 3D), Jackal MWMIK (No 3D), Jet Ski, Land Rover Wolf (No 3D), LCAC, M-Gator ATV (No 3D), M35 Truck, M58 MICLIC, M939A2 5-Ton Truck, M977 HEMTT Cargo Truck, M978 HEMTT Fuel Truck, Male On Bike, MILCOTTS Silverado (No 3D), MIM-104 Patriot Launcher, Mule, Navistar 7000 (No 3D), Police Car, Rigid-Hulled Inflatable Boat, Rigid-Hulled Inflatable Boat - Civilian, Roadside IED, Sailboat, Sheep, Speedboat, Taxicab, Technical Truck, Tractor Trailer Cab, Tractor Trailer with Cab, Water Buffalo, ZIL-135 8x8 Truck
Wheels (off road)	Actros AHSVS (No 3D), Buffalo MRAP III (No 3D), CAESAR SP Howitzer, Cow, Female On Bike, Fire Engine, GAZ-69 Utility Vehicle, Goat, HMMWV Utility Vehicle, HMMWV with Avenger, HMMWV with M2, HMMWV with Shelter, HMMWV with TOW launcher, Horse, Horse with Rider, Iveco LMV (No 3D), Jackal MWMIK (No 3D), Land Rover Wolf (No 3D), M1126 Stryker ICV, M113 APC, M35 Truck, M58 MICLIC, M939A2 5-Ton Truck, M977 HEMTT Cargo Truck, M978 HEMTT Fuel Truck, Male On Bike, MIM-104 Patriot Launcher, MRAP-ATV (No 3D), MRAP-CAT II Cougar (No 3D), MRAP-CAT I MaxxPro (No 3D), Mule, Navistar 7000 (No 3D), SA-9 Gaskin SAM System, Sheep, Technical Truck, VM 90 LSVW (No 3D), Water Buffalo, ZIL-135 8x8 Truck
Wheels (road)	Ambulance, BV206 SUSV (No 3D), Car Bomb, Civilian vehicle, Fuel Tanker Trailer, G-Wagen (No 3D), M-Gator ATV (No 3D), MILCOTTS Silverado (No 3D), Police Car, Taxicab, Tractor Trailer Cab, Tractor Trailer with Cab
GSh-301 Cannon	Mirage 2000, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker
AK-47	Afghan Police Officer, Afghan Soldier, DI AK-47, DI Lasing (CIS), Insurgent, Iranian Soldier, Libyan Police Officer, Libyan Soldier, Mideast Combatant, Pakistani Police Officer, Pakistani Soldier, Somali Police Officer, Somali Soldier, Sudanese Police Officer, Sudanese Soldier, Syrian Police Officer, Syrian Soldier, Taliban, Ugandan Police Officer, Ugandan Soldier
AT4	Chilean AT4, US Army AT4
M16 rifle	Brazilian Soldier, Chilean IMI Galil, Chilean M16-M203, Chilean M16, Chilean M4, Colombian Police Officer, Colombian Soldier, DI Lasing (US), DI SMAW And Rifle, Iraqi Police Officer, Iraqi Soldier, Police Officer, Sergeant M16, Suspect, UK Soldier, USMC M16-M203, USMC M16, USMC M4, US Army M16-M203, US Army M16, US Army M4, US Army M9

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
M240B Machine Gun	Chilean M240, USMC M240, US Army M240
M249 SAW	Chilean M249, USMC M249, US Army M249
M60 Machine Gun	Chilean M60, US Army M60
RPG Launcher	DI RPG, , Mideast Combatant with RPG, US Army Javelin
Handheld UAV	Maveric UAS (No 3D), RQ-11 Raven UAV, RQ-7 Shadow UAV (No 3D), ScanEagle UAV (No 3D)
Homing Torpedo Capability (Forward Launched)	Arleigh Burke-class Destroyer, Astute-class Submarine (No 3D), Dolphin-class Submarine (No 3D), Durand de la Penne-class Destroyer (No 3D), Fridtjof Nansen-class (No 3D), Guided Missile Destroyer, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Iroquois-class Destroyer (No 3D), M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Kingston-class Patrol vessel (No 3D), Los Angeles-Class Submarine, Ohio-class Submarine, Romeo-class Submarine, RUR-5 ASROC (No 3D), Scorpene-class Submarine (No 3D), SH-60 Seahawk, Song-class Submarine, Sovremenny-class Destroyer, Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D), Ula-class Submarine (No 3D), Vanguard-class Submarine (No 3D), Victoria-class (No 3D), Walrus-class submarine (No 3D)
Human Disaggregated Movement	COLT Team (CIS), COLT Team (US), DI Plt (CIS), DI Squad (CIS), USMC FT, USMC SQD, US Army LtInf FT, US Army LtInf SQD, US Army Mech FT A (Javelin), US Army Mech FT B, US Army Mech SQD
Suicide Vest	Suicide Bomber

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Human Default	Afghan Police Officer, Afghan Soldier, Brazilian Soldier, Cameraman, Chicken, Child, Child (Crowd Behavior), Chilean AT4, Chilean IMI Galil, Chilean M16-M203, Chilean M16, Chilean M240, Chilean M249, Chilean M4, Chilean M60, Civilian (Chem/Bio Behavior), Civilian Female, Civilian Male, Colombian Police Officer, Colombian Soldier, DI AK-47, DI Lasing (CIS), DI Lasing (US), DI RPG, DI SMAW And Rifle, Factory Worker, Female Civilian (Crowd Behavior), Female Dancer, Fireman (Extinguisher), Fireman (Hose), Flight Deck Crew, Flight Deck Crewman, Flight Deck Crew (Chock And Chains), Flight Deck Crew (Grounding Cable), Flight Deck Crew (Pallet Handler), Flight Deck Worker, Insurgent, Iranian Soldier, Iraqi Police Officer, Iraqi Soldier, Libyan Police Officer, Libyan Soldier, Male Civilian (Crowd Behavior), Man in Wheelchair, Mideast Combatant, Mideast Combatant with RPG, Mideast Female, Mideast Female (Crowd Behavior), Mideast Male, Mideast Male (Crowd Behavior), Mother, Pakistani Police Officer, Pakistani Soldier, Police Officer, Sergeant M16, Somali Police Officer, Somali Soldier, Sudanese Police Officer, Sudanese Soldier, Suicide Bomber, Suspect, Syrian Police Officer, Syrian Soldier, Taliban, Ugandan Police Officer, Ugandan Soldier, UK Soldier, US Army Medic, US EOD, USMC M16-M203, USMC M16, USMC M240, USMC M249, USMC M4, US Army AT4, US Army Javelin, US Army M16-M203, US Army M16, US Army M240, US Army M249, US Army M4, US Army M60, US Army M9, Western Teen

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
IFF Transponder	A-10 Thunderbolt, A-4 Skyhawk (No 3D), AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), Airbus 310, Airbus 320, Airbus 380 (No 3D), AS-365 Dauphin (No 3D), AS-565 Panther (No 3D), Astute-class Submarine (No 3D), AT-802 (No 3D), AV-8B Harrier II, B-2 Spirit, BAE CT-155 Hawk (No 3D), Bay-class DLS (No 3D), Beechcraft 36 Bonanza (No 3D), Beechcraft C-12 Super King (No 3D), Beechcraft T-6 (No 3D), Bell 412 (No 3D), Bell 206 JetRanger (No 3D), Bell 406 (No 3D), BN-2 Islander (No 3D), Boeing 707, Boeing 737 British Midland, Boeing 747-400, Boeing 757 (No 3D), Boeing 767 (No 3D), Boeing 777 (No 3D), Boeing 787 (No 3D), Bombardier CL-600 (No 3D), Bombardier DHC-8 (No 3D), Sentinel R1 (No 3D), Bombardier CRJ100 Conrad Air, Bombardier CRJ200 Air France, C-130 Hercules, C-17 Globemaster III (No 3D), C-5 Galaxy (No 3D), Canadair CT-114 (No 3D), CH-148 Cyclone (No 3D), CH-149 Cormorant, CH-46E Sea Knight, CH-47 Chinook (No 3D), CH-53E Super Stallion, Cypher VTUAV, Dabur-class Patrol Boat (No 3D), Dornier DO 228-200 LGW, Dragon Warrior VTUAV, Durand de la Penne-class Destroyer (No 3D), E-2 Hawkeye, E-3 Sentry, EA-6B Prowler, EC145 (No 3D), Echo-class Survey Ship (No 3D), EH101 Merlin (No 3D), Eurofighter Typhoon, F-117A Nighthawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, Gulfstream G550 (No 3D), Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), HH-65 Dolphin (No 3D), Horizon-class Destroyer (No 3D), Hunt-class Minehunter (No 3D), IAI 1124 Sea Scan (No 3D), Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), KA-50 Hokum, KD-III-class Destroyer (No 3D), L 14 Albion (No 3D), Lockheed L-1011 (No 3D), LPH01 Ocean (No 3D), MD 500 (No 3D), MH-47 Chinook (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MH-6 Little Bird (No 3D), Mi-24 Hind, Mi-28 Havoc, Mi-2 Hoplite, MiG-27 Flogger, MiG-29 Fulcrum, Mirage 2000, Mirage F1, Nimitz-class Carrier, OH-58 Kiowa, P-3 Orion, RAH-66 Comanche, River-class Patrol Vessel (No 3D), SA 330 Puma (No 3D), Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Saar 5-class Corvette (No 3D), SH-3 Sea King (No 3D), SH-60 Seahawk, Sovremenny-class Destroyer, Space Shuttle, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker, Tornado ADV, Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D), UH-1N Twin Huey (No 3D), UH-60 Blackhawk, UH-72 Lakota (No 3D), V-22 Osprey (No 3D), Westland Lynx (No 3D)
IFF Transponder (continued)	WS-61 Sea King (No 3D)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
IR Sensor	AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), Arleigh Burke-class Destroyer, AS-365 Dauphin (No 3D), AS-565 Panther (No 3D), Bell 412 (No 3D), Bell 206 JetRanger (No 3D), Bell 406 (No 3D), CH-148 Cyclone (No 3D), CH-149 Cormorant, CH-46E Sea Knight, CH-47 Chinook (No 3D), CH-53E Super Stallion, Dabur-class Patrol Boat (No 3D), EC145 (No 3D), Echo-class Survey Ship (No 3D), EH101 Merlin (No 3D), Hermes 450 (No 3D), HH-65 Dolphin (No 3D), HMMWV with Avenger, Island-class Patrol Vessel (No 3D), KA-50 Hokum, Maveric UAS (No 3D), MD 500 (No 3D), MH-47 Chinook (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MH-6 Little Bird (No 3D), Mi-24 Hind, Mi-28 Havoc, Mi-2 Hoplite, MQ-9 Reaper UAV, OH-58 Kiowa, MQ-1 Predator, RAH-66 Comanche, River-class Patrol Vessel (No 3D), RQ-11 Raven UAV, RQ-4 Global Hawk UAV (No 3D), RQ-7 Shadow UAV (No 3D), SA 330 Puma (No 3D), Sandown-class Minehunter (No 3D), ScanEagle UAV (No 3D), SH-3 Sea King (No 3D), SH-60 Seahawk, Type 42 Destroyer (No 3D), UH-1N Twin Huey (No 3D), UH-60 Blackhawk, UH-72 Lakota (No 3D), V-22 Osprey (No 3D), Westland Lynx (No 3D), WS-61 Sea King (No 3D)
Laser Designator	DI Lasing (CIS), DI Lasing (US)
Laser Guided Hellfire Missile Launcher	AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MQ-9 Reaper UAV, RAH-66 Comanche
Laser Guided Missile Dynamics	Laser Guided Hellfire Missile

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Default Armor	Afghan Police Officer, Afghan Soldier, Brazilian Soldier, Cameraman, Chicken, Child, Child (Crowd Behavior), Child (Injured), Chilean AT4, Chilean IMI Galil, Chilean M16-M203, Chilean M16, Chilean M240, Chilean M249, Chilean M4, Chilean M60, Civilian (Chem/Bio Behavior), Civilian Female, Civilian Male, Colombian Police Officer, Colombian Soldier, Cow, DI AK-47, DI Lasing (CIS), DI Lasing (US), DI RPG, DI SMAW And Rifle, Factory Worker, Female Civilian (Crowd Behavior), Female Dancer, Fireman (Extinguisher), Fireman (Hose), Flight Deck Crew, Flight Deck Crewman, Flight Deck Crew (Chock And Chains), Flight Deck Crew (Grounding Cable), Flight Deck Crew (Pallet Handler), Flight Deck Worker, Insurgent, Iranian Soldier, Iraqi Police Officer, Iraqi Soldier, Libyan Police Officer, Libyan Soldier, Male Civilian (Crowd Behavior), Man in Wheelchair, Mideast Combatant, Mideast Combatant with RPG, Mideast Female, Mideast Female (Crowd Behavior), Mideast Male, Mideast Male (Crowd Behavior), Mother, Pakistani Police Officer, Pakistani Soldier, Police Officer, Sergeant M16, Somali Police Officer, Somali Soldier, Sudanese Police Officer, Sudanese Soldier, Suicide Bomber, Suspect, Syrian Police Officer, Syrian Soldier, Taliban, Ugandan Police Officer, Ugandan Soldier, UK Soldier, US Army Medic, US EOD, USMC M16-M203, USMC M16, USMC M240, USMC M249, USMC M4, US Army AT4, US Army Javelin, US Army M16-M203, US Army M16, US Army M240, US Army M249, US Army M4, US Army M60, US Army M9, Western Teen
Limit Entity Existence	CIS Chaff, CIS Flare, Mk-46 MOD 5 Torpedo (No 3D), Mk-48 ADCAP Torpedo, RUR-5 ASROC (No 3D), US Chaff, US Flare
M2HB Machine Gun	Achzarit APC (No 3D), Badger AEV (No 3D), BTR-80 APC, Buffel ARV (No 3D), HMMWV with M2, Leopard 2 Tank, M1126 Stryker ICV, M113 APC, MT-LB APC, Namer APC (No 3D), Rigid-Hulled Inflatable Boat, Taurus ARV (No 3D), Technical Truck, Type 10 MBT (No 3D), Type 90 Kyu-maru MBT (No 3D), VAB APC
M230 Chain Gun	AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), MH-60L Black Hawk DAP (No 3D), RAH-66 Comanche, Westland Lynx (No 3D)
M240 Machine Gun	Arjun MBT (No 3D), Bison APC (No 3D), C1 Ariete MBT (No 3D), Cougar FSV (No 3D), Coyote APC (No 3D), FV4034 Challenger 2 MBT (No 3D), FV432 APC (No 3D), Grizzly APC (No 3D), HH-65 Dolphin (No 3D), Husky ARV (No 3D), Iveco LMV (No 3D), Jackal MWMIK (No 3D), LAV III APC (No 3D), M1A2 Abrams MBT, Mamba APC (No 3D), MBT 2000 (No 3D), MH-47 Chinook (No 3D), MRAP-ATV (No 3D), MRAP-CAT II Cougar (No 3D), MRAP-CAT I MaxxPro (No 3D), PT-91 Twardy MBT (No 3D), T-84 MBT (No 3D), T90A MBT (No 3D), Type 99 MBT (No 3D), USMC M240, V-22 Osprey (No 3D), VM 90 LSVW (No 3D)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
M250 smoke grenade launcher	Achzarit APC (No 3D), Arjun MBT (No 3D), Badger AEV (No 3D), Buffel ARV (No 3D), M1A2 Abrams MBT, Merkava III MBT (No 3D), Merkava IV MBT (No 3D), Namer APC (No 3D), Puma CEV (No 3D), Taurus ARV (No 3D), Type 10 MBT (No 3D)
M252 81mm mortar	M252 Mortar (No 3D)
M270 MLRS Launcher	M993 MLRS
M284 155mm Cannon	AS-90 Artillery, CAESAR SP Howitzer, HMMWV Utility Vehicle, L-118 Howitzer (No 3D), LG1 Howitzer (No 3D), M-71 Howitzer (No 3D), M109 Howitzer, M777 Howitzer (No 3D)
M60 Machine Gun	CH-148 Cyclone (No 3D), CH-47 Chinook (No 3D), MH-60 Black Hawk, SH-3 Sea King (No 3D)
M61 Vulcan	F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F/A-18 Hornet
MAD Sensor	CH-46E Sea Knight, P-3 Orion, SH-60 Seahawk
Missile Default Armor	AA-11 Archer Missile, AA-8 Aphid Missile, AGM-65 Maverick Missile, AIM-9 Sidewinder Missile, AS-12 Kegler Missile, AS-14 Kedge Missile, AS-7 Kerry Missile, AT-6 Spiral Missile, BGM-71 TOW Missile, Exocet Cruise Missile, FIM-92 Stinger Missile, Hellfire Missile, Laser Guided Hellfire Missile, M39 Missile, MIM004 Patriot Missile, RUR-5 ASROC (No 3D), SA-9 Missile, Scud-B Missile, SM-2 Standard Missile, SS-24 Scalpel Missile, SS-24 Scalpel Stage
Guided Missile Dynamics	AA-11 Archer Missile, AA-8 Aphid Missile, AGM-65 Maverick Missile, AIM-9 Sidewinder Missile, AS-12 Kegler Missile, AS-14 Kedge Missile, AS-7 Kerry Missile, AT-6 Spiral Missile, BGM-71 TOW Missile, Exocet Cruise Missile, FIM-92 Stinger Missile, Hellfire Missile, Laser Guided Hellfire Missile, MIM004 Patriot Missile, RUR-5 ASROC (No 3D), SA-9 Missile, SM-2 Standard Missile
Missile Warhead	Exocet Cruise Missile, M107 155mm, M26 Rocket, M374A2 81mm, SM-2 Standard Missile

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
MK 45 Naval Gun	Arleigh Burke-class Destroyer, Bay-class DLS (No 3D), Dabur-class Patrol Boat (No 3D), Durand de la Penne-class Destroyer (No 3D), Echo-class Survey Ship (No 3D), Guided Missile Destroyer, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Hunt-class Minehunter (No 3D), Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), L-class Frigate (No 3D), M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Kingston-class Patrol vessel (No 3D), L 14 Albion (No 3D), LPH01 Ocean (No 3D), Oksoy-class Minehunter (No 3D), River-class Patrol Vessel (No 3D), Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Skjold-class Patrol (No 3D), Sovremenny-class Destroyer, Super Dvora Mark II (No 3D), Tiger-class Fast Attack Craft (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D)
Naval Depth Charge Deployment	B-2 Spirit, P-3 Orion
Naval Mine Deployment	B-2 Spirit, Durand de la Penne-class Destroyer (No 3D), Halifax-class Frigate (No 3D), Horizon-class Destroyer (No 3D), Hunt-class Minehunter (No 3D), Osprey-class Mine CMS (No 3D), P-3 Orion, Sandown-class Minehunter (No 3D), Skjold-class Patrol (No 3D), Sovremenny-class Destroyer, Tiger-class Fast Attack Craft (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D)
Naval Mine Dynamics	Naval Mine, Quickstrike Mk 65, Sea Lance ASW
Naval Mine Explosive Device	Naval Mine, Quickstrike Mk 65, Sea Lance ASW
Naval Mine Sweep	Alta-class Minehunter (No 3D), Hunt-class Minehunter (No 3D), Kingston-class Patrol vessel (No 3D), Osprey-class Mine CMS (No 3D), Oksoy-class Minehunter (No 3D), Sandown-class Minehunter (No 3D), Tiger-class Fast Attack Craft (No 3D)
Passive RADAR Sensor	Arleigh Burke-class Destroyer, Dabur-class Patrol Boat (No 3D), Echo-class Survey Ship (No 3D), Fridtjof Nansen-class (No 3D), Hunt-class Minehunter (No 3D), Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Rapier SAM (No 3D), River-class Patrol Vessel (No 3D), SA-15 Gauntlet SAM, SA-19 Grison, SA-6 Gainful SAM, SA-9 Gaskin SAM System, Sandown-class Minehunter (No 3D), Scorpene-class Submarine (No 3D), Super Tanker (No 3D), Type 209 Submarine (No 3D), Type 23 Frigate (No 3D), Vanguard-class Submarine (No 3D), Victoria-class (No 3D)
Passive SONAR Sensor	Fridtjof Nansen-class (No 3D), L-class Frigate (No 3D), M-class Frigate (No 3D), Oksoy-class Minehunter (No 3D), Sonobuoy (Passive)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Patriot Missile Launcher	MIM-104 Patriot Launcher
Periscope	Astute-class Submarine (No 3D), Dolphin-class Submarine (No 3D), Los Angeles-Class Submarine, Ohio-class Submarine, Romeo-class Submarine, Scorpene-class Submarine (No 3D), Song-class Submarine, Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Ula-class Submarine (No 3D), Vanguard-class Submarine (No 3D), Victoria-class (No 3D), Walrus-class submarine (No 3D)
Rotary Wing Attack	AH-1W SuperCobra, AH-64A Apache, AS-365 Dauphin (No 3D), AS-565 Panther (No 3D), HH-65 Dolphin (No 3D), KA-50 Hokum, Mi-24 Hind, Mi-28 Havoc, RAH-66 Comanche
Rotary Wing Default Armor	AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), AS-365 Dauphin (No 3D), AS-565 Panther (No 3D), Bell 412 (No 3D), Bell 206 JetRanger (No 3D), Bell 406 (No 3D), CH-148 Cyclone (No 3D), CH-149 Cormorant, CH-46E Sea Knight, CH-47 Chinook (No 3D), CH-53E Super Stallion, Cypher VTUAV, Dragon Warrior VTUAV, EC145 (No 3D), EH101 Merlin (No 3D), HH-65 Dolphin (No 3D), KA-50 Hokum, MD 500 (No 3D), MH-47 Chinook (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MH-6 Little Bird (No 3D), Mi-24 Hind, Mi-28 Havoc, Mi-2 Hoplite, OH-58 Kiowa, RAH-66 Comanche, SA 330 Puma (No 3D), SH-3 Sea King (No 3D), SH-60 Seahawk, UH-1N Twin Huey (No 3D), UH-60 Blackhawk, UH-72 Lakota (No 3D), V-22 Osprey (No 3D), Westland Lynx (No 3D), WS-61 Sea King (No 3D)
Cargo	CH-148 Cyclone (No 3D), CH-47 Chinook (No 3D), CH-53E Super Stallion, EH101 Merlin (No 3D), MH-47 Chinook (No 3D), SA 330 Puma (No 3D), SH-3 Sea King (No 3D), V-22 Osprey (No 3D), Westland Lynx (No 3D), WS-61 Sea King (No 3D)
Dipping SONAR Sensor	CH-46E Sea Knight, SH-3 Sea King (No 3D), SH-60 Seahawk
Utility	AH-6 Little Bird (No 3D), Bell 412 (No 3D), Bell 206 JetRanger (No 3D), Bell 406 (No 3D), CH-149 Cormorant, CH-46E Sea Knight, EC145 (No 3D), MD 500 (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MH-6 Little Bird (No 3D), Mi-2 Hoplite, OH-58 Kiowa, SH-60 Seahawk, UH-1N Twin Huey (No 3D), UH-60 Blackhawk, UH-72 Lakota (No 3D)
VTUAV	Cypher VTUAV, Dragon Warrior VTUAV
SA-9 SAM Missile Launcher	Rapier SAM (No 3D), SA-15 Gauntlet SAM, SA-19 Grison, SA-6 Gainful SAM, SA-9 Gaskin SAM System
Scripted Movement	M39 Missile, Scud-B Missile, SS-24 Scalpel Missile, SS-24 Scalpel Stage

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Small boat	Dabur-class Patrol Boat (No 3D), Echo-class Survey Ship (No 3D), Hauk-class Fast Attack (No 3D), Island-class Patrol Vessel (No 3D), Jet Ski, Rigid-Hulled Inflatable Boat, Rigid-Hulled Inflatable Boat - Civilian, River-class Patrol Vessel (No 3D), Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Saar 5-class Corvette (No 3D), Sailboat, Skjold-class Patrol (No 3D), Speedboat, Super Dvora Mark II (No 3D), Tiger-class Fast Attack Craft (No 3D)
Bomb Dynamics	CBU-105 SFW, GBU-31A JDAM, KAB-500N Bomb
SONAR Sensor	Aircraft Carrier, Alta-class Minehunter (No 3D), AN/BLQ-11 UUV (No 3D), Arleigh Burke-class Destroyer, Astute-class Submarine (No 3D), Bay-class DLS (No 3D), CH-46E Sea Knight, Dolphin-class Submarine (No 3D), Durand de la Penne-class Destroyer (No 3D), Fridtjof Nansen-class (No 3D), , Guided Missile Destroyer, Guided Missile Frigate, Halifax-class Frigate (No 3D), Horizon-class Destroyer (No 3D), Hunt-class Minehunter (No 3D), Iroquois-class Destroyer (No 3D), L-class Frigate (No 3D), M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Kingston-class Patrol vessel (No 3D), L 14 Albion (No 3D), Los Angeles-Class Submarine, LPH01 Ocean (No 3D), Harpers Ferry-class LSD, Osprey-class Mine CMS (No 3D), Mk-46 MOD 5 Torpedo (No 3D), Mk-48 ADCAP Torpedo, Nimitz-class Carrier, Ohio-class Submarine, Oksoy-class Minehunter (No 3D), REMUS UUV (No 3D), Romeo-class Submarine, Saar 5-class Corvette (No 3D), Sandown-class Minehunter (No 3D), Scorpene-class Submarine (No 3D), SH-3 Sea King (No 3D), SH-60 Seahawk, Skjold-class Patrol (No 3D), Song-class Submarine, Sonobuoy (Passive), Sovremenny-class Destroyer, Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D), Ula-class Submarine (No 3D), Victoria-class (No 3D)
Sonobuoy Deployer	Bay-class DLS (No 3D), Dabur-class Patrol Boat (No 3D), Echo-class Survey Ship (No 3D), Halifax-class Frigate (No 3D), Horizon-class Destroyer (No 3D), Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), L 14 Albion (No 3D), LPH01 Ocean (No 3D), P-3 Orion, River-class Patrol Vessel (No 3D), Tiger-class Fast Attack Craft (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D)
Space Shuttle	Space Shuttle

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Spot Report Generator	A-10 Thunderbolt, A-4 Skyhawk (No 3D), Aardvark JSFU (No 3D), AAVC7A1 Landing Vehicle, Achzarit APC (No 3D), Actros AHSVS (No 3D), Afghan Police Officer, Afghan Soldier, AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), Aircraft Carrier, AMX-30 MBT, AMX-56 Leclerc MBT, AN/BLQ-11 UUV (No 3D), Arjun MBT (No 3D), Arleigh Burke-class Destroyer, AS-365 Dauphin (No 3D), AS-565 Panther (No 3D), AS-90 Artillery, AV-8B Harrier II, B-2 Spirit, Badger AEV (No 3D), BAE CT-155 Hawk (No 3D), Bay-class DLS (No 3D), Beaver AVLB (No 3D), Bell 412 (No 3D), Bell 206 JetRanger (No 3D), Bell 406 (No 3D), Bison APC (No 3D), BMP-2 AFV, BN-2 Islander (No 3D), Brazilian Soldier, BTR-80 APC, Buffalo MRAP III (No 3D), Buffel ARV (No 3D), C-130 Hercules, C-17 Globemaster III (No 3D), C-5 Galaxy (No 3D), C1 Ariete MBT (No 3D), CAESAR SP Howitzer, Canadair CT-114 (No 3D), CH-148 Cyclone (No 3D), CH-149 Cormorant, CH-46E Sea Knight, CH-47 Chinook (No 3D), CH-53E Super Stallion, Chilean AT4, Chilean IMI Galil, Chilean M16-M203, Chilean M16, Chilean M240, Chilean M249, Chilean M4, Chilean M60, Colombian Police Officer, Colombian Soldier, Cougar FSV (No 3D), Coyote APC (No 3D), Cypher VTUAV, Dabur-class Patrol Boat (No 3D), Defense Satellite, DI AK-47, DI Lasing (CIS), DI Lasing (US), DI RPG, DI SMAW And Rifle, Dolphin-class Submarine (No 3D), Dragon Warrior VTUAV, E-2 Hawkeye, E-3 Sentry, EA-6B Prowler, EC145 (No 3D), Echo-class Survey Ship (No 3D), EH101 Merlin (No 3D), Eurofighter Typhoon, F-117A Nighthawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, Fridtjof Nansen-class (No 3D), FV 510 Warrior (No 3D), FV101 Scorpion CVR, FV107 Scimitar (No 3D), FV4030/4 Challenger MBT (No 3D), FV4034 Challenger 2 MBT (No 3D), FV432 APC (No 3D), F/A-18 Hornet, GAZ-69 Utility Vehicle, Grizzly APC (No 3D), Guided Missile Destroyer, Guided Missile Frigate, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Hermes 450 (No 3D), HH-65 Dolphin (No 3D), HMMWV Utility Vehicle, HMMWV with Avenger, HMMWV with M2, HMMWV with Shelter, HMMWV with TOW launcher, Husky ARV (No 3D), Insurgent, Invincible-class Carrier (No 3D), Iranian Soldier, Iraqi Police Officer, Iraqi Soldier, Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), Iveco LMV (No 3D), Jackal MWMIK (No 3D)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Spot Report Generator	L-class Frigate (No 3D), K1A1 MBT (No 3D), KA-50 Hokum, M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Kingston-class Patrol vessel (No 3D), L 14 Albion (No 3D), L-118 Howitzer (No 3D), Land Rover Wolf (No 3D), LAV III APC (No 3D), LCAC, Leopard 2 Tank, LG1 Howitzer (No 3D), Libyan Police Officer, Libyan Soldier, Lockheed L-1011 (No 3D), Los Angeles-Class Submarine, LPH01 Ocean (No 3D), Harpers Ferry-class LSD, M-71 Howitzer (No 3D), M109 Howitzer, M1126 Stryker ICV, M113 APC, M1A2 Abrams MBT, M2A2 Bradley IFV, M35 Truck, M3A2 Bradley CFV, M577A2 Command Post, M58 MICLIC, M777 Howitzer (No 3D), M88 Medium Recovery Vehicle, M939A2 5-Ton Truck, M977 HEMTT Cargo Truck, M978 HEMTT Fuel Truck, M993 MLRS, M9 ACE, Mamba APC (No 3D), Maveric UAS (No 3D), MBT 2000 (No 3D), MD 500 (No 3D), Merkava III MBT (No 3D), Merkava IV MBT (No 3D), MH-47 Chinook (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MH-6 Little Bird (No 3D), Mi-24 Hind, Mi-28 Havoc, Mi-2 Hoplite, Mideast Combatant, Mideast Combatant with RPG, MiG-27 Flogger, MiG-29 Fulcrum, MIM-104 Patriot Launcher, Mirage 2000, Mirage F1, MQ-9 Reaper UAV, MRAP-ATV (No 3D), MRAP-CAT II Cougar (No 3D), MRAP-CAT I MaxxPro (No 3D), MT-LB APC, Namer APC (No 3D), Navistar 7000 (No 3D), Nimitz-class Carrier, OH-58 Kiowa, Ohio-class Submarine, Oksoy-class Minehunter (No 3D), P-3 Orion, Pakistani Police Officer, Pakistani Soldier, Police Officer, MQ-1 Predator, PT-91 Twardy MBT (No 3D), Puma CEV (No 3D), Queen Elizabeth-class Carrier (No 3D), RAH-66 Comanche, Rapier SAM (No 3D), REMUS UUV (No 3D), Rigid-Hulled Inflatable Boat, River-class Patrol Vessel (No 3D), Romeo-class Submarine, RQ-11 Raven UAV, RQ-4 Global Hawk UAV (No 3D), RQ-7 Shadow UAV (No 3D), SA 330 Puma (No 3D), SA-15 Gauntlet SAM, SA-19 Grison, SA-6 Gainful SAM, SA-9 Gaskin SAM System, Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Saar 5-class Corvette (No 3D), ScanEagle UAV (No 3D), Scorpene-class Submarine (No 3D), Sergeant M16, SH-3 Sea King (No 3D), SH-60 Seahawk, Skjold-class Patrol (No 3D), Somali Police Officer, Somali Soldier, Song-class Submarine, Sonobuoy (Passive), Sovremenny-class Destroyer, Space Shuttle, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker, Sudanese Police Officer, Sudanese Soldier, Super Dvora Mark II (No 3D), Suspect, Syrian Police Officer, Syrian Soldier, T-69 MBT, T-72 MBT, T-80 MBT, T-84 MBT (No 3D), T90A MBT (No 3D), Taliban, Taurus ARV (No 3D), Technical Truck, Tiger-class Fast Attack Craft (No 3D), Tornado ADV

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Spot Report Generator (continued)	Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 10 MBT (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Type 90 Kyu-maru MBT (No 3D), Type 99 MBT (No 3D), Udaloy-class Destroyer (No 3D), Ugandan Police Officer, Ugandan Soldier, UH-1N Twin Huey (No 3D), UH-60 Blackhawk, UH-72 Lakota (No 3D), UK Soldier, Ula-class Submarine (No 3D), US Army Medic, US EOD, USMC M16-M203, USMC M16, USMC M240, USMC M249, USMC M4, US Army AT4, US Army Javelin, US Army M16-M203, US Army M16, US Army M240, US Army M249, US Army M4, US Army M60, US Army M9, V-22 Osprey (No 3D), VAB APC, Vanguard-class Submarine (No 3D), VBL ATV, Victoria-class (No 3D), VM 90 LSVW (No 3D), Walrus-class submarine (No 3D), Westland Lynx (No 3D), WS-61 Sea King (No 3D), XM7 FIST-Bradley, ZIL-135 8x8 Truck, ZPU-4 AA Gun, ZSU-23-4 Shilka

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Spot Report Receiver	A-10 Thunderbolt, A-4 Skyhawk (No 3D), Aardvark JSFU (No 3D), AAVC7A1 Landing Vehicle, Achzarit APC (No 3D), Actros AHSVS (No 3D), Afghan Police Officer, Afghan Soldier, AH-1W SuperCobra, AH-64A Apache, AH-6 Little Bird (No 3D), Aircraft Carrier, AMX-30 MBT, AMX-56 Leclerc MBT, AN/BLQ-11 UUV (No 3D), Arjun MBT (No 3D), Arleigh Burke-class Destroyer, AS-365 Dauphin (No 3D), AS-565 Panther (No 3D), AS-90 Artillery, AV-8B Harrier II, B-2 Spirit, Badger AEV (No 3D), BAE CT-155 Hawk (No 3D), Bay-class DLS (No 3D), Beaver AVLB (No 3D), Bell 412 (No 3D), Bell 206 JetRanger (No 3D), Bell 406 (No 3D), Bison APC (No 3D), BMP-2 AFV, BN-2 Islander (No 3D), Brazilian Soldier, BTR-80 APC, Buffalo MRAP III (No 3D), Buffel ARV (No 3D), C-130 Hercules, C-17 Globemaster III (No 3D), C-5 Galaxy (No 3D), C1 Ariete MBT (No 3D), CAESAR SP Howitzer, Canadair CT-114 (No 3D), CH-148 Cyclone (No 3D), CH-149 Cormorant, CH-46E Sea Knight, CH-47 Chinook (No 3D), CH-53E Super Stallion, Chilean AT4, Chilean IMI Galil, Chilean M16-M203, Chilean M16, Chilean M240, Chilean M249, Chilean M4, Chilean M60, Colombian Police Officer, Colombian Soldier, Cougar FSV (No 3D), Coyote APC (No 3D), Dabur-class Patrol Boat (No 3D), Defense Satellite, DI AK-47, DI Lasing (CIS), DI Lasing (US), DI RPG, DI SMAW And Rifle, Dolphin-class Submarine (No 3D), E-2 Hawkeye, E-3 Sentry, EA-6B Prowler, EC145 (No 3D), Echo-class Survey Ship (No 3D), EH101 Merlin (No 3D), Eurofighter Typhoon, F-117A Night-hawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, Fridtjof Nansen-class (No 3D), FV 510 Warrior (No 3D), FV101 Scorpion CVR, FV107 Scimitar (No 3D), FV4030/4 Challenger MBT (No 3D), FV4034 Challenger 2 MBT (No 3D), FV432 APC (No 3D), F/A-18 Hornet, GAZ-69 Utility Vehicle, Grizzly APC (No 3D), Guided Missile Destroyer, Guided Missile Frigate, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Hermes 450 (No 3D), HH-65 Dolphin (No 3D), HMMWV Utility Vehicle, HMMWV with Avenger, HMMWV with M2, HMMWV with Shelter, HMMWV with TOW launcher, Husky ARV (No 3D), Insurgent, Invincible-class Carrier (No 3D), Iranian Soldier, Iraqi Police Officer, Iraqi Soldier, Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), Iveco LMV (No 3D), Jackal MWMIK (No 3D), L-class Frigate (No 3D), K1A1 MBT (No 3D), KA-50 Hokum

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Spot Report Receiver	M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Kingston-class Patrol vessel (No 3D), L 14 Albion (No 3D), L-118 Howitzer (No 3D), Land Rover Wolf (No 3D), LAV III APC (No 3D), LCAC, Leopard 2 Tank, LG1 Howitzer (No 3D), Libyan Police Officer, Libyan Soldier, Lockheed L-1011 (No 3D), Los Angeles-Class Submarine, LPH01 Ocean (No 3D), Harpers Ferry-class LSD, M-71 Howitzer (No 3D), M109 Howitzer, M1126 Stryker ICV, M113 APC, M1A2 Abrams MBT, M2A2 Bradley IFV, M35 Truck, M3A2 Bradley CFV, M577A2 Command Post, M58 MICLIC, M777 Howitzer (No 3D), M88 Medium Recovery Vehicle, M939A2 5-Ton Truck, M977 HEMTT Cargo Truck, M978 HEMTT Fuel Truck, M993 MLRS, M9 ACE, Mamba APC (No 3D), Maveric UAS (No 3D), MBT 2000 (No 3D), MD 500 (No 3D), Merkava III MBT (No 3D), Merkava IV MBT (No 3D), MH-47 Chinook (No 3D), MH-60L Black Hawk DAP (No 3D), MH-60 Black Hawk, MH-6 Little Bird (No 3D), Mi-24 Hind, Mi-28 Havoc, Mi-2 Hoplite, Mideast Combatant, Mideast Combatant with RPG, MiG-27 Flogger, MiG-29 Fulcrum, MIM-104 Patriot Launcher, Mirage 2000, Mirage F1, MQ-9 Reaper UAV, MRAP-ATV (No 3D), MRAP-CAT II Cougar (No 3D), MRAP-CAT I MaxxPro (No 3D), MT-LB APC, Namer APC (No 3D), Navistar 7000 (No 3D), Nimitz-class Carrier, OH-58 Kiowa, Ohio-class Submarine, Oksoy-class Minehunter (No 3D), P-3 Orion, Pakistani Police Officer, Pakistani Soldier, Police Officer, MQ-1 Predator, PT-91 Twardy MBT (No 3D), Puma CEV (No 3D), Queen Elizabeth-class Carrier (No 3D), RAH-66 Comanche, Rapier SAM (No 3D), REMUS UUV (No 3D), Rigid-Hulled Inflatable Boat, River-class Patrol Vessel (No 3D), Romeo-class Submarine, RQ-11 Raven UAV, RQ-4 Global Hawk UAV (No 3D), RQ-7 Shadow UAV (No 3D), SA 330 Puma (No 3D), SA-15 Gauntlet SAM, SA-19 Grison, SA-6 Gainful SAM, SA-9 Gaskin SAM System, ScanEagle UAV (No 3D), Scorpene-class Submarine (No 3D), Sergeant M16, SH-3 Sea King (No 3D), SH-60 Seahawk, Skjold-class Patrol (No 3D), Somali Police Officer, Somali Soldier, Song-class Submarine, Sovremenny-class Destroyer, Space Shuttle, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker, Sudanese Police Officer, Sudanese Soldier, Super Dvora Mark II (No 3D), Suspect, Syrian Police Officer, Syrian Soldier, T-69 MBT, T-72 MBT, T-80 MBT, T-84 MBT (No 3D), T90A MBT (No 3D), Taliban, Taurus ARV (No 3D), Technical Truck, Tiger-class Fast Attack Craft (No 3D), Tornado ADV, Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 10 MBT (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Type 90 Kyu-maru MBT (No 3D)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Spot Report Receiver (continued)	Type 99 MBT (No 3D), Udaloy-class Destroyer (No 3D), Ugandan Police Officer, Ugandan Soldier, UH-1N Twin Huey (No 3D), UH-60 Blackhawk, UH-72 Lakota (No 3D), UK Soldier, Ula-class Submarine (No 3D), US Army Medic, US EOD, USMC M16-M203, USMC M16, USMC M240, USMC M249, USMC M4, US Army AT4, US Army Javelin, US Army M16-M203, US Army M16, US Army M240, US Army M249, US Army M4, US Army M60, US Army M9, V-22 Osprey (No 3D), VAB APC, Vanguard-class Submarine (No 3D), VBL ATV, Victoria-class (No 3D), VM 90 LSVW (No 3D), Walrus-class submarine (No 3D), Westland Lynx (No 3D), WS-61 Sea King (No 3D), XM7 FIST-Bradley, ZIL-135 8x8 Truck, ZPU-4 AA Gun, ZSU-23-4 Shilka
Stinger Missile Launcher	HMMWV with Avenger
Subsurface Entity Default	AN/BLQ-11 UUV (No 3D), Astute-class Submarine (No 3D), Dolphin-class Submarine (No 3D), Los Angeles-Class Submarine, Mk-46 MOD 5 Torpedo (No 3D), Mk-48 ADCAP Torpedo, Ohio-class Submarine, REMUS UUV (No 3D), Romeo-class Submarine, Scorpene-class Submarine (No 3D), Song-class Submarine, Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Ula-class Submarine (No 3D), Vanguard-class Submarine (No 3D), Victoria-class (No 3D), Walrus-class submarine (No 3D)
Subsurface	AN/BLQ-11 UUV (No 3D), Astute-class Submarine (No 3D), Dolphin-class Submarine (No 3D), Los Angeles-Class Submarine, Mk-46 MOD 5 Torpedo (No 3D), Mk-48 ADCAP Torpedo, Ohio-class Submarine, REMUS UUV (No 3D), Romeo-class Submarine, Scorpene-class Submarine (No 3D), Song-class Submarine, Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Ula-class Submarine (No 3D), Vanguard-class Submarine (No 3D), Victoria-class (No 3D), Walrus-class submarine (No 3D)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Surface Entity Default	Aircraft Carrier, Alta-class Minehunter (No 3D), AN/BLQ-11 UUV (No 3D), Arleigh Burke-class Destroyer, Astute-class Submarine (No 3D), Bay-class DLS (No 3D), Container Ship, Container Ship Loaded, Cruise Ship (No 3D), Dabur-class Patrol Boat (No 3D), Dolphin-class Submarine (No 3D), Durand de la Penne-class Destroyer (No 3D), Echo-class Survey Ship (No 3D), Fishing Boat, Guided Missile Destroyer, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Henry J Kaiser-class Oiler (No 3D), Horizon-class Destroyer (No 3D), Hunt-class Minehunter (No 3D), Iroquois-class Destroyer (No 3D), Island-class Patrol Vessel (No 3D), L-class Frigate (No 3D), Jet Ski, M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Kingston-class Patrol vessel (No 3D), L 14 Albion (No 3D), Los Angeles-Class Submarine, LPH01 Ocean (No 3D), Harpers Ferry-class LSD, Osprey-class Mine CMS (No 3D), Mk-46 MOD 5 Torpedo (No 3D), Mk-48 ADCAP Torpedo, Nimitz-class Carrier, Ohio-class Submarine, Oksoy-class Minehunter (No 3D), Queen Elizabeth-class Carrier (No 3D), REMUS UUV (No 3D), Rigid-Hulled Inflatable Boat, Rigid-Hulled Inflatable Boat - Civilian, River-class Patrol Vessel (No 3D), Romeo-class Submarine, Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Saar 5-class Corvette (No 3D), Sailboat, Sandown-class Minehunter (No 3D), Scorpene-class Submarine (No 3D), Skjold-class Patrol (No 3D), Song-class Submarine, Sovremenny-class Destroyer, Speedboat, Super Dvora Mark II (No 3D), Super Tanker (No 3D), Trafalgar-class Submarine (No 3D), Type 209 Submarine (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D), Ula-class Submarine (No 3D), Vanguard-class Submarine (No 3D), Victoria-class (No 3D), Walrus-class submarine (No 3D)
Surface Disaggregated Movement	

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Large Ship	Aircraft Carrier, Alta-class Minehunter (No 3D), Arleigh Burke-class Destroyer, Bay-class DLS (No 3D), Container Ship, Container Ship Loaded, Cruise Ship (No 3D), Durand de la Penne-class Destroyer (No 3D), Fishing Boat, Fridtjof Nansen-class (No 3D), Guided Missile Destroyer, Guided Missile Frigate, Halifax-class Frigate (No 3D), Henry J Kaiser-class Oiler (No 3D), Horizon-class Destroyer (No 3D), Hunt-class Minehunter (No 3D), Invincible-class Carrier (No 3D), Iroquois-class Destroyer (No 3D), L-class Frigate (No 3D), M-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Kingston-class Patrol vessel (No 3D), L 14 Albion (No 3D), LPH01 Ocean (No 3D), Harpers Ferry-class LSD, Osprey-class Mine CMS (No 3D), Nimitz-class Carrier, Oksoy-class Minehunter (No 3D), Queen Elizabeth-class Carrier (No 3D), Sandown-class Minehunter (No 3D), Sovremenny-class Destroyer, Super Tanker (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D)
Surface Multiple Hit Damage	Guided Missile Frigate
Small Surface Ships	
Tactical Radar Jammer	E-2 Hawkeye, EA-6B Prowler
Torpedo Warhead	Mk-46 MOD 5 Torpedo (No 3D), Mk-48 ADCAP Torpedo
TOW Missile Launcher	HMMWV with TOW launcher, MRAP-ATV (No 3D), MRAP-CAT II Cougar (No 3D), MRAP-CAT I MaxxPro (No 3D), VM 90 LSVW (No 3D)
US Fighter Bomber Bomb Bay	A-10 Thunderbolt, A-4 Skyhawk (No 3D), AV-8B Harrier II, Eurofighter Typhoon, F-117A Nighthawk, F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), F-35 Lightning II, F/A-18 Hornet, Mirage 2000, Mirage F1, Tornado ADV
US Heavy Bomber Bomb Bay	B-2 Spirit, P-3 Orion
Vertical SAM Missile Launcher	Alta-class Minehunter (No 3D), Arleigh Burke-class Destroyer, Durand de la Penne-class Destroyer (No 3D), Guided Missile Destroyer, Guided Missile Frigate, Halifax-class Frigate (No 3D), Hauk-class Fast Attack (No 3D), Horizon-class Destroyer (No 3D), Iroquois-class Destroyer (No 3D), L-class Frigate (No 3D), KD-III-class Destroyer (No 3D), Oksoy-class Minehunter (No 3D), Saar 4-class Missile Boat (No 3D), Saar 4.5-class Missile Boat (No 3D), Saar 5-class Corvette (No 3D), Skjold-class Patrol (No 3D), Sovremenny-class Destroyer, Tiger-class Fast Attack Craft (No 3D), Type 42 Destroyer (No 3D), Type 45 Destroyer (No 3D), Type 22 Frigate (No 3D), Type 23 Frigate (No 3D), Udaloy-class Destroyer (No 3D)

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Visual Sensor	A-10 Thunderbolt, Aardvark JSFU (No 3D), AAVC7A1 Landing Vehicle, Achzarit APC (No 3D), Actros AHSVS (No 3D), Afghan Police Officer, Afghan Soldier, AMX-30 MBT, AMX-56 Leclerc MBT, Arjun MBT (No 3D), AS-90 Artillery, Badger AEV (No 3D), Beaver AVLB (No 3D), Bison APC (No 3D), BMP-2 AFV, Brazilian Soldier, BTR-80 APC, Buffalo MRAP III (No 3D), Buffel ARV (No 3D), C1 Ariete MBT (No 3D), CAESAR SP Howitzer, Car Bomb, CH-46E Sea Knight, Chilean AT4, Chilean IMI Galil, Chilean M16-M203, Chilean M16, Chilean M240, Chilean M249, Chilean M4, Chilean M60, Colombian Police Officer, Colombian Soldier, Container Ship, Container Ship Loaded, Cougar FSV (No 3D), Coyote APC (No 3D), Cruise Ship (No 3D), DI AK-47, DI Lasing (CIS), DI Lasing (US), DI RPG, DI SMAW And Rifle, Durand de la Penne-class Destroyer (No 3D), F-15 Eagle, F-16A Fighting Falcon, F-22 Raptor (No 3D), Fishing Boat, FV 510 Warrior (No 3D), FV101 Scorpion CVR, FV107 Scimitar (No 3D), FV4030/4 Challenger MBT (No 3D), FV4034 Challenger 2 MBT (No 3D), FV432 APC (No 3D), F/A-18 Hornet, GAZ-69 Utility Vehicle, Grizzly APC (No 3D), HMMWV Utility Vehicle, HMMWV with M2, HMMWV with Shelter, HMMWV with TOW launcher, Husky ARV (No 3D), Insurgent, Invincible-class Carrier (No 3D), Iranian Soldier, Iraqi Police Officer, Iraqi Soldier, Iveco LMV (No 3D), Jackal MWMIK (No 3D), L-class Frigate (No 3D), Jet Ski, K1A1 MBT (No 3D), M-class Frigate (No 3D), Land Rover Wolf (No 3D), LAV III APC (No 3D), LCAC, Leopard 2 Tank, Libyan Police Officer, Libyan Soldier, M109 Howitzer, M1126 Stryker ICV, M113 APC, M1A2 Abrams MBT, M2A2 Bradley IFV, M35 Truck, M3A2 Bradley CFV, M577A2 Command Post, M58 MICLIC, M88 Medium Recovery Vehicle, M939A2 5-Ton Truck, M977 HEMTT Cargo Truck, M978 HEMTT Fuel Truck, M993 MLRS, M9 ACE, Mamba APC (No 3D), Maveric UAS (No 3D), MBT 2000 (No 3D), Merkava III MBT (No 3D), Merkava IV MBT (No 3D), Mideast Combatant, Mideast Combatant with RPG, Mirage 2000, MRAP-ATV (No 3D), MRAP-CAT II Cougar (No 3D), MRAP-CAT I MaxxPro (No 3D), MT-LB APC, Namer APC (No 3D), Navistar 7000 (No 3D), Pakistani Police Officer, Pakistani Soldier, Police Officer, PT-91 Twardy MBT (No 3D), Puma CEV (No 3D), Rigid-Hulled Inflatable Boat, Rigid-Hulled Inflatable Boat - Civilian, Roadside IED, RQ-11 Raven UAV, RQ-7 Shadow UAV (No 3D), Sailboat, ScanEagle UAV (No 3D), Sergeant M16

Table C-2: System usage, EntityLevel.sms

System	Entities that Use System
Visual Sensor (continued)	Somali Police Officer, Somali Soldier, Speedboat, SU-25 Frogfoot, SU-27 Flanker, Su-37 Flanker, Sudanese Police Officer, Sudanese Soldier, Suicide Bomber, Super Dvora Mark II (No 3D), Super Tanker (No 3D), Suspect, Syrian Police Officer, Syrian Soldier, T-69 MBT, T-72 MBT, T-80 MBT, T-84 MBT (No 3D), T90A MBT (No 3D), Taliban, Taurus ARV (No 3D), Technical Truck, Type 10 MBT (No 3D), Type 90 Kyu-maru MBT (No 3D), Type 99 MBT (No 3D), Ugandan Police Officer, Ugandan Soldier, UK Soldier, US Army Medic, US EOD, USMC M16-M203, USMC M16, USMC M240, USMC M249, USMC M4, US Army AT4, US Army Javelin, US Army M16-M203, US Army M16, US Army M240, US Army M249, US Army M4, US Army M60, US Army M9, VAB APC, VBL ATV, VM 90 LSVW (No 3D), XM7 FIST-Bradley, ZIL-135 8x8 Truck, ZPU-4 AA Gun, ZSU-23-4 Shilka

Table C-3: Systems and Configured Scripted Tasks, EntityLevel.sms

System	Scripted Tasks
Anti-Submarine Missile (Vertically Launched)	Launch_ASW_Missile
DDG Embedded Support Craft (LAMPS/RHIB)	Embedded_Sector_SAR
Fighter Jet	Fixed_Wing_Takeoff, Fixed_Wing_Land
Heavy Plane	Fixed_Wing_Takeoff, Fixed_Wing_Land
High Maneuverability Fighter	Fixed_Wing_Takeoff, Fixed_Wing_Land
GAU-8 Avenger	Strafe_Ground_Target, Attack_Aircraft_with_Guns
GSh-301 Cannon	Strafe_Ground_Target, Attack_Aircraft_with_Guns
Homing Torpedo Capability (Forward Launched)	Delayed_homing, Delayed_homing_ASW, Launch_Torpedo_Salvo
M61 Vulcan	Strafe_Ground_Target, Attack_Aircraft_with_Guns
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
Periscope	Raise_Periscope, Lower_Periscope

Table C-3: Systems and Configured Scripted Tasks, EntityLevel.sms

System	Scripted Tasks
Dipping SONAR Sensor	Sonar_Dip
Sonobuoy Deployer	Deploy_Sonobuoy, Deploy_-Sonobuoys_Along_Route

Table C-4: System descriptions, AggregateLevel.sms

System	Category	Entity Types	Description
Mortar, 120mm	weapon	aggregate, assemblies	A single 120mm mortar tube. Capable of indirect fire missions.
Active RADAR Sensor	sensor	all	Allows an entity to detect other objects through RADAR. Also publishes emitter beams that represent the emissions from the sensor.
Aerial Refueling Resupply	other	all	Gives the unit the capability to resupply aviation fuel to airborne units from an airborne unit.
122mm Howitzer	weapon	aggregate, assemblies	Artillery piece capable of indirect fire missions.
152mm Howitzer Auto	weapon	aggregate, assemblies	Artillery piece capable of indirect fire missions. Uses auto-loader.
155mm Howitzer	weapon	aggregate, assemblies	Artillery piece capable of indirect fire missions.
Bomb, 2000lb JDAM	weapon	aggregate, assemblies	A precision guided bomb. Uses GPS
Mk 45 5-inch Gun	weapon	aggregate, assemblies	A naval gun capable of indirect fire operations.
9K330 Surface-to-Air Missile Launcher	weapon	aggregate, assemblies	An anti-air missile. Uses radar guidance.
AIM-120 AMRAAM Missile	weapon	aggregate, assemblies	A medium range anti-air missile. Uses radar guidance.
Communications Jammer	electronic warfare	all	Jamming system for radios, cell phones, satellite phones, UAVs, etc.
Harpoon Antiship Missile	weapon	aggregate, assemblies	A long range anti-ship missile. Radar guided.

Table C-4: System descriptions, AggregateLevel.sms

System	Category	Entity Types	Description
Torpedo, Mark 54	weapon	aggregate, assemblies	An anti-ship and anti-submarine torpedo. Can be air-dropped.
Radar Jammer	electronic warfare	all	Jamming system for acquisition and tracking/guidance radars.
Guided Bomb	weapon	aggregate, assemblies	A precision guided bomb. Uses optical or other non-radar guidance.
RIM-66 SAM	weapon	aggregate, assemblies	A surface to air missile. Uses radar guidance.
Sidewinder Anti-Air Missile	weapon	aggregate, assemblies	A short range anti-air missile. Uses IR guidance.
Stinger Anti-Air Missile	weapon	aggregate, assemblies	An anti-air missile with limited altitude. Uses IR guidance.
Tomahawk Cruise Missile	weapon	aggregate, assemblies	A long range guided land attack missile. Uses primarily GPS and optical guidance.
Unguided Bomb	weapon	aggregate, assemblies	A bomb intended to have effects in an area.
Torpedo, Type 40 (RPK-6 Vodopad SUBROC)	weapon	aggregate, assemblies	An anti-ship and anti-submarine torpedo with SUBROC configuration. Can be deck-launched.
Air Defense RADAR Sensor	sensor	all	Allows an entity to detect aircraft through RADAR. Also publishes emitter beams that represent the emissions from the sensor.
Air-Drop Supply	other	all	Gives the unit the capability to resupply ground units from the air.
Active RADAR Sensor	sensor	all	Allows an entity to detect other objects through RADAR. Also publishes emitter beams that represent the emissions from the sensor.
Aircraft Aggregated Movement	aggregated	aggregate	Aircraft aggregated movement.
Biological Weapon	weapon	aggregate, assemblies	A weapon system that create biological contamination areas.
Booby Trap Setter	engineering-object-creation	aggregate, assemblies	A system which provides the capability to create booby traps.

Table C-4: System descriptions, AggregateLevel.sms

System	Category	Entity Types	Description
Bridge Builder	engineering-object-creation	aggregate, assemblies	A system which provides the capability to create bridges.
Chemical Weapon	weapon	aggregate, assemblies	A weapon system that create chemical contamination areas.
Ditch Digger	engineering-object-creation	aggregate, assemblies	A system which provides the capability to create a ditch.
SIGINT Sensor	sensor	all	Allows an entity to detect other entities by their radio and RADAR emissions.
Engagement Report Generator	other	all	Allows the entity to send engagement reports through its radio when it is attacking an enemy.
Engineering Object Sensor	sensor	all	Allows an entity to detect engineering objects.
Engineering Obstacle Destroyer	obstacle-destruction	aggregate, assemblies	A system which provides the capability to destroy/breach engineering obstacles. Meant to be used by units with specialized equipment.
Explosive Hazard and Simple Obstacle Destroyer	obstacle-destruction	aggregate, assemblies	A system which provides the capability to destroy explosive hazards as well as destroy/breach simple obstacles.
FASCAM	weapon	aggregate, assemblies	Artillery used to deploy minefields.
Fortification Builder	engineering-object-creation	aggregate, assemblies	A system which provides the capability to construct fortifications.
Ground Aggregated Movement	aggregated	aggregate	Basic movement capabilities for any ground-based aggregated unit, both vehicle and DI.
Ground Combat Behaviors	other	all	Enables behaviors appropriate for ground combat units.
IFF Transponder	other	all	Sends IFF data, when turned on. IFF is off by default.

Table C-4: System descriptions, AggregateLevel.sms

System	Category	Entity Types	Description
Infantry Aggregated Movement	aggregated	aggregate	Infantry aggregated movement.
IR Sensor	sensor	all	Allows an entity to detect other objects through Infrared.
Limit Entity Existence	other	all	Used to control the life span of the object this system is attached to. Once it's life span is up the object is removed from the simulation.
Mechanized Aggregated Movement	aggregated	aggregate	Mechanized aggregated movement.
Mine Dispenser	engineering-object-creation	aggregate, assemblies	A system which provides the capability to create minefields.
MOPP Gear	other	aggregate, assemblies	Equips the unit with MOPP gear which allows MOPP level transitions.
Motorized Aggregated Movement	aggregated	aggregate	Motorized aggregated movement.
NBC Sensor	sensor	all	Allows an entity to detect areas contaminated with nuclear, biological, or chemical agents.
Nuclear Weapon	weapon	aggregate, assemblies	A weapon system that create nuclear contamination areas.
Obstacle Builder	engineering-object-creation	aggregate, assemblies	A system which provides the capability to construct basic obstacles.
Passive RADAR Sensor	sensor	all	Allows an entity to detect other objects through RADAR. No emitters are published for this sensor.
Ground Resupply	other	all	Gives the unit the capability to resupply units on the ground.
Riverine Aggregated Movement	aggregated	aggregate	Riverine movement system for aggregate of one light patrol craft.
Ship Aggregated Movement	aggregated	aggregate	Ship movement system for aggregate of one type ships.

Table C-4: System descriptions, AggregateLevel.sms

System	Category	Entity Types	Description
Simple Obstacle Destroyer	obstacle-destruction	aggregate, assemblies	A system which provides the capability to destroy/breach simple obstacles. Meant to be used by units without specialized equipment.
SONAR Sensor	sensor	all	Allows an entity to detect other objects through SONAR.
Spot Report Generator	other	all	Allows the entity to send spot reports through its radio when its sensors detect other entities in the scenario.
Spot Report Receiver	other	all	Allows an entity to receive spot reports from other entities over the radio, and add these contacts to its list of known entities. By default, spot reports expire after 5 minutes.
Subsurface Aggregated Movement	aggregated	aggregate	Subsurface movement system for Aggregate SMS submarines.
Tank Aggregated Movement	aggregated	aggregate	Tank aggregated movement.
Visual Sensor	sensor	all	Allows an entity to detect other objects through visible light.

Table C-5: System usage, AggregateLevel.sms

System	Entities that Use System
Mortar, 120mm	AR CAV TRP US, Mortar PLT US, Mortar SP PLT US
Active RADAR Sensor	EA-18G Growler, F-16C (Air Superiority), F/A-18 (Ground Attack), F/A-18 (Ship Attack), Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry, Sub, RU SSP, Lada
Aerial Refueling Resupply	
122mm Howitzer	Artillery BTY RU
152mm Howitzer Auto	Artillery SP BTY RU
155mm Howitzer	Artillery SP BTY US, Artillery BTY US
Bomb, 2000lb JDAM	F/A-18 (Ground Attack)
Mk 45 5-inch Gun	Ship, US DDG, Arleigh Burke

Table C-5: System usage, AggregateLevel.sms

System	Entities that Use System
9K330 Surface-to-Air Missile Launcher	SAM BTY RU
AIM-120 AMRAAM Missile	F-16C (Air Superiority), F/A-18 (Ground Attack)
Communications Jammer	MI PLT US
Harpoon Antiship Missile	F/A-18 (Ship Attack), Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry
Torpedo, Mark 54	Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry
Radar Jammer	EA-18G Growler
Guided Bomb	
RIM-66 SAM	Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry
Sidewinder Anti-Air Missile	EA-18G Growler, F-16C (Air Superiority), F/A-18 (Ground Attack), F/A-18 (Ship Attack)
Stinger Anti-Air Missile	ADA PLT US
Tomahawk Cruise Missile	Ship, US DDG, Arleigh Burke
Unguided Bomb	
Torpedo, Type 40 (RPK-6 Vodopad SUBROC)	Sub, RU SSP, Lada
Air Defense RADAR Sensor	SAM BTY RU
Air-Drop Supply	
Active RADAR Sensor	F-16C (Air Superiority), F/A-18 (Ground Attack), F/A-18 (Ship Attack)
Aircraft Aggregated Movement	Air Assault CO US, Air Attack CO US, Air Cav TRP US, EA-18G Growler, F-16C (Air Superiority), F/A-18 (Ground Attack), F/A-18 (Ship Attack), _Air FW template, _Air RW template
Biological Weapon	
Booby Trap Setter	Eng Mob Aug CO
Bridge Builder	Eng Mob Aug CO
Chemical Weapon	Artillery SP BTY RU
Ditch Digger	Eng Mob Aug CO, Mech Inf CO US
SIGINT Sensor	MI PLT US

Table C-5: System usage, AggregateLevel.sms

System	Entities that Use System
Engagement Report Generator	ADA PLT US, Air Assault CO US, Air Attack CO US, Air Cav TRP US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, CSS CO, EA-18G Growler, Eng Clearance CO, Eng Mob Aug CO, F-16C (Air Superiority), Artillery BTY US, F/A-18 (Ground Attack), F/A-18 (Ship Attack), Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry, MI PLT US, Sub, RU SSP, Lada, Tank CO RU, Tank CO US, Weapons CO US, _Air FW template, _Air RW template, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template
Engineering Object Sensor	ADA PLT US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, CSS CO, Eng Clearance CO, Eng Mob Aug CO, Artillery BTY US, Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry, MI PLT US, Sub, RU SSP, Lada, Tank CO RU, Tank CO US, Weapons CO US, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template
Engineering Obstacle Destroyer	Eng Mob Aug CO
Explosive Hazard and Simple Obstacle Destroyer	Eng Clearance CO
FASCAM	Artillery BTY RU, Artillery SP BTY US, Artillery BTY US
Fortification Builder	Eng Mob Aug CO
Ground Aggregated Movement	
Ground Combat Behaviors	
IFF Transponder	Air Assault CO US, Air Attack CO US, Air Cav TRP US, EA-18G Growler, F-16C (Air Superiority), F/A-18 (Ground Attack), F/A-18 (Ship Attack), _Air FW template
Infantry Aggregated Movement	Mortar PLT US, Rifle CO RU, Rifle CO US, _Unarmored foot template
IR Sensor	ADA PLT US, Air Attack CO US, Air Cav TRP US, AR CAV TRP US, Tank CO US

Table C-5: System usage, AggregateLevel.sms

System	Entities that Use System
Limit Entity Existence	
Mechanized Aggregated Movement	Artillery SP BTY RU, Artillery SP BTY US, Eng Clearance CO, Eng Mob Aug CO, Mech Inf CO RU, Mech Inf CO US, Mortar SP PLT US, SAM BTY RU, Scout Hvy PLT US, _Light armor template, _Mechanized template, _SPA template
Mine Dispenser	Eng Mob Aug CO
MOPP Gear	ADA PLT US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, CSS CO, Eng Clearance CO, Eng Mob Aug CO, Artillery BTY US, Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, MI PLT US, Tank CO RU, Tank CO US, Weapons CO US, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template
Motorized Aggregated Movement	ADA PLT US, Artillery BTY RU, CSS CO, Artillery BTY US, Scout Motorized PLT US, MI PLT US, Weapons CO US, _Towed arty template, _Unarmored wheeled template
NBC Sensor	ADA PLT US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, CSS CO, Eng Clearance CO, Eng Mob Aug CO, Artillery BTY US, Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, MI PLT US, Tank CO RU, Tank CO US, Weapons CO US, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template
Nuclear Weapon	
Obstacle Builder	Eng Mob Aug CO
Passive RADAR Sensor	
Ground Resupply	CSS CO
Riverine Aggregated Movement	
Ship Aggregated Movement	Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry

Table C-5: System usage, AggregateLevel.sms

System	Entities that Use System
Simple Obstacle Destroyer	ADA PLT US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, Artillery BTY US, Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, MI PLT US, Tank CO RU, Tank CO US, Weapons CO US, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template
SONAR Sensor	Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry, Sub, RU SSP, Lada
Spot Report Generator	ADA PLT US, Air Assault CO US, Air Attack CO US, Air Cav TRP US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, CSS CO, EA-18G Growler, Eng Clearance CO, Eng Mob Aug CO, F-16C (Air Superiority), Artillery BTY US, F/A-18 (Ground Attack), F/A-18 (Ship Attack), Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry, MI PLT US, Sub, RU SSP, Lada, Tank CO RU, Tank CO US, Weapons CO US, _Air FW template, _Air RW template, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template
Spot Report Receiver	ADA PLT US, Air Assault CO US, Air Attack CO US, Air Cav TRP US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, EA-18G Growler, Eng Clearance CO, Eng Mob Aug CO, F-16C (Air Superiority), Artillery BTY US, F/A-18 (Ground Attack), F/A-18 (Ship Attack), Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry, MI PLT US, Sub, RU SSP, Lada, Tank CO RU, Tank CO US, Weapons CO US, _Air FW template, _Air RW template, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template
Subsurface Aggregated Movement	Sub, RU SSP, Lada

Table C-5: System usage, AggregateLevel.sms

System	Entities that Use System
Tank Aggregated Movement	AR CAV TRP US, Tank CO RU, Tank CO US, _Tank unit template
Visual Sensor	ADA PLT US, Air Assault CO US, Air Attack CO US, Air Cav TRP US, AR CAV TRP US, Artillery BTY RU, Artillery SP BTY RU, Artillery SP BTY US, CSS CO, EA-18G Growler, Eng Clearance CO, Eng Mob Aug CO, F-16C (Air Superiority), Artillery BTY US, F/A-18 (Ground Attack), F/A-18 (Ship Attack), Mech Inf CO RU, Mech Inf CO US, Mortar PLT US, Mortar SP PLT US, Rifle CO RU, Rifle CO US, SAM BTY RU, Scout Hvy PLT US, Scout Motorized PLT US, Ship, US DDG, Arleigh Burke, Ship, US FFG, Perry, MI PLT US, Sub, RU SSP, Lada, Tank CO RU, Tank CO US, Weapons CO US, _Air FW template, _Air RW template, _Ground master template, _Light armor template, _Mechanized template, _SPA template, _Tank unit template, _Towed arty template, _Unarmored foot template, _Unarmored wheeled template

Table C-6: Systems and Configured Scripted Tasks, AggregateLevel.sms

System	Scripted Tasks
Mortar, 120mm	Indirect_Fire, Limited_Munition_Attack
Aerial Refueling Resupply	Provide_Supplies, Change_Supplying_Command
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
Mk 45 5-inch Gun	Indirect_Fire, Limited_Munition_Attack
9K330 Surface-to-Air Missile Launcher	Limited_Munition_Attack, Attack_with_Anti-Air_Missile, Automatic_Air_Defense, React_To_Air_Threats
AIM-120 AMRAAM Missile	Limited_Munition_Attack, Attack_with_Anti-Air_Missile, Automatic_Air_Defense, React_To_Air_Threats
Communications Jammer	EW_Attack, Activate_Jammer, Deactivate_-Jammer
Harpoon Antiship Missile	Limited_Munition_Attack, Attack_with_Anti-ship_Missile, React_To_Naval_Threats_Missile
Torpedo, Mark 54	Limited_Munition_Attack, Attack_with_Torpedo, React_To_Naval_Threats_Torpedo
Radar Jammer	EW_Attack, Activate_Jammer, Deactivate_-Jammer

Table C-6: Systems and Configured Scripted Tasks, AggregateLevel.sms

System	Scripted Tasks
Guided Bomb	Bomb_Location, Limited_Munition_Attack, Ground_Attack
RIM-66 SAM	Limited_Munition_Attack, Attack_with_Anti-Air_Missile, Automatic_Air_Defense, React_To_Air_Threats
Sidewinder Anti-Air Missile	Limited_Munition_Attack, Attack_with_Anti-Air_Missile, React_To_Air_Threats
Stinger Anti-Air Missile	Limited_Munition_Attack, Attack_with_Anti-Air_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
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
Biological Weapon	Biological_Attack, NBC_Attack
Booby Trap Setter	Create_Booby_Trap, Improve_Booby_Trap
Bridge Builder	Construct_Bridge, Improve_Bridge
Chemical Weapon	Chemical_Attack, NBC_Attack
Ditch Digger	Create_Anti-Tank_Ditch, Improve_Ditch, Create_Dry_Gap
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
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

Table C-6: Systems and Configured Scripted Tasks, AggregateLevel.sms

System	Scripted Tasks
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
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



MÄK Products Glossary

This glossary defines terms used in MÄK 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 MÄK 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 MÄK 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 MÄK 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 MÄK 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 MÄK 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.

MÄK Technologies Lisp

An adaptation of the Lisp language used in configuration files for MÄK 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 MÄK 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 MÄK 3D applications, the point of view into the scene. (Called the eyepoint in MÄK 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 heartbeat) 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

-- command-line option 5-3, 5-11, 5-13, 54-3
  Simulation Object Editor 63-4
--additionalSystemScriptsDirectory command-line option 5-8
--alphaBits command-line option 5-3, 6-13
--antiAliasing command-line option 5-3, 6-13
--appDataDir command-line option 5-3, 5-8
  Simulation Object Editor 63-4
--appIdRange command-line option 5-8
--appNumber command-line option 5-6, 5-7, 5-8
--autoConnect command-line option 5-3
--backend command-line option 5-13
--batchedRendering command-line option 5-4
--col command-line option 54-3
--config command-line option 5-13
--countryCodesMappingFile command-line option 5-8
--daemon command-line option 5-8
--dataDir command-line option 5-3, 5-8
  Simulation Object Editor 63-4
--defaultObjectTimeout command-line option 5-7, 5-8
--defaultSimulationModelSet command-line option 5-3
--depthBits command-line option 6-13
--depthBuffer command-line option 5-3
--destAddrString command-line option 5-7, 5-12
--deviceAddress command-line option 5-7, 5-12
--diGuyAnimationsFile command-line option 5-8
--diGuyCharacerDataFile command-line option 5-8
--dis command-line option 5-3
--disableCallbackQueue command-line option 5-8
--disableKDTrees command-line option 5-4
--disableParallelTick command-line option 5-8
--disPort command-line option 5-7, 5-12
--dispSetting command-line option 5-3
--disVersion command-line option 5-7, 5-12, 5-20
--doNotCheckPluginVersions command-line option 5-3
  Simulation Object Editor 63-4
--doNotLoadVrfPlugins command-line option 5-3, 5-8
  Simulation Object Editor 63-4
--enableChannel command-line option 5-9
--encryptImages command-line option 54-3
--entDispSetting command-line option 5-3
--entTypeMap command-line option 5-3
--envSetting command-line option 5-3
--execname command-line option 5-7, 5-11
--exerciseId command-line option 5-7, 5-12
--extendLevelZero command-line option 54-3
--factoryRootDir command-line option 5-3
--federateName command-line option 5-6, 5-11
--federateType command-line option 5-7, 5-11
--fedFileName command-line option 5-6, 5-11
--file command-line option 54-3
--fileNotifyLevel command-line option 5-9
--fileTransporterReceivePort command-line option 5-4
--fomMapperInitData command-line option 5-6, 5-11
--fomMapperLib command-line option 5-6, 5-11
--fomModules command-line option 5-6, 5-11
--fowPerspective command-line option 5-4
--frameRate command-line option 5-4
--frontend command-line option 5-13
--frontEndPID command-line option 5-8
--fullScreen command-line option 5-4, 74-6
--fullyLoadTerrain command-line option 5-9

--generateFiles command-line option 54-4
--gui command-line option 5-4, 75-2
--help command-line option 5-4, 54-4
  Simulation Object Editor 63-4
--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 5-6, 5-7, 5-9
--ignore_rest command-line option 5-3, 5-11, 54-3
  Simulation Object Editor 63-4
--ignoreZoomForSpeedtreeLod command-line option 5-6
--launcherLocation command-line option, Simulation Object Editor 63-4
--layoutSettingsFile command-line option 5-4
  Simulation Object Editor 63-4
--level command-line option 54-4
--loadObservers command-line option 5-4
--loadPlugin command-line option 5-5, 5-9, 5-16
  Simulation Object Editor 63-4
--loadSimulationObjectGroup command-line option 5-5
--locale command-line option 5-4, 5-13
  Simulation Object Editor 63-4
--localSettingsDesignator command-line option 5-6, 5-11
--logFileName command-line option 5-4, 5-9
--mcastTtl command-line option 5-7, 5-12
--mimModule command-line option 5-11
--missing command-line option 54-4
--msdlSIDCMappingFile command-line option 5-9
--multicastAddresses command-line option 5-7, 5-12
--nonVrfObjectsUUIDScheme command-line option 5-5, 5-9
--noRtiCompilerCheck command-line option 5-9
--notifyLevel command-line option 5-5, 5-9
--noVSync command-line option 6-19
--numCallbackThreads command-line option 5-9
--opdEditorLocation command-line option, Simulation Object Editor 63-4
--outputLogFile command-line option 5-9
--plugin command-line option 5-5
--pseudoFullScreen command-line option 5-5, 74-6
--publishMftOnly command-line option 54-4
--pvd command-line option 5-5
--recvBufferSize command-line option 5-7, 5-12
--regenerateAll command-line option 54-3
--row command-line option 54-5
--rprFomRevision command-line option 5-7, 5-11
--rprFomVersion command-line option 5-7, 5-11
--scenarioFile command-line option 5-5
--searchDir command-line option 54-4
--sendBufferSize command-line option 5-7, 5-12
--sessionId command-line option 5-6, 5-7
--setMainThreadToHighPriority command-line option 5-10
--settingsDirectory command-line option, Simulation Object Editor 63-4
--settingsFile command-line option, vrfSim 5-8
--showConsole command-line option 5-3
  Simulation Object Editor 63-4
--simulationModelSet command-line option 5-10
--siteId command-line option 5-7
--smsFile command-line option, Simulation Object Editor 63-4
--startMinimized command-line option 5-10
--stealth command-line option 5-5
--stencilBits command-line option 5-5, 6-13
--subDir command-line option 54-4
--subnetMask command-line option 5-7, 5-12
--suppressHatIntersectOnAttach command-line option 5-5, 45-6
--suppressSelfReflect command-line option 5-12
--synchronizeOcean command-line option 5-5
--targetFrameRate command-line option 5-10, 6-17
--tcpSendTimeout command-line option 5-5
--textureSize command-line option 54-5
--threads command-line option 54-4
--timeManagement command-line option 4-20, 5-11, 5-18
--unplug command-line option 5-5
--useAbsoluteTimeStamps command-line option 5-6, 5-10
--useAdvisories command-line option 5-6, 5-11
--useAsyncIO command-line option 5-7, 5-12, 5-19
--useFileCommChannel command-line option 5-6
--useGeographicDdm command-line option 5-7
--useIpv6 command-line option 5-7, 5-12
--userDataDir command-line option 5-6, 5-10

--version command-line option 5-6, 5-10, 5-13, 54-4
  Simulation Object Editor 63-4
--visualTerrain command-line option, Simulation Object Editor 63-4
--vtFile command-line option 54-4
--waitQueue command-line option 5-11
-debug command-line option 5-13, B-10
-G command-line option 5-13
  Simulation Object Editor 63-4
-h command-line option, Simulation Object Editor 63-4
-K command-line option 5-4
-v command-line option, Simulation Object Editor 63-4
-Z command-line option 5-6
B-- command-line option 5-13, B-10
F-- command-line option 5-13, B-10

Symbols

@dis comment 52-25
@dis navigation_lights 52-25

Numerics

-1 (wildcard) 68-7
2525B icon 59-2
2D
  icon 16-9
    changing 64-18
    changing size 18-22
    configuring 59-2
    editing 59-2
    label 18-12
    scaling 18-22
    using image for 64-19
  label, shadow text 18-17
  observer, attachment behavior 45-14
  observer coordinate system 44-7
  projection, switching to 3D 43-2
  window 4-14
2D Icon model set 43-11
3D
  model 16-9
    changing 64-17
    DI-Guy 26-14
    licensing 1-22
  observer coordinate systems 44-7
  projection, switching to 2D 43-2
3D (continued)
  window 4-14
3D Cartesian coordinate system 29-18
3D Colorized Models model set 43-11
  color, changing 19-27
3D Models model set 43-11
3DS 56-4
  ASCII Export 64-28
  ASE 26-5
  ASE file 64-26
  file 47-3
3DS ASCII Export format 26-5
9-Line briefing 26-21

A

-A command-line option 5-7, 5-12, 5-19, 54-3
-a command-line option 5-6, 5-7, 5-8, 5-14
Abandon Plan command 32-24
abandoning plan 24-1, 32-24
  confirmation prompt 25-5
abatis, constructing 27-11
absolute, pathname 12-7
Absolute mode 45-6, 45-7
acceleration
  displaying 18-12
  specifying measurement unit 74-9
acceleration-to-rpm-factor parameter 9-18
accessibility check 20-18
action category 28-8, 28-17, 70-15
action table 67-11
actionCategories.tsk 28-17, 70-15
Activate Jammer task 27-4
activating
  jammer 27-4
  scenario event 8-10
  scenario event from plan 8-11
Activation set data request 30-5
active
  sonar 9-17
  stereo 73-21
  tactical graphic 30-5, 35-18
Active Sonar Mode dialog box 30-6
Active Sonar Mode set data request 30-6
active-radar-sensor system 22-18
activity
  adding 66-16
  unit 30-6
Activity dialog box 30-6
Activity set data request 66-16

actuator 15-7, 15-8
  damage 9-22
Add Conditional Expression dialog box 31-6
Add Group of Entities, dialog box 20-20
add image attribute 50-8
Add Model Definition dialog box 57-9, 67-3
Add Props page 50-26, 56-9, 56-13, 56-20
Add Terrain Patch dialog box 50-6, 56-3, 56-21
Add Visualizer Definition Attribute window 57-19
adding
  activities 66-16
  attribute, visual definition 57-19
  category 64-48
    Props Palette 59-13
  channel 73-5
  cockpit model 59-10
  component to object in OPD Editor 69-8
  conditional expression to plan 32-4
  default overlays 64-7
  DI-Guy animation 67-11
  DI-Guy character 67-10
  entity bitmap image 59-9
  entity to unit 23-7
  environment condition 11-7
  extended label 18-19
  feature layer 47-4, 50-11
  formation 65-9
  global commands to plan 32-15
  group of entities 20-20
  image 64-20
  key mapping 76-5
  linked event 8-6
  media to scenario event 8-5
  model, visual 64-20
  model mapping 58-3
  object, to Favorites list 16-26
  object geometry 64-13
  observer 43-6, 43-7
  ocean layer 50-15, 50-16
  parameter
    model definition 57-12
    schema 57-6
  plug-in 4-36
  prop 50-24
    from feature layer 50-26
  sensor component 71-11
  sensor signature 71-5, 71-12
  sensor type 71-9
  slot 64-33, 64-35, 64-38
  subordinate to unit 65-5
adding (continued)
  system to simulation object 64-23
  terrain patch to terrain 50-5
  terrain server 51-4
  text, to title bar 74-8
  text object 35-4
  vertex 35-13
  visual definition 57-16
  window 73-3
additional-opds parameter 12-6
additionalSystemScriptsDirectory parameter A-3
addMulticastAddr parameter A-7
adjusting, sound volume 42-3
advanced navigation 53-2
advisory message, enabling and disabling A-7
agent, contamination 22-7
aggregate, See also unit
Aggregate As
  dialog box 65-3
  specifying a unit as option 65-3
aggregate warfare model 22-2, 66-3
aggregate-comms-jammer, system 22-17
aggregated state 23-9
aggregated unit 21-2
  attack by fire 27-5
aggregate-formation-controller parameter 70-8
aggregate-level
  combat 22-7
  entity, configuring 66-3
  modeling 15-9
aggregate-level modeling 20-1
aggregate-level unit, parameter 66-4
AggregateLevel.sms 15-10, 22-2, 66-3
aggregate-object-param parameter 68-25
aggregate-param parameter B-8
aggregate-radar-jammer system 22-18
aggregateSpatialModelTickAsFastAsPossible-WhilePaused parameter A-3
aggregateSpatialModelTickPeriod parameter A-3
aggregateSpatialModelTickPeriodUsesRealTime parameter A-3
aggregateSpatialModelTickVariance parameter A-3
aggregation
  automatic 23-12
  manual 23-12
  state 23-6
agility, terrain 1-13
aiming, sensor 30-38
Aiming dialog box 30-7
Aiming set data request 30-7

air corridor 35-4
air defense 27-7
aircraft
  attacking with gun 26-6
  See also Fixed-Wing entity and Rotary-Wing entity
  setting speed 30-29
  strafing target 26-58
  taxi and launch 26-60
  taxiing on carrier 26-61
airspace 35-4
Air-to-Ground Attack dialog box 27-4
Air-to-Ground Attack task 27-4
alias, attribute transformation 55-4
all-tasks-exclusive parameter 70-14
alpha 59-22
alpha bits 5-3
Altitude
  condition 31-10
  dialog box 30-7
  set data request 30-7
altitude 18-11, 29-18
  entity, displaying 18-24
  fixed-wing entity 25-22
  flying to 26-28, 26-30
  intervisibility line 41-11
  object, changing 19-5, 35-13
  observer, filtering simulation objects by 6-9
  ordering entity to 26-38
  pasted object 16-24
  route, setting 16-16
  setting 16-14, 16-16, 30-7
    fixed-wing entity 25-24
    in dialog box 16-15
    manually 16-15
  specifying measurement unit 74-9
  switching from 2D to 3D 43-2
  testing entity for 31-10
  vertex
    changing 35-14
    displaying 33-9
altitude point
  changing color 48-6
  deleting 48-5
  inserting 48-5
Altitude set data request 25-24
Always Join Session with Open Database 4-18
Always Join Session with Session Database 4-18
ambient
  color 38-4
ambient (continued)
  occlusion map 1-16, 50-20
  value, material 38-4
ammo parameter 70-3
ammo select table 70-3
  counter measure 9-19
ammunition 22-14
  amount 30-8
  pacing and tracking 30-8
  specifying amount 30-34
  status 18-4
  unit 66-5, 66-8
Ammunition dialog box 30-8
Ammunition Pacing/Tracking dialog box 30-8
Ammunition Pacing/Tracking set data request 30-8
anaglyphic stereo 73-21
  configuring 73-22
angle, damage 70-6
Angle of Attack 59-12
angle-of-incidence parameter 70-6
angleSegmentMax parameter 60-5
animated movement
  configuring 64-26
  movement file, configuring 64-30
Animated Movement dialog box 26-5, 64-30
Animated Movement task 26-5
animation
  attack 37-10
  DI-Guy 5-8, 26-14
    adding 67-11
    choosing 58-6, 67-13
  entity 26-5
  model 61-11
  observer view transition 44-30
  testing DI-Guy 31-11
Answer Questions button 29-22
anti-air
  missile 27-5
  resource 22-14
  weapon 27-7
anti-aliasing 5-3, 6-13
anti-personnel 22-14
anti-ship
  missile 27-5
  resource 22-14
Anti-ship (Fixed Depth) task 26-34
anti-submarine, missile 26-32
anti-tank
  ditch, improving 27-23

anti-tank (continued)
resource 22-14
anti-tank ditch, digging 27-12
Any, as condition parameter 32-7
Any Subordinate of Selected Aggregate check box 32-8
AOA updater 59-12
appData directory 5-8
appearance
bit 38-6
DI-Guy 5-8
random 64-15
pasting 16-23
power plant 9-17
setting 30-9
specifying for DI-Guy model 64-14
testing DI-Guy 31-11
Appearance dialog box 30-9
Appearance set data request 30-9
appending, saved views to view list 44-32
appIdRange parameter A-7
application
data, specifying location of 5-3, 63-4
ID 3-4
number 5-6, 5-7, 5-8, 7-42, 12-9
configuring A-3
default 5-14
specifying at startup 5-14
remote, notifying of VR-Forces run and pause 7-42
Application Settings dialog box 7-22, 23-11
File Caching Settings page 50-31, 50-33, 58-7
GUI Layout page 74-4
Key Mapping Editor page 76-4, 76-9
Mouse Mappings page 44-14
Performance Options page 6-14, 6-17
Scripted Task Options page 28-36
Session Options page 4-8, 7-10, 7-11, 7-12, 7-22, 7-42, 44-21
Spot Report Options page 9-3, 9-5, 9-6, 9-8, 9-9, 30-40
applying, platform update 63-17
applying updated model definitions 57-12
appNumber parameter A-3
arc, segment 60-4
architecture, component 15-7
area 33-2, 33-5
creating for multiple entity creation 20-20
disaggregation 23-12, 33-2
extruded 31-11, 35-4
area (continued)
feature, list of 48-15
fill style 35-17
fortified, constructing 27-17
hazard 22-8
NBC 22-7
suppressive fire at 26-50, 26-51
terrain page-in 33-2, 51-6
areal, feature 52-11
Arm Mine at Depth task 26-6
Armed dialog box 30-9
Armed set data request 30-9
arming
IED 30-9
mine, naval 26-6
arrow
adding to line 35-17
forward and back 64-42, 65-7
freehand line 35-2
sensor contact line 37-15
articulated part 61-2, 61-7
attaching to 45-4
artillery
Fire-for-Effect task 26-25
unit 27-25
ASCII Export file 64-26, 64-28
ASE file 64-26, 64-28
ASE format 26-5
Ask to Join Session 4-18
assembly 66-5, 66-10
adding to entity 66-13
creating 66-11
editing 66-12
removing 66-13
roll up rule 66-14
rolling up 66-15
assertOnBlockingTerrainCalls parameter 47-8, A-3
assigning
echelon ID 15-6
plan to multiple simulation objects 32-22
session ID 4-9
asynchronous I/O 5-19, A-8
using for performance 6-11
attach 45-6
observer, articulated part 45-4
Attach Components to Remote Entities On parameter 7-5
attach mode 1-15, 44-3, 44-8, 45-6
2D behavior 45-14
Absolute 45-7

attach mode (continued)
Follow 45-7
Mimic 45-8
Mimic Location Track 45-9
Mimic Track 45-9
movement, attached context 44-8
prop, affect of 45-5
Space Follow 45-10
Tether 45-11
Tether Location Track 45-12
Tether Track 45-12
Track 45-13
Attach Object to Mouse check box 16-18
attached, context 44-8
Attached Near Clip Altitude, attribute 73-11
attaching observer
to object, with keyboard 45-3
to simulation object or prop 45-2
attachment
list, filtering 45-5
order 45-3
preservation of view changes 45-3
Attachment Panel 45-2
filtering attachment list 45-5
attack
air-to-ground 27-4
animation 37-10
anti-air missile 27-5
indicator 37-10
limited munition 22-7
missile 22-7
Attack Aircraft with Guns task 26-6
Attack By Fire dialog box 27-5
Attack By Fire task 27-5
attack parameter, edting 66-22
Attack with Anti-Air Missile dialog box 27-5
Attack with Anti-Air Missile task 27-5
attack with anti-ship missile 27-5
Attack with Anti-Ship Missile dialog box 27-5
Attack with Anti-Ship Missile task 27-5
Attack with Land-Attack Missile dialog box 27-6
Attack with Land-Attack Missile task 27-6
Attack with Torpedo dialog box 27-6
Attack with Torpedo task 27-6
attribute
Attached Near Clip Altitude 73-11
BOYSHP 52-20
buoy 52-20
building model definition 52-20
cockpit updater 59-10
attribute (continued)
COLOUR 52-20
COLPAT 52-20
DESIGNATOR 52-22
destroyedsymbol 59-8
Dynamic Near Clip Attached Policy 73-11
Dynamic Water Visibility Enabled 73-17
Dynamic Water Visibility Maximum 73-17
earth file, differential processing 52-17
feature 55-4
Fixed Near Clip 73-11
Fixed Near Clip When Attached 73-11
image 50-8
imagesymbolmap 59-7, 59-9
itemname 75-2, 75-6
light characteristic 52-28
LITCHR 52-28
MAK_EARTH_FILE 55-4
MAK_LAYER 55-4
MAK_SOURCE_FILE 55-4
marker-scale 59-20
menuname 75-2
modeldefinitionschema 59-6
Near Far Clip Policy 73-11
OBJNAM 52-22
parent 59-6
Reduce Z Fighting Ratio 73-11
sensor 73-8
setting simulation object 30-9
SIGGRP 52-28
signal group 52-28
signal period 52-28
SIGPER 52-28
SIGSEQ 52-28
TOPSHP 52-27
transformation, alias 55-4
transforming 55-4
type 59-6
visual definition 57-17
adding 57-19
WidgetTheme 59-6
window 73-7
attrition
rate 22-3
unit 22-11
audio 42-2
adding to scenario event 8-5
Audio Settings dialog box 42-2, 42-3, 42-5, 42-6, 42-7, 42-8
Audio Settings page 42-2, 42-3

authentication, proxy 51-5
Auto Generate 50-24
Auto Launch dialog box 30-12
automatic
  aggregation 23-11
  aggregation and disaggregation 23-12
  attachment 35-11
  counter measures 9-19
  embarkation 19-10
  laser 30-25
  reorganization 30-33
  running of scenarios 7-37
Automatic Air Defense task 27-7
Automatically Join Session 4-17
Automatically Open Session Database without Joining Session 4-17
autonomous lasing 9-16
auto-reorganize parameter 12-6
Avoid Roads 30-28
avoidance, collision 3-16, 30-11
avoiding
  collisions, obstacles, and features 19-17
  objects 70-15

B

-B command-line option 5-8, 5-13, 7-40
back arrow 64-42, 65-7
back-end 3-2
  balancing load 7-19
  configuring feature data 50-11
  console A-4
  dropping from exercise 3-7
  enabling time management 4-20, 5-18
  feature representation 47-5
  heartbeat A-6
  late joining 3-7
  mapping objects to 12-9
  missing 3-8
  multiple 3-2, 3-7, 12-9
  reloading OPD A-4
  remapping 3-8
    in batch mode 7-40
  sending messages 3-3
  starting 4-3
    as daemon 5-8
  synchronizing 4-18
background
  2D label 18-17
  intervisibility information 41-11
background process 27-29, 28-5, 29-29
balancing, object load 7-19
ballistic gun, controller descriptor 70-3
ballistic missile 10-8
  firing 10-8
  movement, configuring 64-27
  movement file, configuring 64-30
  scripted movement 64-26
  target event 10-10
Ballistic Missiles dialog box 64-27, 64-30
barbed wire, constructing 27-12
barricade, constructing 27-13
batch file 7-37
  command-line option 5-8
  configuring A-3
  editing 7-38
batch mode 7-36, 7-37
  remapping 7-40
  starting from command line 7-40
batchScenarioFileName parameter A-3
beacon 52-28
  model definition 52-20
  seasonal 52-26
  streaming 52-19, 52-24
  topmark 52-27
  wake 11-18
bead
  DOF 61-7
  switch 61-8
beam 30-33
  emitter 9-17, 30-17
  standard and tracking 30-17
bearing, intervisibility line 41-11
behavior
  scripted 26-5
  simulation object 15-9
Behavior Set 28-9
  assigning to force 25-20
  creating 25-19
  scripted task availability 28-18, 28-20
berm, constructing 27-13
best practice, model design 59-22
binary key mapping 76-4
binding, system definition file variable 68-21
Biological Attack dialog box 27-7
Biological Attack task 27-7
biological contamination 22-7, 22-8, 27-7
bit
  depth 5-3
  depth and stencil 6-13

bitmap 47-4
  adding new 59-9
  adding to scenario event 8-5
  displaying 50-7
  icon 59-4
  location of 2-4
blend image attribute 50-8
blur 40-2, 40-4
BMP 47-4
body coordinates 44-3
bomb
  dropping 26-53
  laser guided 26-53
  target 26-21
Bomb Location dialog box 27-8
Bomb Location task 27-8
booby trap
  creating 27-14
  destroying 27-20
  improving 27-22
boolean, operator 31-14
border, icon 18-21
Boundary Generation Tool 52-13
  running 52-15
  using 52-14
boundary.txt 52-14
bounding box
  selecting objects 17-5
  unit 65-11
bounding volume 19-17, 64-21
  simulation objects 18-31, 18-32
  specifying 64-9
  unit, color 24-11
box
  drawing 35-3
  extruded 35-4
  resizing 35-15
BOYSHP, attribute 52-20
breach, improving 27-23
Breach Obstacles dialog box 27-9
Breach Obstacles task 27-9
breaching
  fortification 27-9
  obstacle 27-9
breaking, wave 11-19
bridge 16-25, 64-8
  building 27-15
  destroying 27-20
  improving 27-23
brightness 40-4
broadcasting
  state update 43-11
  stealth position and orientation 43-12
buffer, size 5-7, 5-12
building 16-25, 64-8
  avoiding 19-17
  bridge 27-15
  conditional statement 32-4
  creating entity in 19-4
  ditch 27-16
  dry gap 27-16
  extruded 52-12
  fortified area 27-17
  prop type 50-30
  setting avoidance 30-11
  terrain 50-5
bump map 1-16, 50-20
buoy
  attributes 52-20
  daymark 52-27
  designator 52-22
  model definition 52-19, 52-20
    using texture atlas 52-23
  seasonal 52-26
  signal sequence 52-28
  streaming 52-19
  topmark 52-27
  unique 52-22
  wake 11-18
buoyancy
  enabling or disabling 39-6
  LOD distance 39-6
burst
  length 20-8, 20-12
  period 20-8, 20-12
  spawn 20-8
  spawn pattern 20-12
button
  Answer Questions 29-22
  common for model definitions 74-7
  Joystick 72-4
  Key Mapping 72-6
  toolbars 4-12

C

-C command-line option 5-3, 5-13
-c command-line option 5-8, 54-3
C++
  scripted task 28-23

C++ (continued)
task 28-3, 29-15
cache
clearing 50-33
clearing model instancing 58-7
model instance 58-7
caching
earth file, offline 50-32, 52-16
file 50-31
terrain data 1-19
terrain server data 50-32
calculating, support point 64-11
camera
effects 40-2
jitter 44-25
moving 44-13
position and orientation 73-20
shaking 44-25
ways to move 44-12
Camera Orientation Offset 73-8
Camera Position Offset 73-8, 73-20
can-be-radar-jammed parameter 22-18
canceling, reactive task 25-17
cancelling, generation of navigation data 53-8
can-pivot parameter 26-61
Capabilities dialog box 30-10
Capabilities set data request 30-10
capability, setting 30-10
car bomb, detonating 30-14
carrier
placing entity on 19-4
taxiing on deck 26-60, 26-61
Cartesian, coordinate system 29-18, 29-20
CAS 26-21
strafing target 26-58
catastrophic-kill parameter 70-6
Categories list 64-8
category
action 28-8, 28-17
adding 64-48
changing 64-3
echelon 15-5
object, changing 64-3
Object Group 16-25
Props Palette, adding 59-13
removing 64-48, 64-49
renaming 64-48, 64-49
undoing changes 64-49
weapon and ammunition, adding 66-26
CDB database 52-16
CEO. See combat engineering object
certainty level for spot reports 9-8
cgfDispatcherReceivePort parameter A-3
chaff 26-33
launching 9-19
Change Hostility dialog box 9-12
Change MOPP Level dialog box 27-9
Change MOPP Level task 27-9, 30-27
Change Posture dialog box 27-10
Change Posture task 27-10, 30-30
changing
2D icon 64-18
2D icon image 64-19
3D model 64-17
altitude, vertex 35-14
channel attributes 73-8
color
altitude point 48-6
force 19-27
tactical graphics 35-16
colorized model 64-17
component priority 69-9
control object 35-12
entity
category 64-3
enumeration 64-4
parameter 64-11
entity name 19-5
extended label index 18-20
field of view 73-15
force hostility 9-10
frame rate mode 12-6
gain 44-28
grid spacing 65-14
icon size 18-22
image display order 50-10
intervisibility object 41-8
key mapping 76-7
line, direction 35-16
model definition, prop 56-24
model set 43-11
MOPP level 27-9
name, tactical graphic 19-5
object category 64-3
observer mode 43-3
overlay name 34-5
posture 27-10, 30-30
prop
model definition 50-30
position or orientation 50-30

changing (continued)
prop
type 50-30
simulation object, name 64-6
site ID 12-9
subordinate order 65-6
terrain database for scenario 12-8
unit of measurement 74-9
unit state 23-11
runtime 23-10
viewport 73-16
width of line 35-17
window attributes 73-7
channel
adding 73-5
attribute, sensor 73-8
changing attributes 73-8
keyword 73-8
compass 44-5
multiple 73-18
observer 43-6
projection resize policy 73-12
property 73-8
removing 73-6
sound mapping 42-8
Channel Name 73-8
ChannelKeyword parameter 59-12
character
DI-Guy 5-8
adding 67-10
specifying for DI-Guy model 64-14
character_animations_table.mtl 67-10
character_digest.xml 67-10
check box, Marker Labels 64-24
checking
version, plug-in 5-3
checking waypoint accessibility 20-18
checkpoint 7-27, 7-29
deleting 7-33
disabling periodic 7-33
periodic, scheduling 7-32
rolling back to 7-25
saving 7-30
saving from snapshot 7-31
Checkpoint Settings page 7-26, 7-32, 7-33, 7-34
Chemical Attack dialog box 27-10
Chemical Attack task 27-10
chemical contamination 22-7, 22-8, 27-10
child, element definition 57-14
chop, wave 11-15
choppiness, ocean 11-17
CID level 9-13
for firing at target 9-15
icon, color 9-13
circle, extruded 35-4
circling, point 26-45
Civilian Idle task 26-7
Civilian Protest task 26-7
civilian visit point 20-22
clamp image attribute 50-8
clamp to border image attribute 50-8
clamp to edge image attribute 50-8
clamping, entities 19-21
class
DtSimComponentManager 68-9
DtSimObjectNetInterface 68-9
DtSimObjectStateRepository 68-9
DtSimTaskManager 68-9
DtVrfObject 68-9
FeatureSet 29-3
Location3D 29-18
Lua 29-3
SimObject 29-4
Vector3D 29-18
VectorGeoc3D 29-20
VectorOffset3D 29-20
vrf 29-4
VR-Forces 68-3
Classified CID level 9-13
clearing
file cache 50-33
model instancing cache 58-7
object console 18-36
snapshot list 7-35
view list 44-31
click to create 16-11
click to locate 16-11, 16-13
climb rate 26-30
clipboard 18-35
clipping plane 59-18
adjusting when creating simulation objects 19-4
clock mode 3-11, 12-6
close air support 26-21, 27-4
strafing 26-58
closing
all windows 4-25
application 4-25
Create Object dialog box 16-13
formation 21-5
simulation 7-27

closing (continued)
simulation engine 4-25
cloud
cover 11-5, 11-8, 11-11
disabling 38-13
skybox cube map 38-12
smoke 26-33
cockpit display 46-6, 58-2, 59-10
adding new 59-10
element definition 57-14
enabling or disabling 46-4
installing 59-11
model definition, creating 59-11
updater 59-10
attribute 59-10
Cockpit Display Settings page 46-5
code
country A-3
laser 30-26
assigning 9-15
coefficient probability 70-4
Collada 47-3
collapsing
unit 24-2, 24-3
to root 24-3
collateral damage 9-22
probability table 70-6
collision avoidance 3-16, 19-17
configuring 70-15
specifying for entity 30-11
Collision Avoidance Types dialog box 30-11
Collision Avoidance Types set data request 30-11
color
2D label background 18-17
altitude point
changing 48-6
deleting 48-5
inserting 48-5
ambient 38-4
compass 44-5
diffuse 38-4
emitter volume, pulse rate 60-2
entity label 18-10
fog 11-5, 11-10
force, changing 19-27
intervisibility, configuring 41-9
partially identified simulation objects 9-13
pattern 52-20
radio communication line 37-22
represents emitter volume frequency 60-2
color (continued)
sensor contact line 37-15
specular 38-4
squawk indicator 37-22
tactical graphics, changing 35-16
terrain, configuring 48-5
unit bounding volume 24-11
coloring, terrain 48-4
colorized model, changing 64-17
COLOUR attribute 52-20
COLPAT attribute 52-20
combat 22-2
aggregate-level 22-7
power 22-3
unit 23-10
graphics 37-10
combat engineering object 22-9
as feature 55-10
completion percentage, setting 30-29
configuring 66-25
creating 22-12
creating new model 66-3
partial completion 22-12
rate of creation 22-12
status 22-12
combat identification level 9-13
combined mode, vrfLauncher 5-14
combining, key mapping 76-8
Come to Stop task 26-7
command
Abandon Plan 32-24
Delete Self 20-9
Disembark 19-11
Embark On 19-9
global, adding to plan 32-15
Issue Plan 32-18
menu, configuring 75-2
Restart Plan 32-23
command line
loading scenario from 7-18
loading terrain 5-5
options 5-3
command-line, Simulation Object Editor 63-4
command-line option 5-1
-- 5-3, 5-11, 5-13, 54-3
-A 5-7, 5-12, 5-19, 54-3
-a 5-6, 5-7, 5-8, 5-14
--additionalSystemScriptsDirectory 5-8
--alphaBits 5-3, 6-13
--antiAliasing 5-3, 6-13

command-line option (continued)
--appDataDir 5-3, 5-8
--appIdRange 5-8
--appNumber 5-6, 5-7, 5-8
--autoConnect 5-3
-B 5-8, 5-13
B-- 5-13, B-10
--backend 5-13
--batchedRendering 5-4
-C 5-3, 5-13
-c 5-8, 54-3
--col 54-3
--config 5-13
--countryCodesMappingFile 5-8
--daemon 5-8
--dataDir 5-3, 5-8
-debug 5-13, B-10
--defaultObjectTimeout 5-7, 5-8
--defaultSimulationModelSet 5-3
--depthBits 6-13
--depthBuffer 5-3
--destAddrString 5-7, 5-12
--deviceAddress 5-7, 5-12
--diGuyAnimationsFile 5-8
--diGuyCharacerDataFile 5-8
--dis 5-3
--disableCallbackQueue 5-8
--disableKDTrees 5-4
--disableParallelTick 5-8
--disPort 5-7, 5-12
--dispSetting 5-3
--disVersion 5-7, 5-12, 5-20
--doNotCheckPluginVersions 5-3
--doNotLoadVrfPlugins 5-3, 5-8
-E 5-3
-e 54-3
--enableChannel 5-9
--encryptImages 54-3
--entDispSetting 5-3
--entTypeMap 5-3
--envSetting 5-3
--execname 5-7, 5-11
--exerciseId 5-7, 5-12
--extendLevelZero 54-3
-F 5-4, 5-13
F-- 5-13, B-10
-f 5-6, 5-11, 5-17, 54-3
--factoryRootDir 5-3
--federateName 5-6, 5-11
--federateType 5-7, 5-11

command-line option (continued)
--fedFileName 5-6, 5-11
--file 54-3
--fileNotifyLevel 5-9
--fileTransporterReceivePort 5-4
--fomMapperInitData 5-6, 5-11
--fomMapperLib 5-6, 5-11
--fomModules 5-6, 5-11
--fowPerspective 5-4
--frameRate 5-4
--frontend 5-13
--frontEndPID 5-8
--fullScreen 5-4, 74-6
--fullyLoadTerrain 5-9
-G 5-4, 5-13, 5-15
-g 54-4
--generateFiles 54-4
--gui 5-4, 75-2
-H 5-6, 5-7
-h 5-4, 5-9, 5-13, 54-4
--help 5-4, 54-4
--highestPriority 5-4
--hla13 5-4
--hla1516 5-4
--hla1516e 5-4
--hostAddressString 5-6, 5-7, 5-9
-I 5-4
-i 4-9, 5-6, 5-7, 5-9
--ignore_rest 5-3, 5-11, 54-3
--ignoreZoomForSpeedtreeLod 5-6
-K 5-4
-L 5-4, 5-9
-l 5-4, 54-4
--layoutSettingsFile 5-4
--level 54-4
--loadObservers 5-4
--loadPlugin 5-5, 5-9, 5-16
--loadSimulationObjectGroup 5-5
--locale 5-4, 5-13
--localSettingsDesignator 5-6, 5-11
--logFileName 5-4, 5-9
-m 54-4
--mcastTtl 5-7, 5-12
--mimModule 5-11
--missing 54-4
--msdlSIDCMappingFile 5-9
--multicastAddresses 5-7, 5-12
-N 5-6, 5-11
-n 5-5, 5-9, 5-15
--nonVrfObjectsUUIDScheme 5-5, 5-9

command-line option (continued)
--noRtiCompilerCheck 5-9
--notifyLevel 5-5, 5-9
--noVSync 6-19
--numCallbackThreads 5-9
-O 5-5
--outputLogFile 5-9
-P 5-7, 5-12, 5-19
-p 5-7, 5-11, 54-4
--plugin 5-5
--pseudoFullScreen 5-5, 74-6
--publishMftOnly 54-4
--pvd 5-5
-q 5-10
-r 5-10, 54-4
--recvBufferSize 5-7, 5-12
--regenerateAll 54-3
--row 54-5
--rprFomRevision 5-7, 5-11
--rprFomVersion 5-7, 5-11
-S 5-5, 5-6, 5-11, 5-12, 5-19, 45-6
-s 5-10, 5-14, 54-4
--scenarioFile 5-5
--searchDir 54-4
--sendBufferSize 5-7, 5-12
--sessionId 5-6, 5-7
--setMainThreadToHighPriority 5-10
--settingsFile 5-8
--showConsole 5-3
Simulation Object Editor
-- 63-4
--appDataDir 63-4
--dataDir 63-4
--doNotCheckPluginVersions 63-4
--doNotLoadVrfPlugins 63-4
-G 63-4
-h 63-4
--help 63-4
--ignore_rest 63-4
--launcherLocation 63-4
--layoutSettingsFile 63-4
--loadPlugin 63-4
--locale 63-4
--opdEditorLocation 63-4
--settingsDirectory 63-4
--showConsole 63-4
--smsFile 63-4
-v 63-4
--version 63-4
--visualTerrain 63-4
command-line option (continued)
--simulationModelSet 5-10
--siteId 5-7
--startMinimized 5-10
--stealth 5-5
--stencilBits 5-5, 6-13
--subDir 54-4
--subnetMask 5-7, 5-12
--suppressHatIntersectOnAttach 5-5, 45-6
--suppressSelfReflect 5-12
--synchronizeOcean 5-5
-T 5-10
-t 54-4
--targetFrameRate 5-10, 6-17
--tcpSendTimeout 5-5
--textureSize 54-5
--threads 54-4
--timeManagement 4-20, 5-11, 5-18
-u 54-4
--unplug 5-5
--useAbsoluteTimeStamps 5-6, 5-10
--useAdvisories 5-6, 5-11
--useAsyncIO 5-7, 5-12, 5-19
--useFileCommChannel 5-6
--useGeographicDdm 5-7
--useIpv6 5-7, 5-12
--userDataDir 5-6, 5-10
-v 5-6, 5-10, 5-13, 54-4
--version 5-6, 5-10, 5-13, 54-4
--vtFile 54-4
-w 54-5
--waitQueue 5-11
-x 5-7, 5-11, 5-12, 5-16, 54-5
DIS 5-20
-Z 5-6
comma-separated values file 10-8
comment
@dis 52-25
MTL parameter A-9
comm-model-name parameter 70-13
commModelParams.mtl file 13-2, 13-3, 70-13, A-8
communication
component 15-7
line 37-19
line-of-sight 37-21
communication model 70-13, A-8
custom 15-7
external 15-6
setting 13-2
simulation object 15-6

communications, jamming 22-16, 22-17
company, pre-configured 23-5
comparison operator 31-13
compass
  color 44-5
  configuring 44-5
  location 44-5
  model definition 44-5
  observer 44-4
compiling, incremental 6-11
component 68-8
  actuator 15-8
  adding to object in OPD Editor 69-8
  architecture 15-7
  attaching to remote entities 7-5
  attaching to remote entity 70-18
  communication among 15-7
  controller 15-8
  creating connection in OPD Editor 69-10
  deleting in OPD Editor 69-7
  descriptor, elements of 68-13
  emitter 30-17
  entity 68-13
  priority 69-9
  sensor 15-8
  simulation object 68-13
  systems 68-16
  tuning 6-12
    performance 6-12
component attachment table 70-18
Component Attachment Table parameter 7-6
Component Manager B-5
Component Visualization 50-22
Component-Attachment parameter 12-5
Component-Attachment-Table parameter 12-5
componentAttachmentTable.mtl 70-18
component-descriptor-type 68-13
component-manager parameter B-5, B-7
component-type 68-13
composing, terrain 47-2
compressing, model and image files 59-23
compression, texture 50-33
computed route, creating 35-7
Concealed dialog box 30-12
Concealed set data request 30-12
concealment, setting 30-12
concurrent task 24-1, 25-4, 28-17
condition
  considerations for using 31-10
  Detect Entity 31-10
condition (continued)
  Engineering Object Breached 31-10
  Entity Altitude 31-10
  Entity Destroyed 31-10
  Entity Di-Guy Animation Check 31-11
  Entity Di-Guy Appearance Check 31-11
  Entity Embarked 31-11
  Entity Has Target 31-11
  Entity In Area 31-11, 35-4
  Entity Left of Line 31-12
  Entity Under Fire 31-12
  Lifeform Surrendered 31-12
  Missile Approach Warning 31-12
  name 32-8
  name condition, pattern 32-7
  pattern 32-9
    editing CSV file 32-9
  Random 31-13
  Receive Text Message 31-13
  Resource 31-13
  Scenario Event 31-13
  Simulation Date/Time 31-14
  Simulation Time 31-13
  Tactical Graphic Active 30-5, 31-14, 35-18
  While 31-9
conditional expression dialog box 32-4
conditional statement 31-4
  adding to plan 32-4
  building 32-4
  filtering simulation object list 25-9
  prefix notation 32-5
  trigger 31-7
*condition-entity-types.csv*file 32-9
configuration
  display engine 73-3
    loading 73-6
    saving 73-6
  entity display 5-3
  environment 5-3
  file
    condition-entity-types.csv 32-9
    RTI 2-10
  joystick, favorite 72-11
  plug-in, deleting 4-38
  simulation model set 7-9
configuration file
  character_animations_table.mtl 67-10
  character_digest.xml 67-10
  commModelParams.mtl 13-2, 13-3, 70-13, A-8
  feature 55-6

configuration file (continued)
for object parameter database 63-5
navigation area 53-10
sysdef 68-16
vrfSim.mtl A-3
vrfSimSettings.xml 6-14
configuring
2D icon 59-2
aggregate-level entity 66-3
back-end
console A-4
heartbeat A-6
ballistic missile movement 64-27
batch file A-3
combat engineering objects 66-25
communication model A-8
compass 44-5
condition pattern enumerations 32-9
context-sensitive menu 75-5
contour lines 48-10
counter measures 9-19
Create Object dialog box 16-13
depth of field 44-24
destination address A-7
detection tables 9-13
dialog box
information 75-3
pages 75-3
DI-Guy A-4, A-6
character and appearance 64-14, 67-10
DIS A-7
display 5-3
drainInput A-5
for DIS A-7
for HLA A-6
earth file 52-3
embarkation 64-33
slot 64-35, 64-38
embedded entity 19-12
emitter 70-11
emitter volume 60-2
segments 60-4
exercise ID A-7
external communication model 13-3
feature avoidance 70-16
feature data 55-2
FED file name A-6
federation execution A-6
name A-6
FOM Mapper A-6
configuring (continued)
formations 70-8
frequency of data sends A-4
GDB soil type 48-2
grid lines 48-8
ground clamping 19-23
GUI elements 75-2
heartbeat threshold A-7
HLA A-6
hostility 9-10
IED 30-14
ingress and egress points 64-35
instrument panel 46-7
interest management 6-9
intervisibility objects 41-9
joystick 72-2
keyboard entity control 72-2
map datum A-6
menu 75-2
model definition 57-8
movement file 64-30
multicast address A-7
multichannel display 73-18
mutually exclusive tasks 70-14
object avoidance 70-15
obstruction avoidance 70-15
ocean planar reflection 38-8
orientation clamping 19-23
path, to OPD file A-4
port number A-7
projection resize policy 73-12
props 47-4
pulse rate 60-2
radio message model A-8
radios 70-13
range rings 18-28
receive port A-3
right-click menu 75-5
rotary-wing lateral movement 25-31
scenario file to use A-5
schema 57-3
sensor contact line 37-15
sensor effects 40-4
sensors 71-2
session 4-17
ID A-5
shadows 38-16
ship wakes 59-14
signature propagator 71-9
simulation management message A-5

configuring (continued)
site ID A-5
slot exclusion 64-40
start mode A-6
stereo
anaglyphic 73-22
polarized 73-23
system for MÄK RTI 2-10
terrain color settings 48-5
terrain database
path A-4
to load A-6
terrain profile 49-3
time management 4-19
time stamp order A-7
toolbar 75-6
track history 37-6
unit, ammunition, weapons, equipment 66-5
visual quality slider 6-15
VR-Forces for external communications effects server 13-2
confirmation prompt
plan, enabling or disabling 25-5
task, enabling or disabling 25-5
connecting, to network automatically 5-3
connection
component 69-10
configuring in system definition file 68-18
information 4-32
sensor 71-3
simulation object group 64-42
Connection Information dialog box 4-32
console
configuring for back-end A-4
messages
sending 32-16
sent to 5-15
object 18-32
setting notification level 30-28
constrained, time 4-18
constraint 45-6
view 44-27
Construct Abatis dialog box 27-11
Construct Abatis task 27-11
Construct Anti-Tank Ditch dialog box 27-12
Construct Anti-Tank Ditch task 27-12
Construct Barbed Wire dialog box 27-12
Construct Barbed Wire task 27-12
Construct Barricade dialog box 27-13
Construct Barricade task 27-13
Construct Berm dialog box 27-13
Construct Berm task 27-13
Construct Booby Trap dialog box 27-14
Construct Booby Trap task 27-14
Construct Bridge dialog box 27-15
Construct Bridge task 27-15
Construct Dry Gap dialog box 27-16
Construct Dry Gap task 27-16
Construct Fortified Area dialog box 27-17
Construct Fortified Area task 27-17
Construct Fortified Line dialog box 27-18
Construct Fortified Line task 27-18
Construct Minefield dialog box 27-18
Construct Minefield task 27-18
Construct Road task 27-27
Construct Strong Point dialog box 27-19
Construct Strong Point task 27-19
constructing
abatis 27-11
barbed wire 27-12
barricade 27-13
berm 27-13
fortified line 27-18
strong point 27-19
terrain 50-5
contamination
agent 22-7
biological 27-7
chemical 27-10
nuclear 27-27
nuclear, biological, chemical 22-7, 22-8
type, setting 30-12
Contamination dialog box 30-12
Contamination set data request 30-12
context-sensitive menu 74-6
configuring 75-2, 75-5
contour line
displaying 48-9
spacing 48-10
thickness 48-10
Contour Lines page 48-10
contrail 37-3
contrast 40-4
control, remote 1-16
control object 33-2, B-7
concepts 33-2
copying 16-21
count of 18-37
creating 16-3
editing 35-12

control object (continued)
list of for scenario 12-10
name 33-2
parameter-type 68-26
pasting 16-21
pedestrian path 20-22, 33-2
selecting 17-4, 17-5
spot report 9-9
controller 15-7
component 15-8
game 19-19
joystick 72-2
making decisions and performing tasks 15-8
controlling
messages 5-15
remote 44-32
conventions, manual liii
convoy 26-8, 26-9
behavior 25-10
Convoy Along task 25-10, 26-8
Convoy To task 25-10, 26-9
coordinate
observer 44-3
point 48-17
UV 50-9
coordinate system 1-13
3D 44-7
3D Cartesian 29-18
Cartesian 29-18, 29-20
changing unit of 74-9
geocentric 47-6
supported 47-6
copying
formation 65-12
model definition 57-11
objects 16-21
performance and 16-21
plan statements 32-23
schema 57-5
scripted task 28-32
spawn pattern 20-14
corridor 35-4
count
object 18-37
simulation object 18-37
counter measure 26-33
ammo select table 9-19
launching 9-19
Counter Measures Auto Launch, set data request 9-19
Counter Measures Auto Launch set data request 30-12
country code A-3
countryCodesMappingFile parameter A-3, A-5
CPU time, giving up 6-17
crawling 30-31
Create Anti-Tank Ditch dialog box 27-12
Create Barbed Wire dialog box 27-12
Create Barricade dialog box 27-13
Create Berm dialog box 27-13
Create Booby Trap dialog box 27-14
Create Bridge dialog box 27-15
Create Connection dialog box 69-10
Create Dry Gap dialog box 27-16
Create Fortified Area dialog box 27-17
Create Fortified Line dialog box 27-18
Create global command 32-17
Create Indirect Fire dialog box 10-3
Create Intelligence Object dialog box 8-8
Create menu 16-3
assigning or removing simulation object 64-6
selecting object 16-8
Create Minefield (ADAM-RAAM) dialog box 27-22
Create Minefield (Volcano) dialog box 27-18
Create Multiple Entities, dialog box 20-19
Create New Component dialog box 69-8
Create New Entity dialog box 64-31
Create New Entity Type Model Mapping dialog box 58-3
Create Object
dialog box 16-3
Attach Mouse to Object 16-18
configuring 16-13
tab 16-11
Create Object dialog box 16-13
Create Resource dialog box 69-11
Create Route dialog box 16-16
Create Scenario Event dialog box 8-3
Create Spawn Pattern dialog box 20-10, 20-14
Create Strong Point dialog box 27-19
create-script-id parameter 28-20
creating 16-9
aggregate-level entity 66-3
area for multiple entities 20-20
assembly 66-11
batch file 7-37
Behavior Set 25-19
booby trap 27-14
combat engineering object 22-12

creating (continued)
component connection 69-10
computed route 35-7
control object 16-3
crowds 20-16
cultural feature 16-25, 64-8
demonstration recordings 32-19
element definition 57-15, 67-6
embedded entity 19-15
entity 16-9, 19-3
adjusting clipping plane 19-4
fixed-wing 25-22
from existing 64-32
in buildings 19-4
new 64-31
fortified area 27-17
fortified line 27-18
freehand line 35-2
global plan 32-11
indirect fire event 10-3
inset view 46-2
joystick configuration 72-3
key map 76-9
MEDF and MEIF files 59-23
minefield 27-18
model, guidelines for 59-22
model definition 57-8, 57-9, 62-2, 67-3
cockpit display 59-11
navigation area 53-3
new scenario 7-3
object 16-3, 16-4, 16-5, 16-8, 16-11
in plan 32-17
mouse with 16-18
setting properties 16-13
objects 16-8
observer mode 43-4
overlay 34-3
plans 32-2
prop 16-9, 16-26, 59-13
reactive task 28-24
scenario event 8-3
schema 57-4
scripted task 28-6
selection group 17-9
selection group from menu 17-9
simulation model set 63-16, 63-17
simulation object 16-3, 16-4, 16-8, 16-11, 19-3
from existing 64-32
group 64-41
multiple 20-17
creating (continued)
simulation objects, multiple 20-16
sink points 20-4
snapshot 7-34
spawn pattern 20-10
spawn points 20-2
strong point 27-19
system definition file 68-17
tactical graphics 16-9, 35-2
terrain 50-3
text object 35-4
unit 23-2, 65-4
aggregation state 23-6
entity 23-3
user-defined formation 70-10
crepuscular ray 38-11
crouching 30-31
crowd
creating 20-16
dispersing 26-13
gathering around object 26-10
gathering at location 26-10
protesting 26-49
wander 26-64
Crowd Along Line dialog box 26-9
Crowd Along Line task 26-9
Crowd Around Location task 26-10
Crowd Around Object task 26-10
Crowd in Front of Entity dialog box 26-11
Crowd in Front of Entity task 26-11
cruise missile, firing 26-24
CSV, file 26-5
CSV file 10-8, 64-26, 64-28
configuring for ballistic missile or animated movement 64-30
CTDB file 47-3
cube map, recalculating 38-12
cultural feature 16-25, 64-8
creating 16-3, 16-4, 16-5, 16-8, 16-11
heading, setting 16-17
information 18-3
cultural-feature-param parameter 68-12, B-8
cultural-feature-state-repository 68-12
Current Speed dialog box 30-13
Current Speed set data request 30-13
cursor 16-9, 16-10
curvature of the earth
display, mode 37-21
custom, plan 20-2, 20-9, 20-12
customizing, object parameter database 68-27

cut-in site 52-13

D

-d command-line option, vrfSim 5-8
DAE file 47-3
daemon
  running vrfSim as A-3
  starting back-end as 5-8
daemon parameter A-3
damage
  actuator 9-22, 70-6
  collateral 9-22
  configuring entity to take 70-7
  file 9-22, 68-26
  model 64-22
    specifying 64-24
  probability table 70-6
  system 9-22
dashed line 35-17
data
  caching terrain 50-32
  configuring frequency of sends A-4
  directory 2-4
    specifying location of 5-3, 63-4
  elevation 47-3
  feature, configuring 55-2
  input 68-13
  path 3-12
  port 15-7
  S-57 50-19
  streamed, processing 52-6
  synchronizing 4-25
data directory
  back-end 5-8
    user 5-10
data distribution management 6-8, 6-9
  See also DDM
database 47-6
  CDB 52-16
  ground-db 48-18
  large 47-7
  LittlePond 48-18
  loading 47-7
  Local World Procedural Terrain 48-18
  Makland 48-18
  object parameter 68-2
  paging 47-8
  SanLuisObispo 48-19
database (continued)
  terrain 1-13
    loading 4-21
  VR-TheWorldOnline 48-19
  VR-Village 48-19
dataset, virtual and tiled 54-3
date 11-2
  setting 11-2, 11-3
  setting in new scenario 7-7
  simulation 7-12
    default 7-12
Date and Time page 11-3, 11-4
datumShiftX parameter A-4
datumShiftY parameter A-4
datumShiftZ parameter A-4
daylight-illumination parameter 71-13
daymark, buoy 52-27
daymark_attr_map.csv 52-27
daymark_layer_list.csv 52-27
daymark.dtxc 52-27
day-night-transition-time is parameter 71-13
DDM 6-8, 6-9
  region size 6-9
DDS texture 50-33, 50-34
  flipping 50-6, 59-17
Deactivate Jammer task 27-19
deactivating, jammer 27-19
dead-reckoning 6-16, A-5
debug mode, running from vrfLauncher 5-14
debugging, shaders 50-22
decal effects 37-3, 58-2
  enabling and disabling 37-4
  models 37-4
decal image attribute 50-8
decay rate for spot reports 9-8
decision making 15-8
decreasing speed of navigation 44-27
default
  formation 65-10
  joystick configuration 72-11
  name 16-11
  plan 20-2, 20-9, 20-12
  Set commands 30-4
  settings, restoring 4-23
  simulation model set 5-3
    configuration 7-9
  simulation model set configuration 7-11
  spawn pattern 20-9
  starting date and time 7-12
  value, parameter 12-10

default (continued)
view 44-31
default overlay
adding 64-7
specifying 64-7
default_guiConfiguration.gui 75-2
default_guiSimpleScenarioPlayer.gui 75-2
DefaultEllipsoidArc, model definition 60-2
defaultHeartbeatThreshold parameter A-7
defaultObjectTimeout parameter A-7
defaultParameterDatabasePath parameter 12-7, A-4
defaultSpawnTemplates.mtl 20-9
default-task-rules.tsk 28-17, 70-14
defaultTerrainDatabasePath parameter 12-7, A-4
defense parameter, editing 66-23
defensive
measure 26-33
movement 27-26
definition
element 57-14
model, buoy 52-19, 52-20
parameter 69-4
degrees of freedom 61-7
Delete button 44-31
Delete Checkpoints dialog box 7-33
Delete Object global command 32-17
Delete Self, command 20-9
Delete Self global command 32-17
deleting
altitude point 48-5
assembly 66-13
category 64-48, 64-49
channel 73-6
checkpoints 7-33
element definition 57-20
entity 19-5
from simulation model set 64-42
environment condition 11-8
extended label 18-20
intervisibility object 41-13
key map 76-9
key mapping 76-7
messages from object console 18-36
model definition 57-10
model mapping 58-6
object 64-42
in plan 32-17
object from selection group 17-11
observer 43-9
mode 43-5
deleting (continued)
overlay 34-5
object 35-11
parameter
model definition 57-14
schema 57-7
parameters and components in OPD Editor 69-7
plan statement 32-10
plug-in
configuration 4-38
XML file 4-39
scenario event 8-13
schema 57-5
scripted task 28-32
selection group 17-11
simulation object 19-5
in plan 32-17
snapshots 7-35
spawn pattern 20-16
terrain, cached 50-33
trigger 31-8
unit 23-13
vertex 35-15
view 44-31
window 73-6
Deliberate-Attack, posture 22-4
Deliberate-Defense, posture 22-4
demonstration
creating 32-19
mode 74-6
Deploy Sonobuoy task 26-11
Deploy Sonobuoys Along Route task 26-12
deploying
embedded entity 19-15
with task 19-15
sonobuoy 26-11, 26-12
depth
bits 5-3
buffer 6-13
focal 44-22
moving to 26-38
ocean 50-19
sonar 30-39
depth charge 26-14, 26-15, 26-23
depth of field 44-22
configuring 44-24
enabling and disabling 44-22
descent rate 26-30

description
parameter 69-4
spawn pattern 20-10
deselecting
object 17-7
simulation object 17-7
DESIGNATOR, attribute 52-22
designator 15-5
buoy 52-22
laser beam 9-15
destAddrString parameter A-7
destination address 5-7, 5-12, A-7
Destroy Bridge dialog box 27-20
Destroy Bridge task 27-20
Destroy Ditch dialog box 27-20
Destroy Ditch task 27-16, 27-20
Destroy Explosive dialog box 27-20
Destroy Explosive task 27-20
Destroy Fortification dialog box 27-21
Destroy Fortification task 27-21
Destroy Obstacle dialog box 27-21
Destroy Obstacle task 27-21
Destroy Road task 27-27
destroyed, setting simulation object to 30-13
Destroyed set data request 30-13
destroyedsymbol attribute 59-8
destroyFedExec parameter A-6
destroying
booby trap 27-20
bridge 27-20
ditch 27-20
federation execution A-6
fortification 27-21
obstacle 27-21
unexploded ordnance 27-20
destroy-script-id parameter 28-20
detaching, observer from object 45-5
Detect Entity condition 31-10
Detected CID level 9-13
detecting, targets 9-13
detection table 9-13
detection types, sensor 71-9
detonating, IED and car bomb 30-14
detonation
effects 37-8, 58-2
event
mapping sound effects to 42-4
sound effects 42-2
line 37-9
result, sound mapping 42-8
detonation (continued)
sound mapping 42-8
Detonation Fuse Type dialog box 30-14
Detonation Fuse Type set data request 30-14
Detonation Sound Editor page 42-8
Device address 5-7, 5-12
deviceAddress parameter A-7
dialog box
Active Sonar Mode 30-6
Activity 30-6
Add Conditional Expression 31-6
Add Group of Entities 20-20
Add Model Definition 57-9, 67-3
Add Terrain Patch 50-6, 56-3, 56-21
Aggregate As 65-3
Aiming 30-7
Air-to-Ground Attack 27-4
Altitude 30-7
Ammunition 30-8
Ammunition Pacing/Tracking 30-8
Animated Movement 26-5, 64-30
Appearance 30-9
Application Settings 7-22, 23-11
File Caching Settings page 50-31, 50-33, 58-7
GUI Layout page 74-4
Key Mapping Editor page 76-4, 76-9
Mouse Mappings page 44-14
Performance Options page 6-14, 6-17
Scripted Task Options page 28-36
Session Options page 4-8, 7-10, 7-11, 7-12, 7-22, 7-42, 44-21
Spot Report Options page 9-3, 9-5, 9-6, 9-8, 9-9, 30-40
Armed 30-9
Attack By Fire 27-5
Attack with Anti-Air Missile 27-5
Attack with Anti-Ship Missile 27-5
Attack with Land-Attack Missile 27-6
Attack with Torpedo 27-6
Audio Settings 42-2, 42-3, 42-5, 42-6, 42-7, 42-8
Auto Launch 30-12
Ballistic Missiles 64-27, 64-30
Biological Attack 27-7
Bomb Location 27-8
Breach Obstacles 27-9
capabilities 30-10
Change Hostility 9-12
Change MOPP Level 27-9

dialog box (continued)
Change Posture 27-10
Chemical Attack 27-10
Collision Avoidance Types 30-11
Concealed 30-12
conditional statement 32-4, 32-8
configuring 75-3
Connection Information 4-32
Construct Abatis 27-11
Construct Anti-Tank Ditch 27-12
Construct Barbed Wire 27-12
Construct Barricade 27-13
Construct Berm 27-13
Construct Booby Trap 27-14
Construct Bridge 27-15
Construct Dry Gap 27-16
Construct Fortified Area 27-17
Construct Fortified Line 27-18
Construct Minefield 27-18
Construct Strong Point 27-19
Contamination 30-12
Create Anti-Tank Ditch 27-12
Create Barbed Wire 27-12
Create Barricade 27-13
Create Berm 27-13
Create Booby Trap 27-14
Create Bridge 27-15
Create Connection 69-10
Create Dry Gap 27-16
Create Fortified Area 27-17
Create Fortified Line 27-18
Create Indirect Fire 10-3
Create Intelligence Object 8-8
Create Minefield (ADAM-RAAM) 27-22
Create Minefield (Volcano) 27-18
Create Multiple Entities 20-19
Create New Component 69-8
Create New Entity 64-31
Create New Entity Type Model Mapping 58-3
Create Object 16-3
configuring 16-13
Create Resource 69-11
Create Route 16-16
Create Scenario Event 8-3
Create Spawn Pattern 20-10, 20-14
Create Strong Point 27-19
Crowd Along Line 26-9
Crowd in Front of Entity 26-11
Current Speed 30-13
Delete Checkpoints 7-33

dialog box (continued)
Destroy Bridge 27-20
Destroy Ditch 27-20
Destroy Explosive 27-20
Destroy Fortification 27-21
Destroy Obstacle 27-21
Detonation Fuse Type 30-14
DI-Guy Appearance 30-15
DI-Guy Hand Item 30-15
Display Settings 4-23, A-2
Cockpit Display Settings page 46-5
Display Units Settings page 74-11
Entity Display Settings page 18-13, 18-14, 18-22, 18-23, 18-24, 18-36, 19-21, 19-23, 19-27, 25-5, 37-12, 37-18, 39-5, 39-8, 43-12
Grid Lines Settings page 48-8
Indirect Fire Settings page 10-7
Interest Management Settings page 6-9
Intervisibility Settings page 41-6, 41-7, 41-9, 41-10, 41-11
Lighting Render Settings page 38-4, 38-5, 38-6
Loader Settings page 50-35
Observer Settings page 18-27, 18-31, 18-32, 19-26, 19-27, 37-4, 37-5, 37-14, 38-7, 39-3, 43-4, 43-5, 43-6, 43-9, 44-5, 44-23, 44-25, 44-26, 44-28, 46-10, 48-7
Ocean Render Settings page 11-18, 38-8, 38-12, 39-4, 50-18, 50-19
Radio Communications Settings page 37-20, 37-21, 37-22
Range Rings Settings tab 18-28
Render Settings page 6-7, 6-18, 38-10, 38-11, 44-24, 48-12, 50-23, 73-13
Sensor Settings page 40-4, 40-5
Shadow Settings page 38-15, 38-16
SpeedTree Settings page 59-19
Symbol Decoration Settings page 18-15
Tactical Graphics Display Settings page 33-7, 33-8, 33-9, 36-3
Terrain Profile Settings page 41-12
Track History Settings page 37-7
Unit Display Settings page 24-7, 24-10, 24-11, 37-11, 37-17
Duplicate Model Definition 57-11
Edit Activities 66-16
Edit Behavior Sets 25-19, 25-20
Edit Spawn Pattern 20-15

dialog box (continued)
Embark On 19-9, 26-21
Embarked 30-16
Emitter 30-18
Engagement Results 30-18
Environment Conditions 11-12
Environment Settings 11-3, 11-4
Environmental Object Information 18-3
Equipment Pacing/Tracking 30-19
Execute Close Air Support 26-21
Experimental Features 11-19, 44-20
Fire Cruise Missile 26-24
Fixed Wing Land 26-26
Fixed Wing Takeoff 26-27
Flee from Entities 26-27
Follow Entity 26-31
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant 30-19
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant Pacing/Tracking 30-20
Force 30-21
Formation 30-22
Heading 30-23
Health 30-23
IFF 30-24
Improve Booby Trap 27-22
Improve Breach 27-23
Improve Bridge 27-23
Improve Ditch 27-23
Improve Fortification 27-24
Improve Obstacle 27-24
Indirect Fire 27-25
Information 18-3, 18-30
information, configuring 75-3
Initial Scenario Information 7-3, 7-11
Jam Targets 30-25
Lase Autonomous 30-25
Laser Code 30-26
License Setup 2-7
Load Batch File 7-40
Load Overlays 34-6
Location 30-26
locking 75-7
Manage Reactive Task 25-16
Manage Reactive Tasks 20-22
MOPP Level 30-27
Morale 30-27
Move Along Route 26-36
Move Into Formation 26-37
dialog box (continued)
Move to Altitude 26-38
Move to Depth 26-38
Move to Location 26-39
Move to Waypoint 26-42
Munition Target Settings 10-3, 10-6, 10-8, 10-10
New Scenario 7-35, 12-3
Notify Level 30-28
Nuclear Attack 27-27
Open Model Set 63-7, 63-15
Ordered Speed 30-29
Overlay Layers 64-7
pages, configuring 75-2
Patrol Between 26-46
Patrol Route 26-45
Pattern Hold (Location) 26-46
Pattern Hold (Waypoint) 26-47
Percent Complete 30-29
Performance Options 6-14
configuring slider 6-15
Place IED 26-48
Plug-ins 4-33
Posture 30-30, 30-31
Preferred Targets 30-32
Protest Along Line 26-49
Protest Around Location 26-49
Radar Mode 30-33
Resources 30-34
Resources Pacing/Tracking 30-34
Resupply Mode 30-35
Rotary Wing Land 26-54
Rules of Engagement 30-36
Sail Heading 26-54
Save Scenario 7-28
Scenario Event 8-11
Scenario Information 7-23
Scenario Snapshots 7-26, 7-31, 7-34, 7-35
Scenario Startup 4-6, 4-8
Sector of Responsibility 30-37
Send NBC Report 27-28
Send Obstacle Report 27-28
Set Scenario End Time 7-36
Set Sensor Signatures 8-9
Simulation Connections Configuration 4-33
Simulation Object Type Mappings 58-2
Sonar Depth 30-39
Spot Reports 30-40
Strafe Ground Target 26-59
Superior 30-41

dialog box (continued)
Supplying 30-41
Surrendered 30-41
Synchronize Laser Code 30-42
Target 30-42
Taxi and Launch from Carrier 26-60
Taxi on Carrier 26-61
Terrain Settings 48-13
Add Props page 50-26, 56-9, 56-13
Edit Existing Props page 50-28, 50-29, 50-30, 56-24, 62-10
Extract Ocean page 50-15
Extract Props page 50-25, 56-21
Raster Maps page 50-7, 50-10, 56-5
SpeedTree Randomization page 59-21
Terrain Contents page 50-5, 50-12, 50-15, 50-16, 50-20, 51-3, 56-3, 56-21
Visible Surfaces page 48-3
Tracer Use 30-43
Turn to Heading 26-61
Unit State 30-43
User Task 26-62
Visual Model Editors 57-8
Wait Duration 26-63
Wait Elapsed 26-63
Weapons Pacing/Tracking 30-44
Weather 11-14
diffuse, color 38-4
digging
anti-tank, ditch 27-12
DI-Guy
action table 67-11
adding new characters 67-10
animation
adding 67-11
testing for 31-11
appearance, testing for 31-11
character data file 5-8
choosing animation 58-6, 67-13
configuring A-4
animations and appearances 5-8
character and appearance 64-14, 67-10
use of A-6
integration with VR-Forces 26-14
license 1-18
model definition 57-9
motion file 67-11
DI-Guy Animation task 26-14
DI-Guy Appearance dialog box 30-15
DI-Guy Appearance set data request 30-15
DI-Guy Hand Item dialog box 30-15
DI-Guy Hand Item set data request 30-15
diGuyAnimationsFile parameter A-4
diGuyCharacterDataFile parameter A-4
dipping, sonar 26-57
direction
line, changing 35-16
Lua 29-18
tidal current 11-15, 11-17
wind 11-9
directory
appData 5-8
back-end data 5-8
back-end user data 5-10
data 2-4
installed 2-4
DirectX 50-34
DIS
command-line options 5-19
configuring A-7
encoding information in OpenFlight format 61-2
exercise ID 5-7, 5-12
heartbeat, threshold A-7
IPV6 5-12
port 5-7, 5-12
port number 5-19
self reflection 5-12
specifying broadcast modes 5-19
starting in 5-3
startup tutorial 4-10
timeout 5-8
WRM specification 61-1
disable_if_checked 68-24
disable_if_multiple_selection 68-24
disabled-by-suppression parameter 70-18
disableParallelPreprocess parameter A-4
disabling
advisory messages A-7
buoyancy 39-6
cockpit displays 46-4
creation of a simulation object type 64-6
decal effects 37-4
editing of overlay 34-4
entity labels 18-14
environment 38-13
frame rate statistics A-5
intervisibility 41-10
joystick 19-21
joystick use 19-19

disabling (continued)
periodic checkpointing 7-33
radio communication lines 37-20
range rings 18-25, 18-27
reactive task 25-14
Scenario Startup dialog box 4-8
sensor contact line 37-13
shadows 38-14
speed scaling 44-27
spot report 9-3
spot report for simulation object 30-40
squawk indicators 37-20
tactical smoke 37-17
terrain paging information 51-7
texture compression 50-33
track histories for all simulation objects 37-5
trailing effects 37-4
view constraint 19-4
wireframe mode 48-11
disaggregated state, unit 23-9
disaggregated unit 21-2
disaggregation
area 23-12, 33-2
automatic 23-12
manual 23-12
Disembark, command 19-11
disembark 19-11
all entities on parent 19-11
in echelon view 19-11
Disembark All task 26-13
Disembark task 26-13
Disembarked set data request 19-11, 30-16
disembarked status, testing for 31-11
disembarking
entities 19-7
entity 30-16
instantly 19-11
Disperse Crowd task 26-13
dispersing, crowd 26-13
display
dragging 44-15
field of view 73-15
loading configuration 5-3
navigating 44-3
perspective 73-15
time, format 11-4
unit, changing 74-9
display engine 73-2
configuration 73-3
loading 73-6
display engine (continued)
configuration 73-3
saving 73-6
window, adding 73-3
Display Engine Configuration Editor Panel 43-6
display mode, curvature of the earth 37-21
Display Settings dialog box 4-23, A-2
Cockpit Display Settings page 46-5
Display Units Settings page 74-11
Entity Display Settings page 18-13, 18-14, 18-22, 18-23, 18-24, 18-36, 19-21, 19-23, 19-27, 25-5, 37-12, 37-18, 39-5, 39-8, 43-12
Grid Lines Settings page 48-8
Indirect Fire Settings page 10-7
Interest Management Settings page 6-9
Intervisibility Settings page 41-6, 41-7, 41-9, 41-10, 41-11
Lighting Render Settings page 38-4, 38-5, 38-6
Loader Settings page 50-35
Observer Settings page 18-27, 18-31, 18-32, 19-26, 19-27, 37-4, 37-5, 37-14, 38-7, 39-3, 43-4, 43-5, 43-6, 43-9, 44-5, 44-23, 44-25, 44-26, 44-28, 46-10, 48-7
Ocean Render Settings page 11-18, 38-8, 38-12, 39-4, 50-18, 50-19
Radio Communications Settings page 37-20, 37-21, 37-22
Range Rings Settings tab 18-28
Render Settings page 6-7, 6-18, 38-10, 38-11, 44-24, 48-12, 50-23, 73-13
Sensor Settings page 40-4, 40-5
Shadow Settings page 38-15, 38-16
SpeedTree Settings page 59-19
Symbol Decoration Settings page 18-15
Tactical Graphics Display Settings page 33-7, 33-8, 33-9, 36-3
Terrain Profile Settings page 41-12
Track History Settings page 37-7
Unit Display Settings page 24-7, 24-10, 24-11, 37-11, 37-17
Display Settings Toolbar 18-16
Display Terrain parameter 7-6
Display Units Settings page 18-24, 33-9, 74-11
displaying
contour lines 48-9
entity information 18-10
feature data 48-14
full screen 74-6
grid line 48-7

displaying (continued)
height-above-terrain lines 18-24, 33-9
intervisibility
lines 41-10
object 41-7
label
2D 18-15
entity 18-12
laser designator 37-16
levels of aggregation in Echelon View 24-4
navigation areas 53-5
navigation data 53-11
objects 33-5, 33-7
other MAK 3D GUI viewers 43-12
panel 74-3
performance statistics 6-6
range rings 18-25
globally 18-27
per entity 18-27
raster maps 50-7
resources 18-30
scenario information 7-21
Scenario Startup dialog box 4-8
segment outlines 60-5
stealth models 43-12
tactical graphics 30-5, 33-7, 35-18
terrain profile, creating line 35-6
toolbars 74-3
track histories 37-5
trailing effects 37-3
transient intervisibility 41-6
vertex labels 33-8
view 44-30
disPort parameter A-7
distance, specifying unit 74-9
Distance To Attached, policy 73-11
Distributed Interactive Simulation 1-23
See also DIS
ditch
anti-tank, digging 27-12
building 27-16
destroying 27-20
improving 27-23
DLL 4-33
cockpit display 59-10
installing cockpit display, installing 59-11
plug-in, removing 4-37
specifying for plug-in 4-37
dockable panel 74-2
docking, panel 74-2
documentation, location of 2-4
DOF bead 61-7
doNotUseConsole parameter A-4
dotted line 35-17
Douglas sea state 11-15, 11-17
drag-and-drop, canceling 16-20
dragging
canceling move 16-20
object to new location 16-19
objects 30-26
simulation object 16-19
terrain 44-14
view 44-15
drainInput
configuring A-5
for DIS A-7
for HLA A-6
dr-allow-gui-overrides parameter 6-15
drawing, box 35-3
driver
OGR 52-8
simple 52-18
driving, on and off roads 25-21
Drop Naval Depth Charge at Location task 26-15
Drop Naval Depth Charge task 26-14
Drop Naval Mine task 26-16
Drop Naval Mines Along Route task 26-17
dropping, mine 26-16, 26-17
dropping bomb 26-53
dropping from exercise 3-7
dry gap, building 27-16
DTED
file 47-3
loading 50-4
loading multiple 52-6
support for 50-4
DtNotifyLevel parameter A-4
DtSimComponentManager class 68-9
DtSimObjectNetInterface class 68-9
DtSimObjectStateRepository class 68-9
DtSimTaskManager class 68-9
DtVrfObject class 68-9
duplicate, key mapping 76-8
Duplicate Model Definition dialog box 57-11
dust cloud 37-3
dynamic avoidance 3-16
dynamic feature 55-10
enabling and disabling A-4
dynamic lighting 38-2, 38-6
dynamic linked library 4-33

Dynamic Near Clip Attached Policy, attribute 73-11
dynamic ocean 1-15, 11-15
  enabling and disabling 11-16
  layer 50-13
  wake and spray, disabling 39-5
  wakes 59-14
Dynamic Water Visibility Enabled, attribute 73-17
Dynamic Water Visibility Maximum, attribute 73-17
dynamicFeaturesDisabled parameter A-4

E

-E command-line option 5-3
-e command-line option 54-3
earth file 1-19, 47-5, 52-3
  adding mask 52-13
  areal features 52-11
  buoys and beacons 52-19
  caching, offline 50-32, 52-16
  CDB support 52-16
  differential processing of attributes and elements 52-17
  hiding layers 48-13
  including files in 52-5
  loading DTED 52-6
Earth Layers page 48-13
echelon ID 15-2, 15-4, 15-5
  assignment of 15-6
  unit member 21-3
echelon level, ghosted icons by 24-7
Echelon View 17-2, 23-7, 23-8, 24-4
  aggregated unit 23-9
  disaggregated unit 23-9
  disembarking 19-11
  expanding and collapsing units 24-4
  ghosted units 24-6
  levels of aggregation 24-4
echelon-level parameter 68-25
ECW 47-4
Edit Activities dialog box 66-16
Edit Behavior Sets dialog box 25-19, 25-20
Edit Existing Props page 50-26, 50-28, 50-29, 50-30, 56-24, 62-10
Edit Spawn Pattern dialog box 20-15
editing 11-14
  assembly 66-12
  attack parameters 66-22
editing (continued)
  batch file 7-38
  channel attributes 73-8
  control object 35-12
  defense parameters 66-23
  electronic warfare parameters 66-23
  element definition 57-16
  ellipse 35-16
  embarkation ingress and egress points 70-16
  entity model 64-3
  environment condition 11-7
  equipment parameters 66-24
  fill style 35-17
  force hostility 9-10
  formation 65-8
  frustum 73-15
  icon, 2D 59-2
  indirect fire event 10-6
  intervisibility object 41-8
  key maps 76-3, 76-4
  line 35-17
  Lua scripts 28-37
  model, visual 64-16
  model definition 57-8, 57-12
  model mapping 58-4
  movement system properties 66-20
  navigation area 53-4
  object
    location, heading, altitude 19-5
    scene 35-13
  object map file 12-9
  object parameters 64-3
  observer mode 43-5
  overlay and overlay object 34-4
  parameter 64-11, 69-4
    model definition 57-13
    schema 57-7
  personnel parameters 66-24
  plan 32-2
    file 12-8
    statements 32-10
  plans 32-2
  posture, transition time 66-21
  scenario description 7-23
  scenario file 12-3
  scripted task 28-28
  sensor 71-4
  sensor parameters 66-21
  sensor signature 71-5
  simulation object 63-2

editing (continued)
simulation objects 19-5
spawn pattern 20-15
supplies parameters 66-24
system 64-25
properties 64-24
tactical graphic 35-12
property 64-45
visual model 64-46
unit 65-4
unit subordinate 65-7
vertex 35-14
visual definition 57-17
parameter 57-18
weather zone, environment 11-14
window attributes 73-7
effect
fire and detonation 37-8
lighting 1-16, 50-20
mapping 58-2
model 56-4
ocean 11-15
trailing 37-3
egress point 64-33, 70-16
configuring 64-35
egress-points parameter 70-16
electromagnetic emissions 30-17, 37-18
segment angle 60-4
sensing 22-19
electronic
emission, sensing 22-16
warfare 22-16, 22-18, 22-19
electronic warfare 22-17
electronic warfare parameter, editing 66-23
element
earth file, differential processing 52-17
informationitem 75-4
informationpagecollection 75-3
mainmenu 75-2
mainmenuconfiguration 75-2
mask 52-13
menuitem 75-5
model 52-7, 52-18, 59-20
option 52-17
pagecollection 75-3
pagecollections 75-3
parent 57-17
rightclickmenu 75-5
toolbar 75-6
toolbarrow 75-6
element definition 57-14
adding visual definition 57-16
common buttons 74-7
creating 57-15, 67-6
deleting 57-20
editing 57-16
filtering list in Simulation Object Type Mappings dialog box 58-5
saving 57-20
simulation object 57-14
element_type 68-24
elevation
coloring terrain by 48-4
data 47-3
intervisibility point, configuring 41-9
Elevation Color page 48-4, 48-5, 48-6
ellipse
extruded 35-4
rotating 35-16
undoing and redoing rotation 35-16
ellipsoid, reference A-4, A-6
Embark On, command 19-9
Embark On dialog box 19-9, 26-21
Embark task 19-8, 26-18
embarkation
automatic 19-10
condition 31-11
configuring 64-33
slot exclusions 64-40
editing ingress and egress points 70-16
egress point 64-35
embedded entity alternative 19-12
immediate 19-8
ingress point 64-35
instant 19-10
occupancy-director-controller 70-16
slot 70-16
configuring 64-35, 64-38
Embarkation View 17-2, 19-7, 19-8, 19-11
embarking objects in 19-10
embarkation-slot-entry parameter 70-16
embarkation-slots parameter 70-16
Embarked dialog box 30-16
embarked object, information 18-6
Embarked set data request 19-9, 30-16
embarking
entities 19-4, 19-7
in Embarkation View 19-10
instantly 19-9

embedded entity 19-12
  assigning tasks to 19-16
  configuring 19-12
  deploying 19-15
  deploying with task 19-15
  recovering 19-16
  restoring parent 19-16
  Simulation Object Editor 19-12
  system 19-12
embedded window 73-2
EM-Emission update message 60-3
emission sensor 22-19
emissions-sensor system 22-19
emitter 30-33
  beam 9-17, 30-17
  component 30-17
  configuring 70-11
  ID 30-18
  information 18-6
  setting 30-17
  systems 37-18
Emitter dialog box 30-18
Emitter Power 60-3
Emitter set data request 27-4, 27-19, 30-17
emitter volume
  color 60-2
  configuring 60-2
  outlines 60-5
  pulse rate 60-2
  radius 60-3
  segment angle 60-4
  segment size 60-4
enable_if_checked 68-24
enableDebugDrawer parameter A-4
Enable/Disable Spot Reports tab 9-3
enabling
  advisory messages A-7
  buoyancy 39-6
  cockpit displays 46-4
  creation of a simulation object type 64-6
  decal effects 37-4
  editing of overlay 34-4
  entity labels 18-14
  external communication model 13-3
  frame rate statistics A-5
  intervisibility 41-10
  joystick use 19-19
  marine effects 11-16
  periodic snapshot 7-34
  radio communication lines 37-20
enabling (continued)
  range rings 18-25, 18-27
  reactive task 25-13
  Scenario Startup dialog box 4-8
  sensor contact line 37-13, 37-14
  shadows 38-14
  sound effects 42-2
  speed scaling 44-27
  spot report 9-3
    simulation object 30-40
  squawk indicator 37-20
  tactical smoke 37-17
  terrain paging information 51-7
  texture compression 50-33
  time management 4-19
    in back-end 4-20, 5-18
  track histories for all simulation objects 37-5
  trailing effects 37-4
  wireframe mode 48-11
ending
  scenario 7-27
  scenario event 8-12
engagement, information 18-7
Engagement Results dialog box 30-18
Engagement Results set data request 30-18
engagement rules 30-36
Engineering Object Breached condition 31-10
Engineering Object Information page 22-12
entitiy, ways to add to scenario 7-16
entity
  See also simulation object
  adding to, unit 23-7
  adding to Props Palette 16-26, 57-8
  aggregate-level
    creating 66-3
    parameters 66-17
  aircraft, attacking with gun 26-6
  altitude
    displaying 18-24
    testing 31-10
  animation 26-5
  appearance 61-2
  attaching observer to 45-2, 45-6
  avoiding obstacles, features, and collisions 19-17
  body frame coordinate system 44-7
  bounding volume 18-31, 18-32
  category, changing 64-3
  centering observer on 44-25
  clamping to ground 19-21

entity (continued)
component, attaching to remote 70-18
components 68-13
configuring
damage by munitions 70-7
emitter 70-11
obstruction avoidance 70-15
console message 18-32
controlling by keyboard 19-21
controlling with joystick 19-19
copying 16-21
count 18-37
creating 16-3, 16-4, 16-5, 16-8, 16-9, 16-11, 19-3
from existing 64-32
in Simulation Object Editor 64-31
creating group of 20-17
creating in plan 32-17
damaging 9-22
deleting 19-5
from simulation model set 64-42
deleting in plan 32-17
disembarking 19-7, 30-16
all 19-11
display settings 5-3
dragging 16-19
editing 19-5, 63-2
element definition 57-14
embarkation
configuring 64-33
testing 31-11
embarking 19-4, 19-7
embedded 19-12
enumeration
in condition 32-7
in plan 32-9
extents, zooming on scenario load 44-21
filtering 6-8, 6-9
attachment list 45-5
fixed-wing 25-22
route following 25-26
following
entity 26-31
route 26-36
generating navigation data 53-8
ghosted 24-6
ground, road driving 25-21
ground clamping 19-23
heading, setting 16-17
entity (continued)
icon
adding new image 59-9
changing size 18-22
image, adding 64-20
inset view 46-2
instrument panel, configuring 46-7
intervisibility 41-4
label 18-15
2D 18-12, 18-15
3D 18-10
color 18-10
enabling and disabling 18-14
pinning to display 18-14
level body frame coordinate system 44-7
mapping sound effects to 42-4
messages, viewing 18-34
model 56-4, 63-5
adding visual 64-20
editing 64-3
editing visual 64-16
movement, scripted 64-26
moving 16-19
moving to
altitude 26-38
location 26-39
object or entity 26-42
name 18-10, 18-11
changing 19-5
pasting 16-23
notification level, object console 18-33
object geometry 64-13
organization 15-5
pasting, control object 16-22, 16-23
patrolling between 26-46
patrolling route 26-45
placing 16-11
new 19-3
on entity 19-4
platform, updating 63-17
primary 45-9, 45-12
properties, specifying at creation 16-13
range ring 18-27
resource
at creation 19-4
monitoring 18-30
routes for aircraft 35-10
saving console messages 18-35
scaling, 3D model 19-24
secondary 45-9, 45-12

entity (continued)
setting
collision avoidance types 30-11
target 30-42
shadows 38-14
sound effects 42-2
spawned 20-2
spawning, maximum simultaneous 20-11
specifying for multiple entity creation 20-20
storing list of 12-10
surface
buoyancy 39-6
configuring wake 59-14
systems 64-22
task, visualizing 37-12
tick rate 6-12
trailing effects 37-3
type, enumeration 18-11, 58-3
unit 23-3
velocity, publishing A-5
VR-Forces 3-9
wake, enabling and disabling 39-5
zooming to extents 44-20, 44-21
Entity Altitude condition 31-10
Entity Behavior Options page 23-6, 23-11
Entity Definition Editor page 59-2, 59-14, 60-6, 62-5
Entity Destroyed condition 31-10
Entity Di-Guy Animation Check condition 31-11
Entity Di-Guy Appearance Check condition 31-11
Entity Display Settings page 18-13, 18-14, 18-22, 18-23, 18-24, 18-36, 19-21, 19-23, 19-27, 25-5, 37-12, 37-18, 39-5, 39-8, 43-12
Entity Editor. See Simulation Object Editor
Entity Embarked condition 31-11
entity enumeration, changing 64-4
entity file, importing 64-52
Entity Has Target condition 31-11
Entity Image Symbol, visual definition 59-2, 59-4, 59-9
Entity In Area, condition 31-11
Entity In Area condition 35-4
Entity Left of Line condition 31-12
Entity List, filtering 17-6
Entity Mapping Settings page 58-3, 58-4, 58-6, 62-8
entity plan 31-2
Entity Resource Manager 70-3
Entity Sound Editor page 42-5, 42-6, 42-7
entity type, changing 64-4
Entity Under Fire condition 31-12
Entity View 17-2
entity-level, modeling 15-9
entity-level modeling 20-1
EntityLevel.sms 15-10
entity-type parameter 70-4
entrance, embarkation 64-35
entry point, scripted task 29-7
entVrfDataTimeThreshold parameter A-4
enumeration 58-3, 68-4
entity, changing 64-4
for condition patterns 32-9
in condition 32-7
object type B-9
published and matching 68-6
environment
disabling 38-13
loading configuration 5-3
weather zone 11-14
editing 11-14
environment condition 11-5
adding 11-7
deleting 11-8
editing 11-7
marine, setting 11-16
Environment Conditions dialog box 11-12
Environment Conditions Settings page 59-16
Environment Settings dialog box 11-3, 11-4
Environment Settings Toolbar 11-4
Environment Settings toolbar 11-3, 11-4
environment variable
MAK_GEOID_GRID 50-4
MAK_VRF_ENABLE_DEBUG_DRAWER 51-8, 51-9
MAKLMGRD_LICENSE_FILE 2-8
OpenSceneGraph, anaglyphic stereo 73-22
OSG_CURL_PROXY 51-5
OSG_CURL_PROXYPORT 51-5
OSG_ICO 6-11
OSGEARTH_CACHE_PATH 50-32
OSGEARTH_CURL_PROXYAUTH 51-5
RTI_CONFIG 2-10
Environmental Object Information dialog box 18-3
Environmentals List, filtering 17-6
Environmentals View 17-2
ephemeris 7-7
model 11-2

equipment
pacing and tracking 30-19
unit 66-5, 66-8
Equipment Pacing/Tracking dialog box 30-19
Equipment Pacing/Tracking set data request 30-19
equipment parameter, editing 66-24
equivalent task 28-22
escaping
plan 32-24
task assignment 25-6
event 15-8
linked, adding 8-6
list, numbering 8-13
scenario 8-2
Event window 8-10, 8-12
EW-Comms-Degradation-Percentage 22-17
EW-Comms-Dependence 22-17
EW-Defense-Strength 22-17
exaggerated reality 43-10
exaggerating, 3D models 19-24
exclusion, slot 64-40
execName parameter A-6
exectool, module 29-6
executable, installation locations 2-4
Execute Close Air Support dialog box 26-21
Execute Close Air Support task 26-21
execution name, HLA, specifying 5-16
exercise
clock 3-10
starting 7-24
exercise ID A-7
DIS 5-7, 5-12
exerciseId parameter A-7
exercise-start-time parameter 12-6
exit, embarkation 64-35
exiting
application 4-25
full screen window 74-6
plan 32-24
scenario 7-27
simulation engine 4-25
expanded label 18-10
pinning to display 18-14
expanding
unit 24-2, 24-3
all 24-3
Experimental Features dialog box 11-19, 44-20
Explode Charge at Depth task 26-23
explosion 37-9
explosive device, arming 30-9
exporting
scripted task 28-30
settings 4-23
extended label 18-18, 33-3
adding 18-19
deleting 18-20
index, changing 18-20
extended name 28-12
extent, zooming to 44-20
external communication model 15-6, 70-13
configuring 13-3
enabling 13-2, 13-3
external communications effects server
configuring VR-Forces for 1-18, 13-2
QualNet 13-4
external reference 59-22
external script, editing 28-37
Extract Ocean page 50-15
Extract Props page 50-25, 56-21
extracting
prop 56-20
from terrain 47-4
from terrain patch 50-24
water texture 50-15
extruded
building 52-12
tactical graphic 35-4
extruded area 31-11
extruded geometry 47-5
eyepoint. *See* observer

F

-F command-line option 5-4, 5-13
-f command-line option 5-6, 5-11, 5-17, 54-3
factory settings, restoring 4-23
fan, intervisibility 41-4
far clipping plane 73-10
FASCAM task 27-22
favorite 16-8
adding or removing from list 16-26
joystick configuration 72-11
observer view 44-31
favorites.mst 16-26
FDD file 2-10
feature 1-14
areal 52-11
attribute 55-4
avoiding 19-17, 70-15

feature (continued)
configuration file 55-6
configuring avoidance of 70-16
coordinate 48-17
data
buoys and beacons 52-19
configuring 55-2
configuring in front-end and back-end 50-11
displaying 48-14
extracting from GDB 55-7
layer 55-7
ocean 50-19
processing 52-6
dynamic 55-10
enabling and disabling A-4
extracting as prop 50-24
geometry 47-5
layer, adding 47-4
query 55-6
querying 55-3
soil type 48-17
vector, list of 48-15
feature layer 52-19
adding 50-11
adding prop from 50-26
name, specifying 55-2
Feature Settings page 48-16
featureconfig.txt
mapping, features 50-11
featureconfig.txt 70-16
FeatureData section in meta file 59-16
featureDataBroker 52-6
FeatureName parameter 59-17
FeatureSet class 29-3
FED file 2-10
name 5-6, 5-11, A-6
specifying 5-18
federate
name 5-6, 5-11, A-6
time managed 4-18
federateName parameter A-6
federateType parameter A-6
federation execution
destroying A-6
name A-6
specifying name 5-16
federation time 4-19
fedFileName parameter A-6
Feet Above Ground Level 59-12
Feet Above Mean Sea Level 59-12
FeetAGL updater 59-12
FeetMSL updater 59-12
FeetPerMinute updater 59-12
field of view 44-18, 44-19, 60-3, 73-15, 73-16
changing 73-15
file
batch 7-37
cache, clearing 50-33
caching 50-31
compressed 59-23
FED 2-10, 5-18
formation 70-9
hostility relationships 12-10
installed 2-4
log 4-40
MTL A-9
object map 12-9
order of battle 12-10
physicalWorldParams.mtl 71-9
plan 12-8
RID 2-10, 5-16
rid.mtl 5-16
scenario 6-1, 12-2, 12-3, 12-7
scenario extras 20-9
simulation model set 63-5
transporter 5-4
vrfGui.log 4-40
vrfSim.log 4-40
vrfSim.mtl A-3
vrfSimSettings.xml 6-14
File Caching Settings page 50-31, 50-33, 58-7
Filename parameter 59-11
filename_shp_map.txt configuration file 50-11
fill style 35-17
filtering
earth file layers 48-13
element definition list 58-5
function list 76-9
model definition list 57-11
object attachment list 45-5
object lists 17-6
scripted task list 28-28
simulation object list 25-9
simulation objects
by observer altitude 6-9
in scene 6-8
finite state machine 29-6
fire
effects 37-8

fire (continued)
event
mapping sound effects to 42-4
sound effects 42-2
indirect 10-2
line 37-9
suppressive 26-50, 26-51
at location 26-50
at point, line, or area 26-50, 26-51
target 26-23
Fire at Target task 26-23
Fire Cruise Missile dialog box 26-24
Fire Cruise Missile task 26-24
Fire Sound Editor page 42-8
Fire-for-Effect task 26-25
firepower-kill parameter 70-6
fire-when-fired-upon 30-36
firing
at target 26-23
ballistic missile 10-8
cruise missile 26-24
effects 58-2
target at 9-13
spot report 9-15
Fixed Near Clip attribute 73-11
Fixed Near Clip When Attached, attribute 73-11
Fixed Wing Land dialog box 26-26
Fixed Wing Land task 25-28, 26-26
Fixed Wing Takeoff dialog box 26-27
Fixed Wing Takeoff task 25-27, 26-27
fixed-frame 4-19
fixed-frame best-effort 3-11, 12-6
fixed-frame keyword 12-6
fixed-frame-run-to-complete 3-11, 4-19
fixed-frame-run-to-complete keyword 12-6
fixed-wing
altitude 26-30
holding pattern 26-46, 26-47
strafing target 26-58
taxiing on carrier 26-61
fixed-wing entity
altitude 26-28
attacking with gun 26-6
circling a point 26-45
counter measures 26-33
creating 25-22
creating routes for 35-10
dropping bombs 26-53
follow entity behavior 26-31
heading 26-29, 26-30
fixed-wing entity (continued)
landing 25-28, 26-26
route following 25-26
routes for 35-10
setting
altitude 25-24
IFF transponder 30-24
speed 30-29
takeoff 25-27
takeoff and landing 25-25
taking off 26-27
terrain following 25-25
writing plans for 31-17
fixed-wing-entity-param parameter 68-12, B-8
fixed-wing-entity-state-repository 68-12
flag, movement in wind 59-16
flaming effects 37-8, 58-2
flare 26-33
launching 9-19
flat earth, coordinate system 44-7
Flee from Entities dialog box 26-27
Flee from Entities task 26-27
FLEXlm 2-6
flipping, DDS textures 50-6, 59-17
FLIR 40-2
floating, panel 74-2
FLT file 47-3, 56-4
Fly Altitude task 26-28
Fly Heading and Altitude task 26-30
Fly Heading task 26-29
flying
altitude to 26-28, 26-30
heading to 26-29, 26-30
focal depth 44-22
focal range 44-22
focus, scene 44-22
fog 11-5
denseness of 11-10
height and color 11-10
fog of war 5-4
fog-of-war 9-2
same side 9-7
folder
adding scripted task to 28-29
removing scripted task from 28-29
scripted task 28-29
adding 28-29
deleting 28-29
renaming 28-29
follow entity, fixed-wing entity behavior 26-31

Follow Entity dialog box 26-31
Follow Entity task 26-31
  in plan 31-16
Follow mode 44-8, 45-6, 45-7
Follow Route task 26-36
following
  paths 3-15
  route 26-36
  simulation objects 26-31
FOM 2-9
  module 5-6, 5-11
FOM Mapper
  configuring A-6
  initialization data 5-6, 5-11
  initialization string A-6
  library name 5-6, 5-11
  specifying at startup 5-17
fomMapperInitData parameter A-6
fomMapperLib parameter A-6
food 22-14
  pacing and tracking 30-20
  setting 30-19
Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking set data request 30-20
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant dialog box 30-19
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant Pacing/Tracking dialog box 30-20
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant set data request 30-19
footprint 37-3
  affect of posture 22-5
  sensor 22-8
  unit 22-4
force 15-5, 18-11
  adding 64-50
  Behavior Set, assigning to 25-20
  color, changing 64-51
  editing 64-51
  hostility
    changing 64-51
    changing in plan 9-12
    matrix 9-10
  name, changing 64-51
  removing 64-52
  setting at runtime 30-21
Force dialog box 30-21
Force set data request 30-21
forceHostilty.mtl 9-10
forceParameterDbReload parameter A-4
format
  OpenFlight 61-2
  time display 11-4
formation 21-4
  adding 65-9
  bounding box 65-11
  closing 21-5
  configuring 70-8
  copying 65-12
  default 65-10
  editing 65-8
  expanding and collapsing in Formation Editor 65-10
  file 70-9
  layout
    automatic 65-13
    manual 65-13
  moving to 26-37
  removing 65-10
  renaming 65-10
  setting 30-22
    in plan 31-16
  snap to grid 65-14
  supported 30-22
  user-defined 70-10
Formation dialog box 30-22
Formation Editor 65-9, 65-10, 65-12, 65-13
Formation set data request 30-22
  in plan 31-16
fortification
  breaching 27-9
  destroying 27-21
  improving 27-24
fortified area, constructing 27-17
fortified line, constructing 27-18
forward arrow 64-42, 65-7
frame, length of 12-6
Frame Mode parameter 7-7
frame rate 5-10, 6-4, 6-17, 6-19
  exercise clock 3-11
  mode, specifying 12-6
  setting 5-4
  statistics
    enabling and disabling A-5
    log file A-4
Frame Rate Monitor panel 6-5
Frame Time parameter 7-7

frame-mode parameter 4-20, 12-6
  time management 4-19
frameRateStatisticsLogFile parameter A-4
frame-time parameter 12-6
free-fly mode 45-7
freehand line 35-2
  sampling rate 35-2
frequency
  color of 60-2
  radio communication line 37-22
  sampling for terrain profile 49-3
front-end 3-2
  coordinating multiple 3-6
  feature representation 47-5
  joining session at startup 4-17
  multiple 3-2
  process ID 5-8
  selecting database 7-3
  sending messages 3-3
  starting 4-3
frustum 73-8, 73-20
  editing 73-15
fsm, module 29-6
fuel 22-14, 30-10
  pacing and tracking 30-20
  setting 30-19
  specifying amount 30-34
  status 18-4
Full Color Mode 40-5
Full Knowledge CID level 9-13
full screen 5-5
  mode 5-4
  window 73-2, 74-6
function
  list, filtering 76-9
  movement 76-2
  profiler 6-7

G

-G command-line option 5-4, 5-15
-g command-line option 54-4
gain 40-2
  changing 44-28
  sensor effects 40-4
game pad 19-19
Gameware Navigation Lab 53-11
gasoline 22-14
GDAL 55-7
GDB, extracting feature data 55-7
GDB terrain 47-3
  configuring soil type 48-2
  shapefile 47-4, 50-11
generating
  navigation data 53-5, 53-6
    entity for 53-8
  traffic 20-2
geocentric
  converting to geodetic A-4, A-6
  coordinate system 44-7, 47-6
    vector 29-20
  terrain 47-6, 51-2
geodetic, converting from geocentric A-4, A-6
geographic DDM 6-8, 6-9
geoid grid file 50-4
geometric, objects 29-18
geometry 59-22
  extruded 47-5
  object, adding to entity 64-13
  sharing 58-7
Geometry Grid Dataset 54-2, 54-6
GeoTIFF 47-4
  image 50-9
GGDS 54-6
ghosted
  icon, by echelon level 24-7
  unit 24-6
GIF 47-4
GL Studio 1-20, 46-4, 59-10
  instrument panel 46-6, 46-7
  license 1-21
global command
  adding to plan 32-15
  Create 32-17
  Delete Object 32-17
  Delete Self 32-17
  Issue Plan 32-18
global plan 11-2, 31-2
  creating 32-11
  file 6-1
global setting 4-25
God ray 38-11
graphic
  adding to scenario event 8-5
  buffer, depth and stencil 6-13
  depth 5-3
  fire and detonation 37-9
  mode, wireframe 48-11
  object, showing and hiding 33-5, 33-7
  unit combat 37-10

graphical object 33-2
  changing, line width 35-17
  deleting 35-11
  spot report 9-9
graphical user interface 3-2
  configuring 75-2
grid line
  configuring 48-8
  displaying 48-7
Grid Lines Settings page 48-8
grid spacing, changing 65-14
ground, attacking from air 27-4
ground clamping 19-21
  configuring 19-23
Ground Clamping Cutoff Distance Scale 19-23
ground truth 9-2, 9-4
ground vehicle
  configuring obstruction avoidance 70-15
  road driving 25-21
ground-db, database 48-18
ground-entity-state-repository 68-12
ground-vehicle-param parameter 68-12, B-8
ground-vehicle-param parameter 68-3
group
  entities, adding 20-20
  selection 17-8
  simulation object 16-25
GUI 3-2
  configuring 75-2
  layout settings 5-4
  localizing 2-11
  locking 75-7
  settings
    file 6-1
    scenario 12-2
  Simulation Object Editor, layout settings 63-4
GUI Layout page 74-4
guidelines 59-22
  creating models 59-22
GuiObserverViews parameter 12-6
GuiScenarioSettings parameter 12-6
Gui-Terrain-Database parameter 12-5
gumball 18-4
gun
  aircraft, attacking aircraft with 26-6

H

-H command-line option 5-6, 5-7
-h command-line option 5-4, 5-9, 5-13, 54-4
Halt Movement task 27-22
Hasty-Attack, posture 22-4
Hasty-Defense, posture 22-4
HAT*, See* height-above-terrain line
hazard, area 22-8
Hazards/Obstacles Palette 11-13, 16-4, 16-5, 22-9
header file*, objTypeEnums.h* B-9
heading 18-11, 44-3
  flying to 26-29, 26-30
  indicator 16-17, 18-12
  intervisibility line 41-11
  object, changing 19-5, 35-13
  observer 44-4
  pasting 16-23
  rotating icon to 18-23
  setting 16-3, 30-23
    manually 16-17
  specifying measurement unit 74-9
  turning to 26-61
Heading dialog box 30-23
Heading set data request 30-23
headlight 38-2, 38-6
Headlight updater 59-13
health 22-2, 22-3
  restoring simulation object 30-35
Health dialog box 30-23
heartbeat
  back-end, configuring A-6
  DIS A-7
  relationship to thresholds 6-14
  varying A-7
heartbeatVariance parameter A-7
height
  fog 11-10
  ocean 50-15, 50-16
height map
  ocean
    regenerating 50-19
    resolution 50-17, 50-18
    texture size 50-17, 50-18
height-above-terrain line 18-24, 33-5, 44-27
  displaying 33-9
helicopter
  rotor wash, high quality 39-4
  terrain avoidance 25-30
help 5-9, 5-13
heuristic, path planning 3-14
hide_by_default 68-24
hiding
  3D models 19-27

hiding (continued)
intervisibility lines 41-10
objects 33-5, 33-7
overlay objects 34-4
panel 74-3
simulation objects 9-2
symbol decorations 18-16
tactical graphics 30-5, 35-18
toolbar 74-3
vertex labels 33-8
hierarchy
displaying levels of aggregation 24-4
element definition 57-14
entity 15-5
high-explosive, resource 22-14
hit probability
file 68-26
table 70-4
HLA 6-8, 6-9
1516
troubleshooting 7-42
command-line options 5-16
federation execution name, specifying 5-16
parameters A-6
specifying federation execution 5-16
starting in 5-4
startup tutorial 4-10
time management 4-18
holding pattern
fixed-wing 26-46, 26-47
rotary-wing 26-46, 26-47
horizontal navigation 76-2
host address 5-6, 5-7, A-6, A-7
hostAddressString parameter A-6, A-7
hostile-fire-only parameter 70-17
hostility
changing, plan in 9-12
matrix, force 9-10
hostility relationships file 6-1, 12-10
hostname, license server 2-7
HUD. *See* cockpit display
human-param parameter 68-12, B-8
human-state-repository 68-12

I

-I command-line option 5-4
-i command-line option 4-9, 5-6, 5-7, 5-9
icon
2525B 59-2
icon (continued)
2D 16-9, 59-4
bitmap 59-4
changing 64-18
displaying label 18-15
editing 59-2
image 64-19
bitmap, adding new 59-9
borders 18-21
common 74-7
configuring 2D 59-2
heading, rotating to 18-23
image 59-4
adding new 59-9
level of nesting 24-6
location of 2-4
notification 18-36
scaling, 2D 18-22
settings 4-23
simulation object 18-22
size, changing 18-22
tactical graphic 64-43
Task menu 28-11
toolbar, setting 75-7
yellow 9-13
ID
echelon 15-4, 15-5
emitter 30-33
object 15-4
script 28-8, 28-11, 28-20, 29-15
session 4-3, 5-6, 5-7
site 5-10, A-5
task 29-15, 29-16
unique 15-3
identification
simulation object, UUID 15-2
Identification Friend from Foe 30-24
identifier, site and application 3-4
identifying, simulation objects 15-2
IED
arming 30-9
detonating 30-14
placing 26-48
If statement 31-4, 31-6
adding to plan 32-4
building 32-4
IFF 30-24
information 18-6
IFF dialog box 30-24
IFF set data request 30-24

illumination 71-12
  model 7-7, 11-4
  setting 11-2
image
  2D icon 64-19
  adding 59-9, 64-20
  attributes 50-8
  compressing 59-23
  display order 50-10
  displaying 50-7
  format, supported 47-4
  GeoTIFF 50-9
  icon 59-4
  texture coordinate system 50-9
imagery, filtering 48-13
imagesymbolmap attribute 59-7, 59-9
immediate embarkation 19-8
importing
  entity file 64-52
  MSDL 7-14, 7-16, A-5
  navigation area 53-10
  navigation data 53-10
  scenarios 7-13, 7-16
  scripted task 28-30, 28-31
  settings 4-23
Improve Booby Trap dialog box 27-22
Improve Booby Trap task 27-22
Improve Breach dialog box 27-23
Improve Breach task 27-23
Improve Bridge dialog box 27-23
Improve Bridge task 27-23
Improve Ditch dialog box 27-23
Improve Ditch task 27-16, 27-23
Improve Fortification dialog box 27-24
Improve Fortification task 27-24
Improve Obstacle dialog box 27-24
Improve Obstacle task 27-24
Improve Road task 27-27
improve-script-id parameter 28-20
improving
  anti-tank ditch 27-23
  booby trap 27-22
  breach 27-23
  bridge 27-23
  ditch 27-23
  fortification 27-24
  obstacle 27-24
  strong point 27-24
improvised explosive device, placing 26-48
inactive, tactical graphic 30-5, 35-18
including XML files in earth files 52-5
increasing speed of navigation 44-27
incremental compiling 6-11
independent mode, tutorial 4-10
independent setting, global and observer 4-25
independent task 24-1, 25-3
index
  extended label, changing 18-20
indicator 18-14
  heading 18-12
  squawk 37-19
indirect fire 10-2
  creating 10-3
  editing 10-6
Indirect Fire dialog box 27-25
Indirect Fire page 10-3, 10-6
Indirect Fire Settings page 10-7
Indirect Fire task 27-25
individual, plan 11-2
individual-lifeform-param parameter 68-12
individual-lifeform-state-repository 68-12
information
  connection 4-32
  entity label 18-10
  multiple objects 18-10
  object 18-3
  scenario, displaying 7-21
Information dialog box 18-3, 18-30
information dialog box, configuring 75-3
information message A-4
informationitem, element 75-4
informationpagecollection, element 75-3
infrared 40-2
ingress point 64-33, 70-16
  configuring 64-35
ingress-points parameter 70-16
inheritance, element definition 57-14
Initial Scenario Information dialog box 7-3, 7-11
Initial Unit State 23-6
initialization string, FOM Mapper A-6
initial-road-preference parameter 30-28
input
  of data 68-13
  port and port group 68-14
inserting, altitude point 48-5
inset view 46-2
installing, RTI 2-10
instancing 5-4
instant embarkation 19-8

instrument panel 46-6
  configuring 46-7
  GL Studio 46-6, 46-7
instrumentPanelCategory parameter 46-7
Intel Collection Area 35-9
intelligence object 8-8
intensity, rain and snow 11-11
interaction 3-12
  signal 37-19
interactive script 29-22
interest management 6-8, 6-9
  configuring 6-9
Interest Management Settings page 6-9
interface, Lua 29-3
intermediate contour line 48-9
intersection, KD trees 5-4
interval
  contour line 48-10
  spawn 20-6, 20-8, 20-12
  timeout 5-7
intervisibility 41-2
  enabling and disabling 41-10
  entity 41-3, 41-4
  fan 41-3, 41-4
    moving 41-8
  grid 41-3
  line 33-3
    hiding 41-10
    information 41-11
  object
    configuring 41-9
    deleting 41-13
    displaying 41-7
    editing 41-8
    transient and permanent 41-5
  pinning object to simulation object 41-10
  point-to-point 41-3
  terrain profile 41-12
  terrain profile line 49-4
  types 41-3
Intervisibility Settings page 41-6, 41-7, 41-9, 41-10, 41-11
invert pattern 32-7
Invert Selection check box 32-8
inverting condition parameter 32-7
IPV6 5-7, A-8
  DIS 5-12
Issue Plan, command 32-18
itemname, attribute 75-2, 75-6
IVE 51-5

J

Jam Targets dialog box 30-25
Jam Targets set data request 30-25
jammer
  activating 27-4
  deactivating 27-19
jamming
  communications 22-16, 22-17
  radar 22-16, 22-18
jamming-defense-factor parameter 22-18
jam-strength-factor parameter 22-19
jitter, camera 44-25
joining exercise, late 3-7
joystick 19-19
  configuration
    creating 72-3
    favorite 72-11
  configuring 72-2
  control, mapping 72-4
  controller 72-2
  disabling 19-21
  keyboard contorl 19-21
Joystick button 72-4
Joystick Configuration page 72-3, 72-4, 72-6
JPEG 47-4
JPG 47-4
JRM Technologies 40-2
JSON, configuration file 59-16
jumping 30-31
  observer 44-16

K

KD tree 5-4
key map 76-2
  creating 76-9
  deleting 76-9
  editing 76-4
key mapping 44-11
  adding 76-5
  binary 76-4
  changing 76-7
  combining 76-8
  deleting 76-7
Key Mapping button 72-6
Key Mapping Editor 76-2, 76-3
  adding mapping 76-5
  binary mapping 76-4
  changing mapping 76-7

Key Mapping Editor (continued)
deleting mapping 76-7
filtering function list 76-9
Key Mapping Editor page 76-4, 76-9
key value pair, reader writer 66-26
keyboard 76-2
attaching to objects from 45-3
changing speed 44-28
control 74-6
mapping 72-6
controlling entities 19-21
creating selection group from 17-10
entity control, configuring 72-2
inset views 46-2
mappings 44-6, 44-10, 76-3
movement functions 44-6
moving observer 44-10
navigation 76-2
key-names, location of configuration files 70-19
keyword
channel 73-8
largeCompass 44-5
smallCompass 44-5
tactical graphic property 64-45
KilometersPerHour updater 59-12
kneeling 30-31
KnotsPerHour updater 59-12

L

-L command-line option 5-4, 5-9, 7-18
-l command-line option 5-4, 54-4
label 15-2
2D
background 18-17
displaying 18-15
showing and hiding 18-16
entity 18-15
2D 18-12
3D 18-10
enabling and disabling 18-14
extended 33-3
object 15-4, 18-15, 33-3, 33-8
overlay 34-4
pinning to display 18-14
simulation object 15-4, 18-15
extended 18-18
spot report 9-9
tactical graphics 33-3, 33-8
Lambert Conformal Conic, coordinate system 44-7
lamppost, prop type 50-30
land attack, missile 27-6
landing
fixed-wing entity 25-25, 25-28, 26-26
rotary-wing entity 26-54
language
scripted task 28-23
specifying 5-4
for GUI 5-15
largeCompass, keyword 44-5
Lase Autonomous dialog box 30-25
Lase Autonomous set data request 9-16, 30-25
Lase Target task 9-16, 26-32
laser
autonomous 30-25
bomb 26-53
code 30-26
range 9-15
synchronizing between entities 9-16, 30-42
using 9-15
designator, displaying 37-16
targeting 26-32
Laser Code dialog box 30-26
Laser Code set data request 9-15, 9-16, 30-26
lasing, autonomous 9-16
Last Clicked Location Panel 48-17
late joiner 3-7
lateral-clearance parameter 70-15
latitude 18-11, 29-18
Launch Anti-Submarine Missile (Vertical) task 26-32
Launch Counter Measures, task 9-19
Launch Smoke task 26-33
Launcher, VR-Forces 4-4
launching
counter measures 9-19
torpedo 26-34
layer
earth file, hiding 48-13
feature 52-19
adding 47-4, 50-11
adding prop from 50-26
feature data 55-7
name 52-8
shapefile 52-8
ocean 50-13
adding 50-16

layout
formation
automatic 65-13
manual 65-13
GUI 5-4, 63-4
LCC, coordinate system 44-7
leader-promotion-id parameter 70-9
length
intervisibility object 41-8
simulation object name 15-3
track history 37-6
LettersNumbers.ctxc 52-23
level, notification for messages 5-15
level observer 44-7
level-of-detail 44-18, 44-19, 59-18
See also LOD
library, dynamic 4-33
license
3D models 1-22
DI-Guy 1-18
GL Studio 1-21
OpenSceneGraph 1-23
osgEarth 1-23
SilverLining 1-20
SpeedTree 1-22
License Manager 2-6
license server, hostname 2-7
License Setup dialog box 2-7
lifeform 31-12
appearance, random 64-15
configuring DI-Guy appearance and character 67-10
setting, posture 30-31
Lifeform Surrendered condition 31-12
lifetime
radio communication line 37-21
squawk indicator and communication line 37-21
light, navigation 52-25
light characteristic attribute 52-28
light_layer_list.csv 52-25
lighting
disabling 38-13
dynamic 38-2, 38-6
effects 1-15, 1-16, 50-20
Lighting Render Settings page 38-4, 38-5, 38-6
limitation, terrain 50-4
limited munition attack 22-7
line
arrow, editing 35-17
line (continued)
changing width 35-17
creating
displaying terrain profile 35-6
opening terrain profile 49-3
direction, changing 35-16
dotted and dashed 35-17
editing 35-17
opening terrain profile 49-3
fire 37-9
fortified, constructing 27-18
freehand 35-2
height-above-terrain 18-24, 33-5
intervisibility 33-3
information 41-11
radio communication 37-19
sensor contact 37-15
style 35-17
supply 37-17
suppressive fire at 26-50, 26-51
task visualization 37-12
terrain profile 33-3, 49-4
for intervisibility 41-12
vertex, showing in terrain profile 49-3
linear feature, list of 48-15
linear obstacle 33-2, 33-5
linear scaling 19-24
line-of-sight
See also intervisibility
communication line 37-21
terrain profile 41-12
unit versus entity 22-8
Linkage section in meta file 59-16
linked event, adding 8-6
lisp B-2
list
Categories 64-8
props 50-28
spot report 9-9
Used By Countries 64-8
vector feature 48-15
listing, components in OPD 68-16
LITCHR attribute 52-28
LittlePond terrain database 48-18
load balancing 7-19
Load Batch File dialog box 7-40
Load Overlays dialog box 34-6
Load Scenario dialog box 7-17, 7-19
load-entry parameter 70-16
load-entry-point parameter 70-16

Loader Settings page 50-35
loading
  database 47-7
  display engine configuration 5-3, 73-6
  DTED 50-4
  entities on entities 19-7
  entity display settings 5-3
  environment configuration 5-3
  MetaFlight 51-5
  models 50-31
  multiple DTED files 52-6
  object type mapping 5-3
  plug-ins 4-33, 4-34, 4-35, 5-3, 5-5, 5-8, 5-16
  preventing 5-5
  saved views 44-32
  scenario 5-5, 5-9, 7-17
    at startup 7-18
    recently used 7-18
  settings 4-23
  tactical graphics 34-6
  terrain 4-21, 50-31
    from command line 5-5
    paged 5-9
loadPluginIfVersionMismatch parameter A-4
local
  object 3-9, 68-9
    network interface 68-12
  simulation object 3-9
Local Weather Zones page 11-12
Local World Procedural Terrain database 48-18
locale, specifying 5-4
localization, specifying 5-4
localizing, GUI 2-11
local-objects parameter 68-9, B-4
localSettingsDesignator parameter A-6
location
  compass 44-5
  fixed-wing at creation 25-22
  jumping to 44-16
  moving to 26-39
  new entity 19-3
  object, changing 19-5, 35-13
  observer 44-3
  ordering entity to 26-39
  ordering simulation object to 26-39
  setting 16-3, 30-26
    by dragging 16-19
  suppressive fire at 26-50
  window 73-7
Location dialog box 30-26
location image attribute 50-9
Location set data request 30-26
Location3D class 29-18
locking
  GUI element 75-7
  overlay 34-4
LOD 44-18, 44-19, 59-18, 59-22
  distance, buoyancy 39-6
  ocean elevation 50-16
  paged, scale factor 44-20
  scale, setting 38-8
log file 4-40, 5-9, 5-10
  frame rate statistics A-4
  messages sent to 5-15
  sending entity messages 18-35
  specifying 5-4
  vrfSim 5-9
logFrameRateStatistics parameter A-5
Logger
  recording to 7-42
  recording view control messages 32-19
Logger Control message 7-41
logger-files-path parameter 7-38
logical, operator 31-14
logistics 22-14
  resupply, starting or stopping 30-41
  resupply modes 30-35
longitude 18-11, 29-18
Lower Periscope task 26-35
Lua 28-3, 28-5
  background process 29-29
  classes 29-3
  direction vector 29-18
  interface 29-3
  locations and vectors 29-18
  method 29-3
  module 29-6
  object 29-3
    location of 29-18
  parameter 29-17
    as table 29-15
  require 29-6
  script, extended name 28-12
  table 29-15
  tasks and subtasks 29-15
  utility functions 29-6
  vector
    geocentric 29-20
    offset 29-20
Lua API Documentation 29-3, 29-15

lubricant 22-14
  setting 30-19
lubricants, pacing and tracking 30-20
LWO file 47-3

M

-m command-line option 54-4
mainmenu, element 75-2
mainmenuconfiguration, element 75-2
MÄK Encrypted Data Format 54-6
MÄK RTI, configuring 2-10
MÄK Technologies LISP A-9
MAK_EARTH_FILE attribute 55-4
MAK_GEOID_GRID, environment variable 50-4
MAK_LAYER attribute 55-4
MAK_SOURCE_FILE attribute 55-4
MAK_VRF_ENABLE_DEBUG_DRAWER, environment variable 51-8, 51-9
making decisions 15-8
Makland, database 48-18
MAKLMGRD_LICENSE_FILE, environment variable 2-8
Manage Reactive Task dialog box 25-16
Manage Reactive Tasks dialog box 20-22
managing, plug-ins 4-33
manual
  aggregation 23-11
  aggregation and disaggregation 23-12
  conventions liii
  location of 2-4
map
  cube, recalculating 38-12
  texture 1-16, 50-20
map datum, configuring A-6
Map Scale Toolbar 48-18
mapping
  joystick controls 72-4
  joystick keyboard controls 72-6
  keyboard 44-11
  keys 76-3
  model
    to object type 58-3
    to remote observer 43-12
  movement to keyboard 44-10
  object type 58-3
  shapefiles to features 50-11
  soil types 50-36
  sound effects 42-4
mapping (continued)
  sound file to entity 42-5
marine
  conditions, setting 11-16
  effects, enabling and disabling 11-16
marker, system 64-24
Marker Labels, check box 64-24
marker-scale attribute 59-20
marking, remote simulation object 31-17
marking text 18-11, 64-6
mask
  adding to earth file 52-13
  element 52-13
matching object type 68-6
material, ambient value 38-4
Max Sheaf Radius 27-25
max-detonation-distance-for-passing-rounds parameter 70-17
maxDrainInputReads parameter A-7
maxDrainInputTime parameter A-5
maximum sheaf parameter 66-12
max-range parameter 26-53
max-range-altitude parameter 26-53
max-slope parameter 25-27
max-speed parameter 30-29
max-thrust parameter 25-27
mcastTtl parameter A-7
MEDF 54-6
  creating 59-23
  file 47-3
medfTool 54-6
media, adding to scenario event 8-5
MEIF, creating 59-23
memory 6-4
  improving use of 58-7
menu
  configuring 75-2
  context-sensitive 74-6
  locking 75-7
  popup 75-5
  separator 75-2
  Settings 4-23
  Task 28-4
  tearing off 74-5
menuitem, element 75-5
menuname, attribute 75-2
merging, scenarios 7-13
message 5-9
  console 5-9

message (continued)
entity 18-32
saving 18-35
frequency of A-4
information and warning A-4
notification level 5-5
notification of 18-36
object, viewing 18-34
object console, sending 32-16
radio 26-55, 26-56
receiving 70-13
sending 70-13
between front-end and back-end 3-3
text 26-57
sent 15-8
session, configuring 4-17
setting levels 5-15
Start/Resume 7-42
state update 43-11
Stop/Freeze 7-42
time stamp ordered, TSO 4-19
view control 44-32
message-timeout parameter 13-3
meta extension 59-16
meta file, configuring for windsocks and flags 59-16
metadata
in system definition file 68-16
script 28-5
metaFileName parameter 59-16
MetaFlight
building efficient terrains 51-12
loading 51-5
processing 54-2
method, Lua 29-3
mftTool 54-3, 54-5
middle-click menu 74-6
migrating
object parameter database to new release 68-27
SMS 63-12, 63-13
MilesPerHour updater 59-12
Military Scenario Definition Language, importing 7-14
Military Symbol Icon, visual definition 59-2
MIL-STD 2525B color 19-27
unit 24-11
Mimic Location Track mode 45-9
Mimic mode 44-8, 45-6, 45-8
Mimic Track mode 45-9
mimModule parameter A-6
minDrainInputTime parameter A-6
mine
dropping 26-16, 26-17
naval, arming 26-6
sweeping for 26-60
minefield, creating 27-18
minimum lift speed 25-27
minimum tick period 6-12
minimum tick period variance 6-12
min-tick-period parameter 6-12
min-tick-period-variance parameter 6-12
mirror image attribute 50-8
missile
anti-air 27-5
anti-ship 27-5
anti-submarine 26-32
ballistic 10-8
firing 10-8
target event 10-10
cruise 26-24
land attack 27-6
Missile Approach Warning condition 31-12
missile attack 22-7
Missile Target, page 10-8, 10-10
missile target event 10-10
missile-entity-state-repository 68-12
missile-param parameter 68-12, B-8
Mission Oriented Protective Posture. See MOPP
mobility, modifier 22-9
mobility-kill parameter 70-6
mode
Absolute 45-7
batch 7-37
C and 5 in IFF 30-24
Follow 45-7
Mimic 45-8
Mimic Track 45-9
observer 43-2
Space Follow 45-10
specifying 5-19
speed scaling 44-28
Tether 45-11
Tether Track 45-12
Track 45-13
view 44-8, 45-6
model
3D 16-9, 56-4
changing 64-17
DI-Guy 26-14
exaggerating entity 19-24

model (continued)
3D 16-9, 56-4
hiding 19-27
adding
as prop 59-13
tutorial 62-2
visual 64-20
aggregate-level, new 66-3
animation 61-11
calculating support point from 64-11
colorized, changing 64-17
communication A-8
compressing 59-23
creating 59-22
decal effects 37-4
definition, buoy 52-20
DI-Guy, choosing 58-6, 67-13
editing model mapping 58-4
element 52-6, 52-7, 59-20
entity 63-5
editing 63-2, 64-3
file caching 50-31
flipping DDS textures 59-17
illumination 11-4
instance, caching 58-7
instancing 5-4
layer, filtering 48-13
mapping
adding 58-3
common buttons 74-7
deleting 58-6
editing 58-4
to remote observer 43-12
name 18-11
object 63-5
physical 15-8
schema 57-8
set 43-11, 57-16
changing 43-11
showing 64-21
sharing resources 58-7
simulation 63-5
stealth 43-12
tactical graphic 64-46
topological 3-12
trailing effects 37-4
types 56-4
vehicle 1-7
visual, editing 64-16
model definition
applying updated 57-12
buoy 52-19, 52-20
building from attributes 52-20
texture atlas 52-23
compass 44-5
copying 57-11
creating 57-8, 57-9, 62-2, 67-3
cockpit display 59-11
DefaultEllipsoidArc 60-2
deleting 57-10
DI-Guy character 57-9
editing 57-12
filtering list of 57-11
parameter
adding 57-12
deleting 57-14
editing 57-13
prop, changing 50-30, 56-24
saving 57-14
schema 57-3
SpeedTree 57-9
tactical graphic texture 64-47
wake 11-18
Model Definition Editor page 57-8, 57-9, 57-10, 57-11, 57-12, 57-13, 57-14, 59-9, 59-17, 60-4, 60-5, 60-6, 62-2, 67-3
model element 52-18
model instancing 58-7
cache, clearing 58-7
model set 43-11
Model Specification for DIS Interoperability 61-2
modeldefinitionschema, attribute 59-6
modeling
aggregate-level 15-9, 20-1
entity-level 15-9, 20-1
modifier 22-3
combat engineering object 22-9
mobility 22-9
visibility 22-10
vulnerability 22-10
modifiers 66-17
modifying
control object 35-12
sensor signature 35-9
module
exectool 29-6
fsm 29-6
location of 29-6
Lua 29-6

monitor, multiple 73-18
monitoring, entity resources 18-30
MOPP level 22-8
  changing 27-9
  setting 30-27
MOPP Level dialog box 30-27
morale, setting 30-27
Morale dialog box 30-27
motion, file 67-11
mouse
  changing speed 44-28
  drag terrain 44-14
  drag view 44-15
  locking to object 16-18
  moving observer 44-14, 44-15, 44-18
  orbiting terrain 44-15, 44-18
  teleporting observer 44-16
Mouse Mappings page 44-14, 44-18
Move Along Route dialog box 26-36
Move Along Route Retrograde task 27-26
Move Along Route task 26-36
Move Into Formation dialog box 26-37
Move Into Formation task 26-37
Move to Altitude dialog box 26-38
Move to Altitude task 26-38
Move to Depth dialog box 26-38
Move to Depth task 26-38
Move to Location (Plan Along Roads) task 25-21, 26-40
Move to Location dialog box 26-39
Move to Location Retrograde task 27-26
Move to Location task 26-39
  in plan 26-40
Move to Waypoint (Plan Along Roads) task 26-43
Move to Waypoint dialog box 26-42
Move to Waypoint Retrograde task 27-26
Move to Waypoint task 25-21, 26-42
movement
  DI-Guy 26-14
  lifeform 30-31
  model 64-22
    specifying 64-24
  retreat 27-26
  scripted 26-5
  speed, affect of posture 22-5
  speed scaling 44-27
  stopping 27-22
  system 64-23, 68-16
  task, affect of aggregation and disaggregation 23-10
movement (continued)
  unit 21-4, 22-6
    restriction 55-9
movement file, configuring 64-30
movement parameter 66-19
movement system
  properties, editing 66-20
moving
  along a route 26-36
  camera 44-13
  in formation 31-16
  intervisibility fan 41-8
  object 16-19
  objects 16-19, 30-26
    between overlays 35-5
  observer 44-3, 44-12
    mouse with 44-14, 44-15, 44-18
  options 44-12
  to altitude 26-38
  to location 26-39
  to points or simulation objects 26-42
  vertex 35-14
moving object parameter B-8
moving-object-param parameter 68-12, B-8
moving-object-param parameter type B-8
MSDL, importing 7-14, 7-16, A-5
MTD file 47-3
MTL file 47-3, A-9
  vrfSim.mtl A-3
MTP file 47-3
multicast address, configuring A-7
multicast mode, specifying 5-19
multicast time-to-live 5-19, A-7
multichannel, configuring 73-18
MultiGen-Paradigm, OpenFlight format 61-2
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 18-10
  parameter values 57-18
  simulation model set 7-8
  VR-Forces processes 3-2
multiple entity creation 20-17
  specifying entity type 20-20
Multiple Object Information window 18-10
munition
  configuring entity for damage by 70-7
  damaging entity 9-22

munition (continued)
display of path and detonation 37-9
Munition Target Settings dialog box 10-3, 10-6, 10-8, 10-10
munition-type parameter 70-3
mutually exclusive task 24-1, 25-4, 31-7
configuring 70-14
mutually-exclusive-task-groups parameter 70-14
muzzle flashes 37-8

N

-N command-line option 5-6, 5-11
-n command-line option 5-5, 5-9, 5-15
name 15-2
changing
simulation object 64-6
tactical graphic 19-5
changing overlay 34-5
component 68-13
control object 33-2
default 16-11
displaying 18-12
entity 18-10, 18-11
extended 28-12
feature layer, specifying 55-2
FED file A-6
federate A-6
in condition 32-7, 32-8
layer 52-8
object 64-6
setting 16-3
short 64-6
simulation object 15-3, 18-10, 64-6
spawn pattern 20-10
named query 55-3
Naval Depth Charge Deployment, system 26-14, 26-15
Naval Mine Deployment, system 26-16, 26-17
navigating
unit hierarchy 65-7
view 44-3
navigation
advanced 53-2
attached context 44-8
function, editing key map 76-4
key map 76-2
keyboard 76-2
light 52-25
streaming 52-25
navigation (continued)
options to 44-12
speed scaling 44-27
navigation area
cancelling generation of navigation data 53-7
configuration file 53-10
creating 53-3
displaying 53-5
editing 53-4
reverting 53-5
generating navigation data for 53-6
importing 53-10
pedestrian path 20-22, 33-2
removing 53-5
status 53-7
Navigation Areas page 53-3
navigation data 3-12, 53-2
displaying 53-11
generating 53-5, 53-6
canceling 53-8
removing from queue 53-7
generating for entity 53-8
importing 53-10
profiles 53-6
using 53-8
Navigation Lab 53-11
viewing simulation objects in 53-14
Navigation Preferences set data request 30-28
navigational aid light 52-28
navigation-preference parameter 30-28
NBC
area 22-7, 22-8
report 27-28
sending 27-28
near clipping plane 73-10
Near Far Clip Policy, attribute 73-11
net-interface-min-tick-period parameter 6-13, B-5, B-7
net-interface-min-tick-period-variance parameter 6-13, B-5, B-7
network
connecting to, automatically 5-3
vector 25-21
network interface B-5
hierarchy 68-12
network-interface parameter B-5, B-7
new
spawn pattern 20-10
terrain 50-3
window 4-14
New Scenario dialog box 7-35, 12-3

night vision 40-2
night-illumination parameter 71-13
no dialog box 5-14
node, switch 59-17
noise 40-4
  propulsion, RPM 9-18
non-VR-Forces, simulation object 31-17
normal map 1-16, 50-20
notation, prefix 32-5
notification, icon 18-36
notification level 5-9, A-4
  setting 5-15, 30-28
  setting for entity console 18-33
  specifying 5-5
Notify Level dialog box 30-28
Notify Level set data request 30-28
Nuclear Attack dialog box 27-27
Nuclear Attack task 27-27
nuclear contamination 22-7, 22-8, 27-27
numbering, events 8-13
number-of-runs parameter 7-39
NVG 40-2

O

-O command-line option 5-5
oberver, camera shake 44-25
OBJ 56-4
  file 47-3
object
  adding
    resources in OPD Editor 69-11
    to Props Palette 16-26, 57-8
    to selection group 17-9, 17-10
  altitude, changing 19-5, 35-13
  attaching observer to 45-2
  category, changing 64-3
  changing
    line width 35-17
    name 19-5
  combat engineering 22-9
  concepts 3-8
  configuring, avoidance of 70-15
  control 33-2
  copying 16-21
  count 6-4, 18-37
  creating 16-3, 16-4, 16-5, 16-8, 16-9, 16-11
    in plan 32-17
  creation, mouse locking 16-18
  cultural feature 16-25, 64-8
object (continued)
  deleting 64-42
    from plan 32-17
  deleting from simulation model set 64-42
  detaching observer from 45-5
  disembarking 30-16
  dragging 16-19
    to new location 30-26
  editing 35-12
  embarking in Embarkation View 19-10
  enabling and disabling creation of 64-6
  entity enumeration, changing 64-4
  exaggerating models 19-24
  fill style 35-17
  filtering list of 17-6
  geometric 29-18
  heading, changing 19-5, 35-13
  hiding 9-2, 34-4
  ID 15-2, 15-4
  intelligence 8-8
  intervisibility 41-5
    pinning to simulation object 41-10
  label 15-4, 18-15, 33-3, 33-8
  list 12-10
  local 3-9
  location, changing 19-5, 35-13
  locking mouse to 16-18
  Lua 29-3
  messages, viewing 18-34
  model 56-4, 63-5
  moving 16-19
  moving to 26-42
  multiple, information 18-10
  name 64-6
  object type, changing 64-4
  orbiting 44-13
  overlay 33-2
  palette 16-4
    specfiying 64-7
  parameter B-3
    default value 12-10
  parameters 68-3
    editing 64-3
  pasting 16-21, 16-22, 16-23
    altitude 16-24
  placing 16-11
  preprocessing A-4
  properties
    setting 16-3
    specifying 16-13

object (continued)
remote 3-9
selecting 17-4, 64-3
for creation 16-4
in Objects List Panel 17-6
showing and hiding 33-7
subcomponents 68-9
super-type 68-5
timeout threshold A-7
type 68-4
unselecting 17-7
vertex
adding 35-13
deleting 35-15
editing 35-14
views of 17-2
VR-Forces 3-9
object console 18-32
clearing 18-36
notification of messages 18-36
setting notify level 18-33
Object Console Summary Panel 17-4, 18-34, 29-22, 44-25
answering questions 18-35
selecting simulation object 17-6
object geometry, adding 64-13
Object Group category 16-25
object map file 6-1, 12-2, 12-9
missing back-ends 3-8
object parameter database 3-8, 63-5, 68-2
adding, component in OPD Editor 69-8
component list order 68-16
configuring, sensors 71-2
configuring path to A-4
loading in OPD Editor 69-3
migrating to new release 68-27
pathname interpretation 68-26
reloading A-4
saving 69-12
system definition file 68-16
Object Parameter Database Editor. *See* OPD Editor
object type
mapping 58-3
loading 5-3
published and matching 68-6
object type enumeration B-9
objectConsoleNotifyLevel parameter 18-33
objectConsoleNotifyLevel parameter A-5
Object-Map parameter 12-5
objects 16-9, 16-11
Objects Console Summary Panel 18-9
Objects List Panel 17-2, 17-4, 17-6, 19-7, 19-10, 23-6, 24-4
Embarkation View 19-7
selecting object in 17-6
object-type parameter B-3
OBJNAM, attribute 52-22
objTypeEnums.h header file B-9
observer 44-3
adding 43-6, 43-7
altitude, filtering simulation objects by 6-9
attaching
articulated part 45-4
to simulation object or prop 45-2
to simulation objects 45-6
attachment
in 2D 45-14
order 45-3
camera 44-13
changing 73-20
centering on entity 44-25
compass 44-4
coordinate system
2D 44-7
3D 44-7
detaching from object 45-5
dragging
terrain 44-14
view 44-15
heading 44-4
inset view, in 46-2
location and orientation 44-3
LOD, ocean 50-16
mode 43-2
changing 43-3
creating 43-4
deleting 43-5
editing 43-5
renaming 43-5
setting 4-25
switching projection 43-2
view control message 32-19
XR 43-10
movement 45-6
functions 44-6, 76-2
speed 44-28
moving
keyboard with 44-10
mouse with 44-14, 44-18

observer (continued)
orbiting 44-13
orientation 44-15
rain and wave effects 39-2
range for sound effects 42-3
removing 43-9
saving view 44-29
teleporting 44-16
view
attachment change 45-3
saving 44-29
transitioning between 44-30
visibility 11-5
ways to move 44-12
zooming 44-18
2D 44-20
3D 44-19
entity extents 44-21
zooming to extents 44-20
Observer Control Panel 44-27
Observer Information Panel 44-3
Observer Name 73-8
Observer Settings page 18-27, 18-31, 18-32, 19-26, 19-27, 37-4, 37-5, 37-14, 38-7, 39-3, 40-4, 43-4, 43-5, 43-6, 43-9, 44-5, 44-23, 44-25, 44-26, 44-28, 46-10, 48-7
Observer Views Panel 44-29
loading saved views 44-32
saving views 44-31
obstacle 33-2, 33-5
avoidance 19-17
breaching 27-9
configuring avoidance of 70-15
destroying 27-21
fill style 35-17
improving 27-24
linear 33-2, 33-5
report 27-28
sending 27-28
obstructed terrain, viewing in terrain profile 41-12
obstruction, configuring avoidance of 70-15
occupancy-director-controller, role in embarkation 70-16
ocean 1-21
choppiness 11-17
depth 50-19
dynamic, enabling and disabling 11-16
effect 11-15
height 50-15, 50-16
ocean (continued)
height map
regenerating 50-19
resolution 50-17
texture size 50-17, 50-18
layer 50-13
adding 50-15, 50-16
planar reflection 38-7
configuring 38-8
performance 38-8
texture height 38-8
texture width 38-8
quality 11-19
swell 11-17
synchronized 5-5
Ocean Height Map Regeneration Threshold 50-19
ocean height map resolution 50-18
Ocean LOD Elevation 50-16, 73-17
Ocean Render Settings page 11-18, 38-8, 38-12, 39-4, 50-18, 50-19
off road driving 25-21
offset, burst 20-8
offset start time 20-12
OffsetX parameter 59-12
OffsetY parameter 59-12
OffsetZ parameter 59-12
OGR 55-7
driver 52-8
ogr2ogr 55-7
oil 22-14
pacing and tracking 30-20
setting 30-19
opacity
2D label background 18-17
prop 50-29
setting 50-29
OPD, reloading A-4
OPD Editor 1-19
adding, resources 69-11
changing, component priority 69-9
compared to Simulation Object Editor 63-2
connecting components 69-10
deleting parameters and components 69-7
editing, parameters 69-4
loading object parameter database 69-3
opening from Simulation Object Editor 64-25
restoring values 69-7
saving object parameter database 69-12
starting 69-2
OPE file 68-26

Open Model Set dialog box 63-7, 63-15
OpenAL 42-3
OpenFlight 56-4
  converting texture data 50-36
  file 47-3
  format 1-7, 61-2
OpenGL 50-34
opening
  Create Object dialog box 16-13
  inset view 46-2
  simulation model set 63-15
  terrain 47-7
OpenSceneGraph 1-3, 1-20
  environment variables for anaglyphic stereo 73-22
  license 1-23
operator
  boolean 31-14
  comparison 31-13
  logical 31-14
optimizing, performance 6-4
option
  command-line 5-3
  element 52-17
  navigation 44-12
orbit, observer 44-13
Orbit task 26-45
orbiting
  terrain, with mouse 44-15, 44-18
order of battle 6-1, 12-10
  file 12-2, 12-10, 34-6
    default values in 12-10
ordered speed, setting 30-29
Ordered Speed dialog box 30-29
Ordered Speed set data request 30-29
  fixed-wing for 30-29
Order-Of-Battle parameter 12-5
ordnance
  unexploded, destroying 27-20
organization
  entity of 15-5
  unit 21-3
orientation 44-3
  camera 73-20
  clamping 19-23
  displaying 18-12
  icon 18-23
  observer 44-3, 44-14, 44-15, 44-18, 45-6
  prop, changing 50-30
  specifying measurement unit 74-9
OSG file 47-3
OSG_CURL_PROXY, environment variable 51-5
OSG_CURL_PROXYPORT, environment variable 51-5
OSG_ICO, environment variable 6-11
osgEarth 1-20, 51-2, 52-3
  adding terrain servers 51-4
  license 1-23
osgEarth Boundary Generation Tool 52-13, 52-15
osgearth_cache 1-19, 50-32, 52-16
OSGEARTH_CACHE_PATH, environment variable 50-32
OSGEARTH_CURL_PROXYAUTH, environment variable 51-5
outline
  emitter volume segment 60-5
  segment 60-5
output, port group 68-14
Out-the-Window, observer mode 4-25
out-the-window view 73-18
overlay 1-14
  changing name 34-5
  concepts 33-2
  creating 34-3
  default
    adding 64-7
    specifying 64-7
  deleting 34-5
  editing, enabling and disabling 34-4
  file 12-2, 34-6, 35-7
  hiding 34-4
    by default 34-4
  hiding graphics on 33-7
  introduction to 34-2
  label 34-4
  locking and unlocking 34-4
  moving objects between 35-5
  object 33-2
    selecting 17-4
  setting 16-3
  tab 34-4
  text, adding 35-4
Overlay Layers dialog box 64-7
overlay object
  copying 16-21
  deleting 35-11
  disaggregation area 23-12
  moving, to different overlay 35-5
  pasting 16-21, 16-22, 16-23

overlay object (continued)
spot report 9-9
Overlay parameter 12-5
Overlays tab 33-7
Overlays view 17-2
Override Position 19-9
overriding
default scenario parameters 7-5
plan 32-24

P

-P command-line option 5-7, 5-12, 5-19
-p command-line option 5-7, 5-11, 54-4
pacing
ammunition 30-8
equipment 30-19
resource 30-34
weapon 30-44
package, scripted task 28-30
page
Add Props 56-20
Audio Settings 42-2, 42-3
Checkpoint Settings 7-26, 7-32, 7-33, 7-34
Date and Time 11-3, 11-4
Detonation Sound Editor 42-8
dialog box, configuring 75-3
Display Units Settings 18-24, 33-9
Earth Layers 48-13
Edit Existing Props 50-26
Elevation Color 48-4, 48-5, 48-6
Engineering Object Information 22-12
Entity Behavior Options 23-6, 23-11
Entity Definition Editor 59-2, 59-14, 60-6, 62-5
Entity Mapping Settings 62-8
Entity Sound Editor 42-5, 42-6, 42-7
Environment Conditions Settings 59-16
Feature Settings 48-16
Fire Sound Editor 42-8
Indirect Fire 10-3, 10-6
Joystick Configuration 72-3, 72-4, 72-6
Local Weather Zones 11-12
Missile Target 10-8, 10-10
Model Definition Editor 59-9, 59-17, 60-4, 60-5, 60-6
Mouse Mappings 44-18
Navigation Areas 53-3
Observer Settings 40-4, 43-6
Raster Maps 50-21
page (continued)
Render Settings 44-24
Resource Information 18-30
Scenario Event Settings 8-14
Schema Editor 57-9
Sensor Settings 40-4
Session Options 18-19, 18-20, 74-8
Shadow Settings 38-15, 38-16
Supplies 22-12
Symbol Decoration Settings 18-17
Tactical Graphics Settings 35-18
Terrain Contents 50-15
Weather 11-6, 11-7, 11-8, 11-9, 11-10, 11-11, 11-16, 73-17
pagecollection, element 75-3
pagecollections, element 75-3
Paged LOD Magnification Scale Factor 44-20
paged terrain
flipping DDS textures 50-35
loading 5-9
preprocessing 51-5
paging
terrain 47-8
manually 51-6
palette
Entity 16-3
Hazards/Obstacles 11-13, 16-4, 16-5, 22-9
object creation 16-4
specifying 64-7
Props 16-3
Tactical Graphics 16-5
panel
displaying and hiding 74-3
dockable 74-2
docking 74-2
floating 74-2
Frame Rate Monitor 6-5
Last Clicked Location 48-17
Object Console Summary 18-34
undocking 74-2
parachuting 30-31
parameter
acceleration-to-rpm-factor 9-18
adding, to model definition 57-12
additional-opds 12-6
additionalSystemScriptsDirectory A-3
addMulticastAddr A-7
aggregate-formation-controller 70-8
aggregate-level entity 66-17
aggregate-level unit 66-4

parameter (continued)
aggregate-object-param 68-25
aggregate-param B-8
aggregateSpatialModelTickAsFastAsPossible-WhilePaused A-3
aggregateSpatialModelTickPeriod A-3
aggregateSpatialModelTickPeriodUsesRealTime A-3
aggregateSpatialModelTickVariance A-3
all-tasks-exclusive 70-14
ammo 70-3
angle-of-incidence 70-6
angleSegmentMax 60-5
appIdRange A-7
appNumber A-3
assertOnBlockingTerrainCalls 47-8, A-3
Attach Components to Remote Entities On 7-5
attack, editing 66-22
auto-reorganize 12-6
batchScenarioFileName A-3
can-be-radar-jammed 22-18
can-pivot 26-61
catastrophic-kill 70-6
cgfDispactherReceivePort A-3
ChannelKeyword 59-12
commenting out A-9
comm-model-name 70-13
Component Attachment Table 7-6
component list order 68-16
Component-Attachment 12-5
Component-Attachment-Table 12-5
component-manager B-5, B-7
countryCodesMappingFile A-3, A-5
create-script-id 28-20
cultural-feature-param 68-12, B-8
daemon A-3
datumShiftX A-4
datumShiftY A-4
datumShiftZ A-4
daylight-illumination 71-13
day-night-transition-time is 71-13
defaultHeartbeatThreshold A-7
defaultObjectTimeout A-7
defaultParameterDatabasePath 12-7, A-4
defaultTerrainDatabasePath 12-7, A-4
defense, editing 66-23
definitions 69-4
deleting
from model definition 57-14
in OPD Editor 69-7
destAddrString A-7
parameter (continued)
destroyFedExec A-6
destroy-script-id 28-20
deviceAddress A-7
diGuyAnimationsFile A-4
diGuyCharacterDataFile A-4
DIS A-7
disabled-by-suppression 70-18
disableParallelPreprocess A-4
Display Terrain 7-6
disPort A-7
doNotUseConsole A-4
dr-allow-gui-overrides 6-15
DtNotifyLevel A-4
dynamicFeaturesDisabled A-4
echelon-level 68-25
editing in OPD Editor 69-4
editing in Simulation Object Editor 64-11
egress-points 70-16
electronic warfare, editing 66-23
embarkation-slot-entry 70-16
embarkation-slots 70-16
enableDebugDrawer A-4
entity-type 70-4
entry B-2
entVrfDataTimeThreshold A-4
equipment, editing 66-24
execName A-6
exerciseId A-7
exercise-start-time 12-6
FeatureName 59-17
federateName A-6
federateType A-6
fedFileName A-6
Filename 59-11
firepower-kill 70-6
fixed-wing-entity-param B-8
fomMapperInitData A-6
fomMapperLib A-6
forceParameterDbReload A-4
Frame Mode 7-7
Frame Time 7-7
frame-mode 4-20, 12-6
time management 4-19
frameRateStatisticsLogFile A-4
frame-time 12-6
ground-vehicle-param 68-3
ground-vehicle-param B-8
GuiObserverViews 12-6
GuiScenarioSettings 12-6

parameter (continued)
Gui-Terrain-Database 12-5
heartbeatVariance A-7
HLA A-6
hostAddressString A-6, A-7
hostile-fire-only 70-17
human-param 68-12, B-8
improve-script-id 28-20
ingress-points 70-16
initial-road-preference 30-28
instrumentPanelCategory 46-7
jamming-defense-factor 22-18
jam-strength-factor 22-19
lateral-clearance 70-15
leader-promotion-id 70-9
load-entry 70-16
load-entry-point 70-16
loadPluginIfVersionMismatch A-4
local-objects 68-9, B-4
localSettingsDesignator A-6
logFrameRateStatistics A-5
logger-files-path 7-38
Lua, table 29-15
max-detonation-distance-for-passing-rounds 70-17
maxDrainInputReads A-7
maxDrainInputTime A-5
maximum sheaf 66-12
max-range 26-53
max-range-altitude 26-53
max-slope 25-27
max-speed 30-29
max-thrust 25-27
mcastTtl A-7
message-timeout 13-3
metaFileName 59-16
mimModule A-6
minDrainInputTime A-6
min-tick-period 6-12
min-tick-period-variance 6-12
missile-param B-8
mobility-kill 70-6
model definition, editing 57-12, 57-13
movement 66-19
moving-object-param B-8
mutually-exclusive-task-groups 70-14
navigation-preference 30-28
net-interface-min-tick-period 6-13, B-5, B-7
net-interface-min-tick-period-variance 6-13, B-5, B-7
network-interface B-5, B-7
night-illumination 71-13
parameter (continued)
number-of-runs 7-39
object 68-3, B-3
editing 64-3
objectConsoleNotifyLevel 18-33
objectConsoleNotifyLevel A-5
Object-Map 12-5
object-type B-3
OffsetX 59-12
OffsetY 59-12
OffsetZ 59-12
Order-Of-Battle 12-5
Overlay 12-5
parameter-type 68-3, 68-8, 68-12, 68-26, B-2, B-3
passing-rounds-suppress 70-17
personnel, editing 66-24
Perspective 59-11
PerspectiveHeight 59-12
PerspectiveWidth 59-12
physical 66-18
Plan 12-5
plan-manager B-5, B-7
pluginsDirectory A-5
position-offset 70-9
power-plant-active 9-18
promotion-id 70-9
providing information to components 15-7
publish-transmitter 70-13
publish-transmitters 70-13
publishZeroVelocityWhenPaused A-5
Radio Button Choice 28-14
Random Number Seed 7-7
random-number-seed 12-6
range-determinant 70-4, 70-6
range-list 70-4
recvBufferSize A-8
relative-heading 19-13
relative-position 19-13
remote-objects 68-9, B-6
rotary-wing-entity-param B-8
RotDir 59-17
RotOffsetDeg 59-17
rprFomVersion A-7
RSOName 59-11
RTI_timeMgmt 4-19
RTI_useRtiExec 4-19
run-duration 7-39
runInTimeManagementMode 4-20, A-7

parameter (continued)
scenario
file 12-4
setting 7-5
Scenario Name 7-5
scenario-data 12-6
ScenarioDescription 12-6
ScenarioExtentInformation 12-6
scenarioFileName A-5
scenario-filename 7-39
scenario-name 12-6
Scenario-Scripts 12-5
schema
adding 57-6
deleting 57-7
editing 57-7
scripted task 28-12
script-ids 28-20
SelectionGroups 12-5
sendBackendLogToNetwork A-5
sendBufferSize A-8
sendFedTime A-7
send-spot-reports-on-own-force 9-7
sensor, editing 66-21
separation-distance 25-10
separation-tolerance 25-10
sessionId A-5
setMainThreadToHighPriority A-5
setting simulation object 30-4
showLines 60-5
simMgmtPduProcessLevel A-5
Simulation Model Sets 7-6
Simulation Object 28-15
Simulation Terrain 7-5
simulationModelSet A-5
Simulation-Model-Set-Files 12-6
simulation-run-duration 7-39
siteId A-5
sms-filename 7-39
soil-list 19-18
specifying for task 25-6
speed-to-rpm-factor 9-18
spheroidSemiMajor A-6
spheroidSemiMinor A-6
startInRunMode A-6
state-repository B-4, B-6
state-repository-min-tick-period 6-13
state-repository-min-tick-period-variance 6-13
statusUpdateHeartbeat A-6
stopping-distance-factor 70-15
parameter (continued)
structure of entries B-2
subnetMask A-8
substitute-message-size 13-3
subsurface-entity-param B-8
sunrise-time 71-13
sunset-time 71-13
supplies, editing 66-24
suppression-degrades-sensor-performance-by 70-18
suppression-detonation-insult 70-17
suppression-detonation-range 70-17
suppression-insult-level 70-17
suppression-recovery-time 70-17
suppressSelfReflect A-8
surface-entity-param B-8
targetFrameRate A-6
targeting-capable 30-17
task, Lua 29-17
task-manager B-5, B-7
terrain-check 25-30
Terrain-Database 12-5
terrainDatabase A-6
Time Multiplier 7-6
time-multiplier 12-5
underFireDistance 30-36
underFireTime 30-36
useAbsoluteTimeStamps A-7, A-8
use-actual-message-size 13-3
useAdvisories A-7
useAsyncIO A-8
useCustomMapDatum A-6
useDISGuy A-6
useIpv6 A-8
use-logger-control 7-38
using remote entities as 31-17
value, restoring in OPD Editor 69-7
vel-to-align-actual 25-31
vel-to-align-desired 25-31
visual definition, editing 57-18
vrf-base-object-param 68-26, B-3
wakeLength 11-18
warhead 70-3
WindSpeed 59-17
WindState 59-17
parameter-type, control object 68-26
parameter-type parameter 68-3, 68-8, 68-12, 68-26, B-2, B-3
parent
disembarking from 19-11
element 57-17

parent (continued)
element definition 57-14
parent, attribute 59-6
particle effect, sea spray 39-4
passing-rounds-suppress parameter 70-17
passive sonar 9-17
paste
special, altitude 16-24
Paste Special 16-23
pasting
appearance 16-23
objects 16-21, 16-22, 16-23
performance and 16-21
plans 16-22
simulation object 16-23
patch, terrain 47-3
path
data 3-12
finding 3-14
following 3-15
planning 3-14, 26-40, 26-43
using road data 30-28
specifying for RTI 2-10
pathname
interpreting by object parameter database 68-26
scenario files 12-7
Patrol Between dialog box 26-46
Patrol Between task 26-46
Patrol Route dialog box 26-45
Patrol Route task 26-45
patrolling
between objects 26-46
route 26-45
pattern
color 52-20
editing condition 32-9
in condition 32-7, 32-9
Pattern Hold (Location) dialog box 26-46
Pattern Hold (Location) task 26-46
Pattern Hold (Waypoint) dialog box 26-47
Pattern Hold (Waypoint) task 26-47
Pattern tab 32-9
pause
scenario, using PDU 7-42
pause mode A-6
pausing
scenario, automatically 7-7, 7-35
simulation 7-25
PDU
sending 43-11
PDU (continued)
siman, configuring A-5
Start/Freeze 7-42
Start/Resume 7-42
Stop/Freeze 7-42
Underwater Acoustics 9-17
view control 44-32
pedestrian path 26-36
control object 20-22, 33-2
pen style 35-17
Percent Complete dialog box 30-29
Percent Complete set data request 30-29
performance 6-8, 6-9, 6-13, 44-19, 50-17
affected by copy and paste 16-21
anti-aliasing 5-3
asynchronous I/O 5-19, 6-11
attached observer 45-6
compressing texture 50-33
cube map calculation 38-12
factors affecting 6-3
frame rate 5-10, 6-17
function profiler 6-7
graphics, depth 5-3
HAT lines 18-24, 33-9
improving 58-7
instancing 5-4
interest management 6-8
load balancing 7-19
main thread 5-10
model, best practices 59-22
network, tuning 6-14
ocean planar reflections 38-8
optimizing 6-3, 6-4
setting application priority 5-4
shadows 38-16
statistics, displaying 6-6
stencil bits 5-5
streamed data 6-11
terrain 50-4
tuning
network interface 6-13
state repository 6-13
visual for 3D application 6-14
visual 6-19
Performance Options dialog box 6-14, 6-15
Performance Options page 6-14, 6-17
period, burst 20-8
periodic checkpointing
disabling 7-33
scheduling 7-32

periodic snapshot, enabling 7-34
periscope
  lowering 26-35
  raising 26-52
permanent intervisibility object 41-5
personnel, status 18-4
personnel parameter, editing 66-24
perspective, in display 73-15
Perspective parameter 59-11
PerspectiveHeight parameter 59-12
PerspectiveWidth parameter 59-12
phase line 33-2, 33-4
  Entity Left of Line 31-12
physical model 15-8
physical parameter 66-18
physicalWorldParams.mtl, file 71-9
pinning
  range rings 18-27
  sensor contact lines 37-14
  task visualization 37-13
pitch 44-3
Pitch updater 59-12
pixel 59-22
Place IED dialog box 26-48
Place IED task 26-48
placing 16-11
  IED 26-48
  new entity 19-3
  props 16-11
  simulation objects 16-11
  tactical graphics 16-11
plan
  abandoning 24-1, 32-24
    confirmation prompt 25-5
    from a plan 32-24
  adding
    condition 32-4
    statement 32-3
    task 32-3
  aircraft 31-17
  altitude condition 31-10
  assigning to multiple simulation objects 32-22
  concepts 30-1
  conditional statement 31-4
  considerations for creating 32-2
  console message, sending 32-16
  copying 32-23
  creating object 32-17
  custom 20-2, 20-9, 20-12
  default 20-2, 20-9, 20-12
plan (continued)
  deleting
    object 32-17
    simulation object 32-17
    statement 32-10
  editing 32-2
    statement in 32-10
  entity 31-2
    when aggregated 23-11
  file 6-1, 12-2
    editing 12-8
  Follow Entity task 31-16
  force hostility 9-12
  global 11-2, 31-2
    target simulation object 32-15
    writing 32-11
  global command, adding 32-15
  global task, view control 32-19
  including scenario events 8-11
  individual 11-2
  inverting condition parameter 32-7
  issuing 32-18
  manager B-5
  Move to Location task 26-40
  name in condition 32-8
  pasting 16-22, 16-23
  pattern in condition 32-9
  printing 32-10
  remote simulation object 31-17, 32-22
  restarting 32-23
  saving 32-11
  simulation time condition 31-13
  Tasked by Superior 31-16
  trigger 31-7
  unit, for 23-12, 32-22
  viewing 31-14
  When statement 31-7
  While condition 31-9
  writing 32-2
Plan parameter 12-5
plan view 43-10
Plan View window 4-14
Plan window 31-14, 32-2, 32-3, 32-4, 32-10, 32-11, 32-23
planar reflection 38-7
plan-manager parameter B-5, B-7
planning, path 3-14, 26-40, 26-43
plant, avoiding 19-17
platform
  regenerating 69-12

platform (continued)
update, applying 63-17
platform file
variable binding
adding 68-22
adding to Simulation Object Editor 68-23
variable bindings 68-21
platoon, 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, 4-35, 5-3, 5-5, 5-8, 5-16
managing 4-33
prevent loading 5-5
version, checking 5-3
XML file 4-36
deleting 4-39
Plug-ins dialog box 4-33
Plug-ins Editor 4-34, 4-36, 4-37, 4-38, 4-39
pluginsDirectory parameter A-5
PNG 47-4
point 33-4
civilian visit 20-22
feature, list of 48-15
moving to 26-42
orbiting around 44-13
patrolling between 26-46
reversing 35-16
strong, constructing 27-19
suppressive fire at 26-50, 26-51
terrain profile, sampling rate 49-3
point feature, in front-end and back-end 47-5
point-to-point communications
configuring A-7
specifying 5-19
point-to-point intervisibility 41-3
POL, status 18-4
polarized stereo 73-21
configuring 73-23
policy
Distance To Attached 73-11
Reduce Near Clip 73-11
Reduce Z Fighting 73-11
polygon 33-5, 47-7, 59-22
count 50-4
polygon (continued)
not going through 44-27
underwater 50-19
popup menu 74-6, 75-5
port 15-7, 68-14
default 5-19
for scenario files A-3
group 15-7, 68-14
number A-7
specifying at startup 5-19
proxy 51-5
specifying, at startup 5-19
position 44-3
camera 73-20
observer 44-14, 44-18, 45-6
prop, changing 50-30
window attribute 73-7
position-offset parameter 70-9
posture
adding 66-26
affect on footprint, sensing, sensor signature, speed 22-5
changing 27-10, 30-30
Deliberate-Attack 22-4
Deliberate-Defense 22-4
Hasty-Attack 22-4
Hasty-Defense 22-4
Reconnaissance 22-4
Rout 22-4
setting 30-31
transition time, editing 66-21
Travel 22-4
unit 22-4
Posture dialog box 30-30, 30-31
Posture set data request 30-30, 30-31
power, combat 22-3
power-plant-active parameter 9-18
precedence, components in OPD 68-16
precipitation 11-5, 11-8
intensity 11-11
type 11-11
preconfigured unit 23-5
Prefer Roads 30-28
preferred movement mode 26-63
preferred road data 26-63
Preferred Targets dialog box 30-32
prefix notation 32-5
preprocessing
objects A-4
terrain, paged 51-5

primary color, unit 24-11
primary simulation object 45-9, 45-12
printing
  plan 32-10
  window 4-14
priority
  component, changing 69-9
  weather zone 11-12
prisoner-of-war 30-41
probability table, direct and collateral damage 70-6
probability-coefficient list 70-4
process
  background 29-29
  object mappings to 12-9
  VR-Forces 3-2
process ID, front-end 5-8
processing
  MetaFlight 54-2
  streamed data 52-6
product
  technical support lii
  upgrades lii
profile, navigation data 53-6
profiler, function 6-7
projected terrain database 44-7
projection
  supported 1-14
  switching 2D to 3D 43-2
Projection Resize Policy 73-8
  configuring 73-12
Projection Units 73-8
promotion-id parameter 70-9
prompt
  confirmation, abandoning plan 25-5
  task confirmation, enabling or disabling 25-5
prone 30-31
prop 16-10, 47-5
  adding
    from feature layer 50-26
    to Props Palette 16-26, 57-8, 59-13
    to terrain 50-24
  attaching observer to 45-2, 45-5
  configuring 47-4
  creating 16-3, 16-4, 16-5, 16-8, 16-9, 16-11, 16-26, 59-13
  extracting 50-24, 56-20
    from terrain 47-4
  filtering attachment list 45-5
  heading, setting 16-17
  model definition, changing 50-30, 56-24
prop (continued)
  opacity 50-29
  opaque 50-29
  placing 16-11
  saving 16-26, 59-13
  transparent 50-29
  type 50-30
  viewing list of 50-28
property
  channel 73-8
  editing system 64-24
  object, setting 16-3
  tactical graphic, default 64-44
  window 73-7
Props Palette 16-3, 16-4, 16-5, 16-11
  adding objects to 16-26, 57-8, 59-13
  category, adding 59-13
propulsion noise 9-17, 9-18
protest, civilian 26-7
Protest Along Line dialog box 26-49
Protest Along Line task 26-49
Protest Around Location dialog box 26-49
Protest Around Location task 26-49
protesting, crowd 26-49
Provide Suppressive Fire on Location task 26-50, 26-52
Provide Suppressive Fire task 26-50, 26-51
providing, supplies 22-16
proximity fuse 30-14
proxy
  connecting through 51-5
  port 51-5
published object type 68-6
publishing
  entity velocity when paused A-5
  radio trasmitter 70-13
  tactical graphics 35-7
publish-transmitter parameter 70-13
publish-transmitters parameter 70-13
publishZeroVelocityWhenPaused parameter A-5
pulse repetition rate 60-2

Q

-q command-line option 5-10
Qt Designer 68-23
  UI file, details 68-24
Qt Linguist 2-11
quality, ocean 11-19

QualNet
example scenario 13-5
scenario 13-5
using with VR-Forces 13-4
query
feature 55-6
named 55-3
question
asking from script 29-22
scripted task 18-35
Quick Launch toolbar 8-10
quitting
plan 32-24
retask procedure 25-6

R

-r command-line option 5-10, 54-4
radar 30-33
jamming 22-16, 22-18
sensor 30-17
radar mode
search 30-33
track 30-33
Radar Mode dialog box 30-33
Radar Mode set data request 30-33
Radar-Jamming-Strength-Receiving 22-18
radio 30-9, 30-14
communication line 37-19
color 37-22
enabling and disabling 37-20
lifetime 37-21
configuring 70-13
message
configuring A-8
set data request 26-55
task 26-56
Radio Button Choice parameter 28-14
Radio Communications Settings page 37-20, 37-21, 37-22
radio transmitter, publishing 70-13
radius
emitter volume 60-3
sheaf 27-25
rain 11-5, 11-8
intensity 11-11
splash effect 39-2
Raise Periscope task 26-52
random, DI-Guy appearance 64-15
Random condition 31-13
Random Number Seed parameter 7-7
random-number-seed parameter 12-6
range
focal 44-22
ground clamping 19-23
intervisibility line 41-11
laser codes 9-15
range ring 18-25
configuring display of 18-28
displaying for entity 18-27
enabling and disabling 18-27
pinning 18-27
sensor 18-25
Range Rings Settings tab 18-28
range-determinant parameter 70-4, 70-6
range-list parameter 70-4
raster map
display order 50-10
displaying 50-7
Raster Maps page 50-7, 50-10, 50-21, 56-5
rate
climb, turn, and descent 26-30
turn 26-29
ray
crepuscular 38-11
intervisibility 41-4
sun 38-11
reactive task 20-22, 25-11, 26-64, 28-5, 29-8, 29-28
canceling 25-17
creating 28-24
disabling 25-14
enabling 25-13
Visit Interest Points 20-22
reader writer key value pair 66-26
Real DB 1-22
real time
checkpointing in 7-32
running faster or slower than 3-11
recalling
view 44-30
view state 44-29
receive buffer size 5-7, 5-12, A-8
Receive Text Message condition 31-13
receiving
radio messages 70-13
supplies 22-16
recently used scenario 7-18
Reconnaissance, posture 22-4
recording, to Logger 7-42

recovering, embedded entity 19-16
recovery 30-10
recvBufferSize parameter A-8
redo, ellipse rotation 35-16
Reduce Near Clip, policy 73-11
Reduce Z Fighting, policy 73-11
Reduce Z Fighting Ratio, attribute 73-11
reference ellipsoid A-4, A-6
  configuring A-6
Reference for Enumeration for Simulation Interoperability 68-4
Reference for Enumerations for Simulation Interoperability 3-8
reflection
  map 1-16, 50-20
  planar 38-7
refresh rate 6-19
regenerating
  ocean height map 50-19
  platform 69-12
region
  DDM 6-9
  update threshold 6-9
register, trigger 31-8
registrant_feature_list.csv 52-6
regulating, time 4-18
rejoining exercise 3-7
relative path 12-7
relative-heading parameter 19-13
relative-position parameter 19-13
Release Bomb on Laser Spot task 26-53
Release Bomb on Location task 26-53
Release Bomb on Target task 26-53
remapping
  back-ends 3-8
  batch mode 7-40
remote
  attachment, key-names 70-19
  control 1-16
    of the observer 44-32
  entity, attaching components to 7-5, 70-18
  object 3-9, 68-9, B-6
    network interface 68-12
  simulation object 3-9, 25-3
    markings 31-17
    plan 32-22
    use in plans 31-17
  stealth, displaying model 43-12
remoteAttachment example scenario 70-18
remoteConfigurations directory 70-19
remote-configurations entry 70-19
remote-objects parameter 68-9, B-6
removing
  assembly 66-13
  category 64-48, 64-49
  channel 73-6
  force 64-52
  formation 65-10
  navigation area 53-5
  object from Favorites list 16-26
  objects from selection group 17-11
  observer 43-9
  simulation objectfrom unit 23-8
  subordinate 65-5
  system 64-25
  view 44-31
  window 73-6
renaming
  category 64-48, 64-49
  formation 65-10
  observer mode 43-5
  overlay 34-5
  scenario 7-29
  selection group 17-11
Render Settings page 6-7, 6-18, 38-10, 38-11, 44-24, 48-12, 50-23, 73-13
Reorganize set data request 30-33
reorganizing
  concepts for units 21-4
  unit 30-33
    versus closing formation 21-5
repair 30-10
repeat image attribute 50-8
replace image attribute 50-8
replacing
  saved view list 44-32
  subordinate 65-6
replicating, settings and data 4-25
report
  NBC 27-28
  obstacle 27-28
repository B-4
require statement, Lua 29-6
reregistering, trigger 31-16
Reserve Space 19-9
resizeable window 73-2
resizing
  boxes and text overlay objects 35-15
  windows 73-12

resolution
ocean height map 50-18
tree 59-18
resource
adding to object 69-11
information 18-6
monitoring entity 18-30
pacing and tracking 30-34
specifying 30-34
status at creation 19-4
tracking unit 23-10
Resource condition 31-13
Resource Information, page 18-30
resource-item, adding 66-26
Resources dialog box 30-34
Resources Pacing/Tracking dialog box 30-34
Resources Pacing/Tracking set data request 30-34
Resources set data request 30-34
Restart Plan command 32-23
restarting, plan 32-23
Restore set data request 30-35
restoring
embedded entity parent 19-16
parameter values in OPD Editor 69-7
simulation object health and stores 30-35
unit 23-10, 30-35
view 44-30
restriction, unit movement 55-9
result
detonation, sound mapping 42-8
resuming, simulation 7-24
resupply 22-16, 30-10
modes 30-35
Resupply Mode dialog box 30-35
Resupply Mode set data request 30-35
retreat 27-26
retrograde movement 27-26
reusable software object 59-10
Reverse Direction 26-36
reversing, line direction 35-16
reverting
category changes 64-49
navigation area edits 53-5
parameter values 69-7
revision, RPR FOM 5-7, 5-11
revolving, observer 44-13
rewinding
scenario 7-25
scripted task 28-27
to a snapshot or checkpoint 7-25
RGB 47-4
RID file 2-10, 5-16
rid.mtl file 2-10, 5-16
right-click menu 74-6
configuring 75-5
rightclickmenu, element 75-5
ring, range 18-25
road
data, using in path planning 30-28
driving on 25-22
network 25-21
using route as 26-36
roll 44-3
roll up, rule 66-14
Roll updater 59-13
rolling back to a snapshot 7-25
rolling up assembly 66-15
Rotary Wing Land dialog box 26-54
Rotary Wing Land task 26-54
rotary-wing entity
altitude 26-28, 26-30
attacking with gun 26-6
creating, routes for 35-10
heading 26-29, 26-30
holding pattern 26-46, 26-47
landing 26-54
lateral movement, configuring 25-31
routes for 35-10
terrain avoidance 25-30
writing plans for 31-17
rotary-wing-entity-param parameter 68-12, B-8
rotary-wing-entity-state-repository 68-12
rotating, ellipse 35-16
rotation path, smoothing 6-16
RotDir parameter 59-17
RotOffsetDeg parameter 59-17
rotor wash 39-2
high quality 39-4
roughness type, mapped to soil type 19-18
round 10-2
Rout, posture 22-4
route 33-2, 33-5
altitude
displaying 33-9
setting 16-16
computed, creating 35-7
evaluating for aircraft 35-10
extruded 35-4
following 26-36
fixed-wing 25-26

route (continued)
for aircraft 35-10
patrolling 26-45
using as road 26-36
RPR FOM 5-17
revision 5-7, 5-11
specifying version A-7
version 5-7, 5-11
rprFomVersion parameter A-7
RSO 59-10
RSOName parameter 59-11
RTI 2-9
configuring for time management 4-19
definition of 2-9
RTI_CONFIG environment variable 2-10
RTI_timeMgmt parameter 4-19
RTI_useRtiExec parameter 4-19
rtiexec 4-19
rule, roll up 66-14
rules of engagement
pasting 16-23
setting 30-36
Rules of Engagement dialog box 30-36
Rules of Engagement set data request 30-36
run
scenario, using PDU 7-42
run mode A-6
starting back-end in 5-10
run-duration parameter 7-39
runInTimeManagementMode parameter 4-20, A-7
running 30-31
batch scenario 7-40
Boundary Generation Tool 52-15
simulation 7-24
runway 25-27
rwKeywordValuePairs.xml 66-26

S

-S command-line option 5-5, 5-6, 5-11, 5-12, 5-19, 45-6
-s command-line option 5-10, 5-14, 54-4
S-57
buoys and beacons 52-19
converting to shapefile 55-7
data 50-19
signal sequence 52-28
s57_attr_map.csv 52-21, 52-24
s57_defaults.csv 52-24
s57_layer_list.csv 52-19, 52-24
s57_modeldef_by_name_map.csv 52-22
Sail Heading dialog box 26-54
Sail Heading task 26-54
salvo 10-2
sample, scenario 7-23
sampling rate
freehand line 35-2
intervisibility, configuring 41-9
SanLuisObispo terrain database 48-19
Save Scenario dialog box 7-28
saved views file 6-1
scenario 12-2
saving
checkpoint 7-30
display engine configuration 73-6
element definition 57-20
model definition 57-14
object parameter database 69-12
plan 32-11
prop 16-26, 59-13
scenario 3-7, 7-27
existing 7-29
under new name 7-29
schemas 57-3
scripted tasks 28-27
settings 4-23, 4-25
tactical graphics 34-6
terrain 50-5
view 44-29, 44-30
to file 44-31
scale
factor, material ambient and diffuse value 38-4
SpeedTree, random 59-20
terrain 48-18
Scaled Speed check box 44-28
scaling
3D models 19-24
icon, 2D 18-22
model, linear and XR 19-24
scenario 6-1
batch 7-37
building blocks 7-16
closing 7-27
collaborative development 7-13
component attachment table 7-6
creating new 7-3
date, setting 7-7
description, editing 7-23
end time 7-7, 7-35
new scenario 7-35

scenario (continued)
file 3-7, 6-1, 12-2
editing 12-3
parameters 12-4
pathnames 12-7
specifying which to use A-5
frame mode 7-7
frame time 7-7
importing 7-13, 7-16
information, displaying 7-21
loading 5-5, 5-9, 7-17
at startup 5-9
recently used 7-18
merging 7-13
MSDL, importing 7-14
name 7-5
order of battle 12-10
parameter
setting 7-5
version 12-6
QualNet 13-5
random number seed 7-7
remoteAttachment 70-18
resuming 7-24
rewinding 7-25
scripted task 28-27
rolling back to snapshot or checkpoint 7-25
running 7-24
sample 7-23
saving 3-7, 7-27
existing 7-29
under new name 7-29
simulation model set 7-6
SMS, multiple 7-8
starting 7-24
starting date and time 7-12
default 7-12
startup observer view 44-31
temporary directory 12-11
terrain 12-8
display 7-6
simulation 7-5
time, setting 7-7
time multiplier 7-6
zoom to extents on load 44-21
scenario event 8-2
adding audio, video, and graphics 8-5
creating 8-3
deleting 8-13
ending 8-12
scenario event (continued)
intelligence object 8-8
sending 8-10
sending from plan 8-11
Scenario Event condition 31-13
Scenario Event dialog box 8-11
Scenario Event Manager 8-3, 8-8, 8-10, 8-12, 8-13
Scenario Event Settings page 8-14
Scenario Event Toolbar 8-2
scenario extras file 12-2, 20-9
Scenario Information dialog box 7-23
Scenario Name parameter 7-5
scenario scripted task 28-3
Scenario Snapshots dialog box 7-26, 7-31, 7-34, 7-35
Scenario Startup dialog box 4-6, 4-8
Scenario Toolbar 7-34
scenario-data parameter 12-6
ScenarioDescription parameter 12-6
ScenarioExtentInformation parameter 12-6
scenarioFileName parameter A-5
scenario-filename parameter 7-39
scenario-name parameter 12-6
Scenario-Scripts parameter 12-5
scenario-specific
GUI settings file 12-2
spawn pattern 20-9
scheduling
periodic checkpoint 7-32
spawn event 20-6
schema 57-3
copying 57-5
creating 57-4
deleting 57-5
model definition 57-3
parameter
adding 57-6
deleting 57-7
editing 57-7
saving 57-3
tactical graphic model 64-46
visual definition 57-3
Schema Editor page 57-4, 57-5, 57-6, 57-7, 57-9
SciTE 28-4
Screen Number attribute 73-7
script
extended name 28-12
ID 28-20
location, object 29-18

scripted
behavior 26-5
movement, configuring 64-26
scripted task 28-3, 28-4
actioncategory 70-15
adding to folder 28-29
adding to task toolbar 28-22
adding to toolbar 75-6
availability, by simulation object type or Behavior Set 28-18
background process 29-29
copying 28-32
creating 28-6
deleting 28-32
editing 28-28
entry point 29-7
execution flow 29-9
exporting and importing 28-30
filtering list 28-28
folder
adding 28-29
deleting 28-29
renaming 28-29
importing 28-31
in system definition 28-20
interactive 29-22
language 28-23
making available by Behavior Set 28-20
metadata 28-5
organizing 28-29
package 28-30
parameter types 28-12
previewing dialog box 28-10
question 18-35
reactive task 25-11, 29-8, 29-28
removing from folder 28-29
rewinding 28-27
saving 28-27
script, editing 28-37
script ID 28-8, 28-11, 29-15
script storage location 28-30
simulation object type supported 28-19
system
creating 28-33
making available on Task menu 28-34
storage location 28-33
system and scenario 28-3
Task menu location 28-16
text editor, specifying 28-36
Update_EW_Degradation 22-18
Scripted Task Options page 28-36
script-ids parameter 28-20
scripts file 12-2
sea
choppiness 11-5
spray 39-2
enabling and disabling 39-4
state 11-5, 11-17
Douglas 11-15
swell 11-5, 11-17
search, radar mode 30-33
seasonal buoys and beacons 52-26
secondary
simulation object 45-9, 45-12
window 4-14
sector, unit 18-25, 22-4
sector of responsibility, setting 30-37
Sector of Responsibility dialog box 30-37
Sector of Responsibility set data request 30-37
segment
angle, setting 60-4
arc size 60-4
edge 60-5
emitter volume 60-4
outline, displaying 60-5
Selected Simulation Engine Toolbar 16-6, 16-8
selecting
database for front-end 7-3
entity, from Object Console Summary Panel 17-6
object 17-4
in Objects List Panel 17-6
to create 16-4
objects 64-3
observer 43-6
selection group 17-10
simulation objects 17-5
unit 23-6, 24-2
selection box 17-5
selection group 20-17
adding object to new 17-9
adding to 17-10
creating 17-9
creating from keyboard 17-10
creating from menu 17-9
deleting 17-11
file, scenario 12-2
introduction 17-8
removing object from 17-11
renaming 17-11

selection group (continued)
selecting 17-10
Selection Group View 17-2
SelectionGroups parameter 12-5
self, as condition parameter 32-7
self reflection, DIS 5-12
send buffer size 5-7, 5-12, A-8
Send NBC Report dialog box 27-28
Send Obstacle Report dialog box 27-28
Send Radio Set task 26-55
Send Radio Task task 26-56
Send Text Message task 26-57
sendBackendLogToNetwork parameter A-5
sendBufferSize parameter A-8
sendFedTime parameter A-7
sending
console message 32-16
data, configuring A-4
messages 15-8
NBC report 27-28
obstacle report 27-28
radio messages 70-13
scenario events 8-10
from plan 8-11
set data request via radio message 26-55
task via radio message 26-56
text message 26-57
view control, in plan 32-19
send-spot-reports-on-own-force parameter 9-7
SenSim 40-2
sensing, electronic emission 22-16
sensor 9-13, 15-7, 15-8, 71-2
See also. sensor signature
adding new 71-9
affect of posture 22-5
aiming 30-38
attaching to remote entity 70-18
channel attribute 73-8
configuring 71-2
connecting 71-3
detection types 71-9
emission 22-19
footprint 22-8
information 18-6
mode, full color 40-5
parameter, editing 66-21
radar 30-17
range ring 18-25
sensor contact line 37-13
SIGINT 22-19
sensor (continued)
signature-object-sensor 71-2
system 64-22
system definition file 68-16
editing 71-4
unit 22-8
visual, illumination 71-12
zooming 30-38
Sensor Aim set data request 30-38
sensor component, adding 71-11
sensor contact line
configuring 37-15
enabling 37-14
enabling and disabling 37-13
specifying objects that can display 37-14
sensor contact line pinning 37-14
sensor effects 40-2, 40-3
blur, noise, and gain 40-4
configuring 40-4
Sensor Settings page 40-4, 40-5
sensor signature 9-17, 64-23, 71-5
adding 71-12
affect of posture 22-5
modifying 35-9
Sensor Zoomset data request 30-38
separation-distance parameter 25-10
separation-tolerance parameter 25-10
separator, menu 75-2
server
terrain 47-7, 47-9
caching 50-32
cutting in site 52-13
session 3-5
ID 4-3, 5-6, 5-7
assigning 4-9
configuring A-5
setting at startup 5-9
joining at startup 4-17
message, configuring 4-17
starting and stopping 7-42
Session Options page 4-8, 7-10, 7-11, 7-12, 7-22, 7-42, 18-19, 18-20, 44-21, 74-8
sessionId parameter A-5
Set Ammunition 30-8
Set Appearance 38-6
Set command, default 30-4
set data request 24-2
Activation 30-5
Active Sonar Mode 30-6
Aiming 30-7

set data request (continued)
Altitude 25-24, 30-7
Ammunition Pacing/Tracking 30-8
Appearance 30-9
Armed 30-9
Capabilities 30-10
Collision Avoidance Types 30-11
Concealed 30-12
Contamination 30-12
Counter Measures Auto Launch 9-19, 30-12
Current Speed 30-13
Destroyed 30-13
Detonation Fuse Type 30-14
DI-Guy Appearance 30-15
DI-Guy Hand Item 30-15
Disembarked 19-11, 30-16
Embarked 19-9, 30-16
Emitter 27-4, 27-19, 30-17
Engagement Results 30-18
Equipment Pacing/Tracking 30-19
Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking 30-20
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant 30-19
Force 30-21
Formation 30-22
in plan 31-16
Heading 30-23
IFF 30-24
Jam Targets 30-25
Lase Autonomous 9-16, 30-25
Laser Code 9-15, 9-16, 30-26
Location 30-26
Navigation Preferences 30-28
Notify Level 30-28
Ordered Speed 30-29
Percent Complete 30-29
Posture 30-30, 30-31
Radar mode 30-33
Reorganize 30-33
Resources 30-34
Resources Pacing/Tracking 30-34
Restore 30-35
Resupply Mode 30-35
Rules of Engagement 30-36
Sector of Responsibility 30-37
sending by radio 26-55
Sensor Aim 30-38
Sensor Zoom 30-38
Sonar Depth 30-39
set data request (continued)
Spot Reports 30-40
Superior 30-41
Supplying 30-41
Surrendered 30-41
Synchronize Laser Code 9-16, 30-42
Target 30-42
Tasked by Superior 25-11, 30-43
Tracer Use 30-43
Unit State 30-43
Weapons Pacing/Tracking 30-44
Set MOPP Level 30-27
Set Morale 30-27
Set Scenario End Time dialog box 7-36
Set Sensor Signatures dialog box 8-9
set statement 24-2
See also. set data request
Set toolbar, adding to 75-6
setMainThreadToHighPriority parameter A-5
setting
altitude 16-14, 30-7
dialog box 16-15
manually 16-15
route 16-16
altitude for fixed-wing entity 25-24
appearance 30-9
autonomous lasing 30-25
capabilities 30-10
collision avoidance types 30-11
color, unit bounding volume 24-11
combat engineering object percent complete 30-29
communication model 13-2
concealment 30-12
date 11-3
emitter 30-17
emitter volume radius 60-3
factory and default 4-23
formation 30-22
frame rate 5-4
heading 16-3, 30-23
manually 16-17
icons 4-23
IFF 30-24
laser code 30-26
loading 4-23
location 16-3, 30-26
LOD scale 38-8
MOPP level 30-27
morale 30-27

setting (continued)
name 16-3
notification level 5-15, 30-28
entity 18-33
ordered speed 30-29
overlay 16-3
posture 30-31
precipitation
intensity 11-11
type 11-11
prop opacity 50-29
properties, object 16-3
resources 30-19
rules of engagement 30-36
saving 4-23
sector of responsibility 30-37
segment arc size 60-4
simulation object 30-4
parameters 30-4
to be destroyed 30-13
simulation speed with time management 7-24
sound volume 42-3
superior 30-41
support point 64-11
target 30-42
time of day 11-4
time zone 11-4
user 4-25
visibility 11-10
weather 11-8
settings
GUI layout 5-4
property, default 64-44
scenario GUI 12-2
Simulation Object Editor, GUI layout 63-4
synchronizing 4-25
Settings menu 4-23
shader 1-16, 50-20
debugging 50-22
shadow
configuring 38-16
enabling and disabling 38-14
text, 2D label 18-17
Shadow Settings page 38-15, 38-16
shake, camera 44-25
shaking camera 44-25
shape terrain data 47-3
shapefile
buoys and beacons 52-19
converting S-57 to 55-7
shapefile (continued)
layer name 52-8
mapping to feature 50-11
sharing, resources 58-7
sheaf radius 27-25
ship
wake, configuring 59-14
shooter, laser 9-15
short name 64-6
Show Database Correlation Warning Dialogs 4-18
Show Function Profiler Overlay 6-7
Show Ground Truth For 9-6
Show Only Active Environmentals 35-18
Show Performance Statistics Overlay check box 6-7
Show Session Terrain Change Prompts 4-18
Show Spot Reports from Viewpoint of Force 9-5, 9-6
Show Spot Reports Sent By 9-6
showHuds 73-8
showing
objects 33-5, 33-7
symbol decoration 18-16
vertex labels 33-8
showLines parameter 60-5
SHP, file 47-3
SIGGRP, attribute 52-28
SIGINT sensor 22-19
signal
interaction 37-19
sequence 52-28
signal group attribute 52-28
signal period attribute 52-28
signal_slot 68-24
signature, sensor 9-17, 71-5
signature propagator, configuring 71-9
signature-object-sensor 71-2
SIGPER, attribute 52-28
SIGSEQ attribute 52-28
SilverLining 1-20
license 1-20
simMgmtPduProcessLevel parameter A-5
SimObject class 29-4
simple driver 52-18
simple-radio-comm-model 15-6
Simthetiq 1-22
simulated world, navigating 44-3
simulation
closing 7-27
date 7-12

simulation (continued)
engine 3-2
objects 6-4
pausing 7-25
resuming 7-24
rewinding 7-25
running 7-24
automatically 7-37
speed, with time management 7-24
start mode A-6
starting 7-24
time 3-10, 4-18, 7-12
checkpointing in 7-32
evaluating in plan 31-13
Simulation Connections Configuration dialog box 4-33
Simulation Control Toolbar 4-12, 7-24, 7-25, 7-26
Simulation Date/Time condition 31-14
simulation engine
closing 4-25
exiting 4-25
specifying 16-6, 16-8
simulation management message, configuring A-5
simulation model set 63-5
AggregateLevel.sms 22-2
configuration 7-9
default 7-11
creating 63-16
default 5-3
deleting entity 64-42
entity-level and aggregate-level 15-10
file 63-5
hostility file 9-10
multiple 7-8
new, creating 63-17
opening 63-15
specifying on command-line 5-10
simulation model set configuration, default 7-9
Simulation Model Sets parameter 7-6
simulation object
See also entity and unit
adding to, selection group 17-10
assigning plan to multiple 32-22
attaching observer to 45-6
attaching tactical graphic to 35-11
attribute, information 18-6
automatically attaching tactical graphic to 35-11
behavior 15-9
simulation object (continued)
centering observer on 44-25
changing
name 64-6
undoing changes 64-8
changing category 64-3
collision and obstacle avoidance 19-17
communication model 15-6
components 68-13
concepts 15-2
console, setting notification level 30-28
console message 18-32
saving 18-35
sending 32-16
viewing 18-34
copying 16-21
count 18-37
creating 16-3, 16-4, 16-8, 16-11, 19-3
from existing 64-32
in plan 32-17
multiple 20-17
creating multiple 20-16
creation, enabling and disabling 64-6
deleting 19-5
in plan 32-17
deleting in plan 32-17
destroyed 30-13
dragging 16-19
to new location 30-26
editing 19-5, 63-2
element definition 57-14
enumeration, published and matching 68-6
filtering attachment list 45-5
filtering list of 25-9
filtering using interest management 6-8
firing at target 9-13
heading 30-23
setting 16-17
hiding 9-2
hostility 9-10
icon 18-22
CID level and color 9-13
image 59-9
identifiers 15-2
information 18-3
multiple 18-10
inset view 46-2
label 15-4, 18-15
extended 18-18
list, spot report 9-9

simulation object (continued)
local and remote 3-9
model 63-5
editing 64-3
move to location 26-39
name 18-10, 64-6
length restrictions 15-3
pasting 16-23
patrol between 26-46
patrol route 26-45
pinning intervisibility object to 41-10
placing 16-11
plan
issuing to 32-18
viewing 31-14
platform, updating 63-17
properties, specifying at creation 16-13
remote
in task 25-3
plans 32-22
removing, from unit 23-8
resource, monitoring 18-30
resource at creation 19-4
restoring health and stores 30-35
selecting 17-4, 17-5
from Object Console Summary Panel 17-6
in Objects List Panel 17-6
setting
appearance 30-9
location 30-26
state and parameters 30-4
specifying resource 30-34
spot report, enabling or disabling 30-40
state 15-8, 30-4
systems
adding 64-23
editing 64-22
taking damage from munition 9-22
taking damage from munitions 70-7
target detection 9-13
target in global command 32-15
task 15-9
tasking by superior 30-43
tick rate, tuning 6-12
track history 37-5
trailing effect 37-3
unselecting 17-7
UUID 15-2, 15-3
viewing in Navigation Lab 53-14
views of 17-2
simulation object (continued)
waiting 26-62
ways to add to scenario 7-16
writing plan 32-2
zooming to extents 44-21
Simulation Object Editor 1-19, 63-2
command-line arguments 63-4
configuring DI-Guy character and appearance 64-14
embedded entity 19-12
force
adding 64-50
editing 64-51
removing 64-52
formation, bounding box 65-11
generating navigation data for entity 53-8
managing forces 64-50
navigating edited simulation objects 64-42
opening OPD Editor from 64-25
simulation object group 64-41
starting 19-20, 63-3
variable binding
adding 68-23
system definition file 68-25
simulation object group 7-16, 16-25
connection 64-42
creating 64-41
terrain 5-5
Simulation Object parameter 28-15
simulation object type
scripted task availability 28-18
scripted task validity 28-19
Simulation Object Type Mappings dialog box 58-2
filtering element definition list 58-5
Simulation Objects Palette 16-3, 16-4, 16-5
assigning or removing simulation object 64-6
Simulation Ocean Settings page 50-20
Simulation Selection Toolbar 4-12
Simulation Terrain parameter 7-5
Simulation Time condition 31-13
Simulation Time Scale Toolbar 4-12, 7-24
Simulation Time Toolbar 4-12
simulationModelSet parameter A-5
Simulation-Model-Set-Files parameter 12-6
simulation-run-duration parameter 7-39
sink point 20-2
creating 20-4
SISO, Enumeration Document 3-8
SISO Enumeration Document 68-4

site, cutting in 52-13
site ID 3-4, 5-7, 5-10, 12-9
  configuring A-5
  specifying 5-10
    at startup 5-14
siteId parameter A-5
sitting 30-31
situational awareness 43-10
size
  region 6-9
  SpeedTree, randomizing 59-20
  window 73-7
Size group box 64-9, 64-11
Skip Task, task 25-11
skipping task 25-9
skybox cube map 38-12
  omitting clouds 38-12
Skybox Cube Map Regeneration Threshold 38-12
slider
  visual quality, configuring 6-15
slot
  adding 64-33, 64-35, 64-38
  configuring 64-35, 64-38
  embarkation 64-33, 70-16
  exclusion 64-40
smallCompass, keyword 44-5
smoke
  creating 26-33
  plume 37-8
  trail 37-3
smooth period 6-16
smoothing 6-16
SMS
  migrating 63-12, 63-13
  upgrading 63-13
  vrfSim.opd 63-9
sms-filename parameter 7-39
snap to grid, formation 65-14
snapshot 7-29
  automatic deletion 7-25
  clearing list 7-35
  creating 7-34
  deleting 7-35
  enabling periodic 7-34
  rolling back to 7-25
  saving as checkpoint 7-31
snow 11-5, 11-8
  intensity 11-11
SO 4-33
SOEUILayoutSettings 63-4
SOG. See simulation object group
soil type 50-36
  configuring for GDB 48-2
  mapped to roughness type 19-18
  point and feature 48-17
soil-list parameter 19-18
solid line 35-17
sonar
  active, mode 30-6
  active and passive 9-17
  depth 30-39
  dipping 26-57
  information 18-6
  sonobuoy 26-11
  thermocline 9-17, 11-20
Sonar Depth dialog box 30-39
Sonar Depth set data request 30-39
Sonar Dip task 26-57
sonobuoy, deploying 26-11, 26-12
sound effects 42-2
  adjusting volume 42-3
  enabling 42-2
  mapping, weapon fire and detonation 42-8
  mapping to entities 42-4
  range 42-3
sound file, mapping to entity 42-5
sound mapping
  channel 42-8
  detonation result 42-8
Space Follow mode 45-10
spacing, contour line 48-10
spawn burst 20-8
spawn event, scheduling 20-6
spawn interval 20-6, 20-8, 20-12
  variance 20-6, 20-8, 20-12
spawn pattern 20-5
  bursts 20-12
  copying 20-14
  creating 20-10
  default 20-9
  deleting 20-16
  description 20-10
  editing 20-15
  name 20-10
  new 20-10
  scenario-specific 20-9
  timing 20-11
Spawn Pattern Manager 20-2, 20-4, 20-10, 20-14, 20-15, 20-16

spawn point 20-2
  creating 20-2
spawned, entity 20-2
specifying
  application number 5-8
  bounding volume 64-9
  damage model 64-24
  default *default.sms* 63-17
  default SMS 63-17
  entity to create 20-20
  FED file 5-18
  frame length 12-6
  frame rate mode 12-6
  HLA federation execution 5-16
  language 5-4
  log file 5-4
  movement model 64-24
  notification level 5-5
  point-to-point and multicast modes 5-19
  port, at startup 5-19
  resource 30-34
  segment angle 60-4
  simulation model set 5-10
  site ID 5-10
  target simulation object in global command 32-15
  terrain profile sampling frequency 49-3
  unit target 30-32
specular
  color 38-4
  map 50-20
specular map 1-16
speed 18-11
  gain 44-28
  scaling 44-27
    enabling and disabling 44-27
  setting 30-29
  specifying measurement unit 74-9
  tidal current 11-15, 11-17
  wind 11-8, 11-9
speed scaling, mode 44-28
speed-to-rpm-factor parameter 9-18
SpeedTree 1-20, 52-6
  configuring 59-18
  license 1-22
  LOD, zoom 5-6
  model definition 57-9
  scale, random 59-20
SpeedTree Randomization page 59-21
SpeedTree Settings page 59-19
speedTreeManager 52-6
spheroidSemiMajor parameter A-6
spheroidSemiMinor parameter A-6
splash effect 39-2
spot report 5-4, 9-2, 27-28
  affect on performance 6-11
  custom viewpoint 9-6
  decay rate 9-8
  enabling and disabling 9-3
    for simulation object 30-40
  label 9-9
  role in target detection 9-15
  tactical graphics, for 9-9
  using in task 9-9
  viewpoint 9-4, 9-5
Spot Report Options page 9-3, 9-5, 9-6, 9-8, 9-9, 30-40
Spot Reports dialog box 30-40
Spot Reports set data request 30-40
spotlight 38-2, 38-6
spray, enabling and disabling 39-5
squatting 30-31
squawk indicator 37-19
  color 37-22
  enabling and disabling 37-20
  lifetime 37-21
standard beam 30-17
standing 30-31
starting
  back-end 4-3
  front-end 4-3
  OPD Editor 69-2
  scenario event 8-10
  simulation 7-24
  Simulation Object Editor 19-20, 63-3
  VR-Forces 4-3
startInRunMode parameter A-6
Start/Resume message 7-42
startup
  option 5-1
  tutorial 4-10
state
  sea 11-17
  simulation object, information 18-6
  update, sending 43-11
state repository B-4
  hierarchy 68-10, 68-12
statement
  adding to plan 32-3
  conditional 32-4

statement (continued)
deleting 32-10
editing in plan 32-10
state-repository parameter B-4, B-6
state-repository-min-tick-period parameter 6-13
state-repository-min-tick-period-variance parameter 6-13
statistics
displaying 6-6
frame rate, log file A-4
status
combat engineering object 22-12
entity 18-30
navigation area 53-7
personnel, ammunition, weapons, and POL 18-4
subtask 29-16
task 29-16
statusUpdateHeartbeat parameter A-6
Stealth
control 32-19
observer mode 4-25
window 4-14
stencil bits 5-5
stencil buffer 6-13
stereo display, types of 73-21
Stop/Freeze message 7-42
stopping
movement 27-22
scenario event 8-12
subtask 29-16
task 25-9, 29-16
stopping-distance-factor parameter 70-15
stores, restoring simulation object 30-35
Strafe Ground Target dialog box 26-59
Strafe Ground Target task 26-58
strafing, target 26-21
streamed data
performance 6-11
processing 52-6
streaming
beacon 52-24
buoys and beacons 52-19
navigation lights 52-25
terrain 47-9
street light 38-2, 38-6
strength 22-3
strong point
creating 27-19
improving 27-24
style
fill 35-17
line 35-17
pen 35-17
subcomponent B-4
classes 68-9
subgoal 3-15
submarine
lowering periscope 26-35
moving to depth 26-38
raising periscope 26-52
subnet mask 5-7, A-8
subnetMask parameter A-8
subordinate
adding to unit 65-5
disaggregated state 23-9
editing 65-7
in condition 32-7
information 18-7
order, changing 65-6
removing from unit 65-5
replacing 65-6
unit, aggregated state 23-9
Subordinates tab, unit 23-9
substitute-message-size parameter 13-3
subsurface entity
moving to depth 26-38
RPM 9-18
subsurface-entity-param parameter 68-12
subsurface-entity-param parameter B-8
subsurface-entity-state-repository 68-12
subtask
Lua 29-15
state
canceled 29-16
complete 29-16
running 29-16
status 29-16
stopping 29-16
sun, ray 38-11
sunbeam 38-11
sunrise-time parameter 71-13
sunset-time parameter 71-13
superior
setting 30-41
tasking simulation object 30-43
Superior dialog box 30-41
Superior set data request 30-41
super-type 68-5

supplies 22-14
information 18-7
providing 22-16
receiving 22-16
starting or stopping delivery 30-41
Supplies page 22-12
supplies parameter, editing 66-24
supply line 37-17
Supplying dialog box 30-41
Supplying set data request 30-41
support, technical lii
support point, setting 64-11
suppression-degrades-sensor-performance-by parameter 70-18
suppression-detonation-insult parameter 70-17
suppression-detonation-range parameter 70-17
suppression-insult-level parameter 70-17
suppression-recovery-time parameter 70-17
suppressive fire
at location 26-50
at point, line, or area 26-50, 26-51
suppressSelfReflect parameter A-8
surface
characteristics 50-36
clamping to 19-21
damage specification 70-6
transparency 11-15, 11-17, 73-17
type, mapped to roughness type 19-18
surface entity
buoyancy 39-6
RPM 9-18
wake 37-3
configuring 59-14
surface-entity-param parameter 68-12, B-8
surface-entity-state-repository 68-12
surfChar.map 50-36
surge depth 11-5, 11-15, 11-17
surrender
lifeform, testing 31-12
surrendered 31-12
Surrendered dialog box 30-41
Surrendered set data request 30-41
swell 1-15, 11-5, 11-15
sea 11-17
switch bead 61-8
switch node 59-17
visual model 64-21
symbol, location of 2-4
symbol decoration, showing and hiding 18-16
Symbol Decoration Settings page 18-15, 18-17
synchronization, vertical 6-19
Synchronize Laser Code dialog box 30-42
Synchronize Laser Code set data request 9-16, 30-42
synchronized, ocean 5-5
synchronizing
back-ends 4-18
data and settings 4-25
laser codes 30-42
sysdef file. See system definition file
system 68-16
active-radar-sensor 22-18
adding to simulation object 64-23
aggregate-comms-jammer 22-17
aggregate-radar-jammer 22-18
connection 68-18
damage 9-22
editing 64-25
properties 64-24
embedded entity 19-12
emissions-sensor 22-19
entity 64-22
marker in 3D model view 64-24
movement, properties 66-20
Naval Depth Charge Deployment 26-14, 26-15
Naval Mine Deployment 26-16, 26-17
removing 64-25
scripted task 28-3
creating 28-33
sensor 71-2
simulation object, editing 64-22
system definition, specifying scripted task in 28-20
system definition file 68-16
applying new variable bindings 63-17
configuring connections 68-18
creating 68-17
sensor 71-4
variable binding 68-21
adding 68-22
addingto Simulation Object Editor 68-25
system scripted task, making available on Task menu 28-34

T

-T command-line option 5-10
-t command-line option 54-4
tab
Create Object 16-11
overlay 34-4

table
detection 9-13
Lua 29-15
taskParameters 29-17
tactical graphic 33-2, 58-2
active and inactive 30-5, 35-18
attaching to simulation object 35-11
automatically attaching to simulation object 35-11
box, drawing 35-3
changing name 19-5
color, changing 35-16
compared to combat engineering object 22-12
console, notification level 18-33
creating 16-3, 16-4, 16-5, 16-8, 16-9, 16-11, 35-2
editing 35-12
element definition 57-14
extruded 35-4
HAT lines 33-9
hiding from Overlays tab 33-7
information 18-3
label 33-8
labels 33-3
loading 34-6
menu icon 64-43
model 64-46
moving, to different overlay 35-5
placing 16-11
property
default value 64-44
editing 64-45
keyword 64-45
setting at creation 16-13
publishing 35-7
saving 34-6
selecting 17-5
showing and hiding 30-5, 33-5, 33-7, 35-18
texture 64-46, 64-47
vertex
adding 35-13
deleting 35-15
volumetric 35-4
Tactical Graphic Active condition 30-5, 31-14, 35-18
Tactical Graphics Display Settings page 33-7, 33-8, 33-9, 36-3
Tactical Graphics Palette 16-4, 16-5
Tactical Graphics Settings page 35-18
tactical smoke, enabling or disabling 37-17
taillight 38-2, 38-6
takeoff
carrier from 26-60
fixed-wing entity 25-25, 25-27
taking off 26-27
target
bomb 26-21
detection 9-13
spot reports 9-15
firing at 9-13, 26-23
identifying with laser beam 9-15
setting 30-42
specifying using laser 26-32
strafe 26-21
strafing 26-58
unit, preferred 30-32
Target dialog box 30-42
Target set data request 30-42
targetFrameRate parameter A-6
targeting-capable parameter 30-17
target-type parameter 70-3
task 25-3
Activate Jammer 27-4
adding to plan 32-3
Air-to-Ground Attack 27-4
Animated Movement 26-5
Anti-ship (Fixed Depth) 26-34
Arm Mine at Depth 26-6
assigning to unit 25-10
Attack Aircraft with Guns 26-6
Attack By Fire 27-5
Attack with Anti-Air Missile 27-5
Attack with Anti-Ship Missile 27-5
Attack with Land-Attack Missile 27-6
Attack with Torpedo 27-6
Automatic Air Defense 27-7
background 27-29
background process 29-29
Biological Attack 27-7
Bomb Location 27-8
Breach Obstacles 27-9
by superior 30-43
C++ 28-3, 29-15
Change MOPP Level 27-9, 30-27
Change Posture 27-10, 30-30
Chemical Attack 27-10
Civilian Idle 26-7
Civilian Protest 26-7
Come to Stop 26-7
concepts 24-1

task (continued)
concurrent 24-1, 25-4, 28-8, 28-17
configuring mutually exclusive 70-14
confirmation prompt, enabling or disabling 25-5
Construct Abatis 27-11
Construct Anti-Tank Ditch 27-12
Construct Barbed Wire 27-12
Construct Barricade 27-13
Construct Berm 27-13
Construct Booby Trap 27-14
Construct Bridge 27-15
Construct Dry Gap 27-16
Construct Fortified Area 27-17
Construct Fortified Line 27-18
Construct Minefield 27-18
Construct Road 27-27
Construct Strong Point 27-19
Convoy Along 25-10, 26-8
Convoy To 25-10, 26-9
Crowd Along Line 26-9
Crowd Around Location 26-10
Crowd Around Object 26-10
Crowd in Front of Entity 26-11
Deactivate Jammer 27-19
Deploy Sonobuoy 26-11
Deploy Sonobuoys Along Route 26-12
Destroy Bridge 27-20
Destroy Ditch 27-16, 27-20
Destroy Explosive 27-20
Destroy Fortification 27-21
Destroy Obstacle 27-21
Destroy Road 27-27
DI-Guy Animation 26-14
Disembark 26-13
Disembark All 26-13
Disperse Crowd 26-13
Drop Naval Depth Charge 26-14
Drop Naval Depth Charge at Location 26-15
Drop Naval Mine 26-16
Drop Naval Mines Along Route 26-17
Embark 19-8, 26-18
escaping retask procedure 25-6
Execute Close Air Support 26-21
Explode Charge at Depth 26-23
FASCAM 27-22
filtering simulation object list 25-9
Fire at Target 26-23
Fire Cruise Missile 26-24
Fire-for-Effect 26-25

task (continued)
Fixed Wing Land 25-28, 26-26
Fixed Wing Takeoff 25-27, 26-27
Flee from Entities 26-27
Fly Altitude 26-28
Fly Heading 26-29
Fly Heading and Altitude 26-30
Follow Entity 26-31
Halt Movement 27-22
ID 29-15, 29-16
Improve Booby Trap 27-22
Improve Breach 27-23
Improve Bridge 27-23
Improve Ditch 27-16, 27-23
Improve Fortification 27-24
Improve Obstacle 27-24
Improve Road 27-27
independent 24-1
Indirect Fire 27-25
Lase Target 9-16, 26-32
Launch Anti-Submarine Missile (Vertical) 26-32
Launch Counter Measures 9-19
Launch Smoke 26-33
Lower Periscope 26-35
Lua 29-15
parameter 29-17
Move Along Route 26-36
Move Along Route Retrograde 27-26
Move Into Formation 26-37
Move to Altitude 26-38
Move to Depth 26-38
Move to Location 26-39
for plan 26-40
Move to Location (Plan Along Roads) 25-21, 26-40
Move to Location Retrograde 27-26
Move to Waypoint 26-42
Move to Waypoint (Plan Along Roads) 25-21, 26-43
Move to Waypoint Retrograde 27-26
movement, affect of aggregation state change 23-10
mutually exclusive 24-1, 25-4, 31-7
Nuclear Attack 27-27
Orbit 26-45
parameters 25-6
Patrol Between 26-46
Patrol Route 26-45
Pattern Hold (Location) 26-46

task (continued)
Pattern Hold (Waypoint) 26-47
performing 15-8
Place IED 26-48
procedures, introduction 26-4
Protest Along Line 26-49
Protest Around Location 26-49
Provide Suppressive Fire 26-50, 26-51
Provide Suppressive Fire on Location 26-50, 26-52
Raise Periscope 26-52
reactive 20-22, 25-11, 26-64, 28-5, 29-8, 29-28
Release Bomb on Laser Spot 26-53
Release Bomb on Location 26-53
Release Bomb on Target 26-53
retask 25-3
Rotary Wing Land 26-54
Sail Heading 26-54
scripted 28-3, 28-4
action category 70-15
adding to task toolbar 28-22
adding to toolbar 75-6
available by Behavior Set 28-20
execution flow 29-9
filtering list 28-28
Send Radio Set 26-55
Send Radio Task 26-56
Send Text Message 26-57
simulation object 15-9
Skip Task 25-11
skipping 25-9
Sonar Dip 26-57
spot report, using in 9-9
state
canceled 29-16
complete 29-16
running 29-16
status 18-6, 29-16
stopping 25-9, 29-16
Strafe Ground Target 26-58
Taxi and Launch from Carrier 26-60
Taxi On Carrier 26-61
toolbar, adding to 75-6
Turn to Heading 26-61
unit concepts 21-5
user 26-62
viewing current 31-14
visualizing 37-12
Wait 26-62
Wait Duration 26-63
task (continued)
Wait Elapsed 26-63
Wander 26-63
Wander In Area 26-64
who can execute 25-6
Task menu 28-4
extended name 28-12
icon 28-11
location for scripted task 28-16
system scripted task, hiding 28-34
task visualization, pinning 37-13
Tasked by Superior set data request 25-11, 30-43
using in plans 31-16
tasking, by superior 30-43
task-manager parameter B-5, B-7
taskParameters, table 29-17
taxi, aircraft 26-60
Taxi and Launch from Carrier dialog box 26-60
Taxi and Launch from Carrier task 26-60
Taxi on Carrier dialog box 26-61
Taxi On Carrier task 26-61
TDB Tool 46-1
tearing off menu 74-5
technical support lii
teleporting observer 44-16
template, unit 66-3
temporary directory, scenario 12-11
terminating, application 4-25
terrain
See also terrain database and terrain server
agility 1-13, 47-2
altitude point
changing color 48-6
deleting 48-5
inserting 48-5
analyzing 49-4
avoidance 25-30
building 50-5
caching earth files 1-19
coloring by elevation 48-4
composability 47-2
coordinates 48-17
creating 50-3
deleting cached 50-33
displaying raster image on 50-7
dragging with mouse 44-14
DTED support 50-4
extracting props from 47-4
feature data, configuring 55-2
file caching 50-31

terrain (continued)
following 35-10
formats, supported 47-3
GDB, soil type 48-2
geocentric 47-6
large 47-7
loading, from command line 5-5
MetaFlight
designing 51-12
loading 51-5
processing 54-2
opening 47-7
orbiting, with mouse 44-15, 44-18
paged
flipping DDS textures 50-35
loading 5-9
preprocessing 51-5
page-in area 33-2, 51-6
paging 47-8
manually 51-6
MetaFlight 51-5
patch 47-3
adding to terrain 50-5
extracting props from 50-24
placement of new entities 19-3
profile. See terrain profile
prop
adding 50-24
extracting 56-20
list 50-28
querying 55-3
saving 50-5
scale 48-18
server. See terrain server
shape 47-3
simulation object group 5-5
soil type 48-17
streaming 47-9
tutorial 56-2
underwater 50-19
wireframe mode 48-11
Terrain Contents page 50-5, 50-12, 50-15, 50-16, 51-3, 56-3, 56-21
terrain database 1-13, 47-6
changing for a scenario 12-8
configuring A-6
path to A-4
loading 4-21
at startup 5-10
provided with application 48-18
Terrain Information Panel 48-15
terrain profile
concepts 49-2
configuring 49-3
displaying, line creation 35-6
line 33-3, 49-4
opening automatically 49-3
sampling frequency 49-3
terrain profile line 49-4
track history 37-8
vertex, showing 49-3
Terrain Profile Settings page 41-12
Terrain Profile window 49-2, 49-3
intervisibility line 41-12
terrain server 47-7, 47-9, 51-2
adding 51-4
caching data 50-32
connecting through proxy 51-5
cutting in site 52-13
mask, adding 52-13
Terrain Settings dialog box 48-13
Add Props page 50-26, 56-9, 56-13
Edit Existing Props page 50-28, 50-29, 50-30, 56-24, 62-10
Extract Ocean page 50-15
Extract Props page 50-25, 56-21
Navigation Areas page 53-3
Raster Maps page 50-7, 50-10, 56-5
Simulation Ocean Settings page 50-20
SpeedTree Randomization page 59-21
Terrain Contents page 50-5, 50-12, 50-15, 50-16, 51-3, 56-3, 56-21
Visible Surfaces page 48-3
terrain-check parameter 25-30
Terrain-Database parameter 12-5
terrainDatabase parameter A-6
terrain-following, fixed-wing entity 25-25
TerraPage file 47-3
TerrraPage file 51-5
testing, amount of resources 31-13
Tether Location Track mode 45-12
Tether mode 44-8, 44-25, 45-6, 45-11
Tether Track mode 45-12
text
creating 35-4
in title bar 74-8
object, resizing 35-15
sending 26-57
shadow 18-17

text editor 28-4
  scripted task, specifying 28-36
texture 59-22
  atlas, buoy and beacon 52-23
  compression, enabling and disabling 50-33
  coordinate system for image 50-9
  data, converting 50-36
  DDS 50-34
  flipping, paged terrain 50-35
  height, ocean planar reflection 38-8
  map 1-16, 50-20
  sharing 58-7
  size, ocean height map 50-17, 50-18
  tactical graphic 64-46, 64-47
  tiled and virtual 54-3
  water 50-13
  width, ocean planar reflection 38-8
thermocline, sonar 9-17, 11-20
thickness, contour line 48-10
this, Lua object 29-4
threat ring 18-25
threshold
  data updates A-4
  DIS heartbeat A-7
  object timeout A-7
tick
  rate, tuning 6-12
tidal current
  direction 11-17
  speed 11-17
  speed and direction 11-15
tidal stream, wake 11-18, 59-16
TIFF 47-4
tiled texture 54-3
time
  evaluating in plan 31-13
  scenario
    end 7-35
    ending 7-7
  setting 11-2, 11-4
  setting in new scenario 7-7
  simulation 3-10, 7-12
    default 7-12
  zone 7-12, 11-4
    setting 11-4
time constrained 4-18
time management
  concepts 4-18
  configuring 4-19
time management (continued)
  enabling 4-19
    for back-end 4-20, 5-18
  RTI requirements 4-19
  setting simulation speed 7-24
Time Multiplier parameter 7-6
Time Multiplier toolbar 12-5
time of day 3-10, 11-2
  illumination 71-12
time regulating 4-18
time stamp 5-6, 5-10
time stamp order, configuring A-7
time stamp ordered message 4-19
time zone 11-2
timed fuse 30-14
time-multiplier parameter 12-5
timeout
  DIS 5-8
  interval 5-7
  object A-7
timestamp 5-6
timing, spawn pattern 20-11
tire track 37-3
title bar
  text, adding 74-8
To Echelon Level 24-7
toolbar
  adding task or set 75-6
  Attachments 45-2
  button 4-12
  configuring 75-2, 75-6
  Display Settings 18-16
  displaying and hiding 74-3
  element 75-6
  Environment Settings 11-3, 11-4
  icon, setting 75-7
  locking 75-7
  Map Scale 48-18
  Objects 17-6, 19-7, 19-10, 23-6, 24-4
  Quick Launch 8-10
  Scenario 7-34
  Scenario Event 8-2
  Simulation Control 7-24, 7-25, 7-26
  Simulation Objects Palette 16-5
  Simulation Time Scale 7-24
  task, adding scripted task 28-22
toolbarrow, element 75-6
topmark, buoy and beacon 52-27
topmark_attr_map.csv 52-27
topological model 3-12

TOPSHP attribute 52-27
torpedo 27-6
  anti-submarine 26-32
  launching 26-34
tracer 30-43, 37-8
Tracer Use dialog box 30-43
Tracer Use set data request 30-43
track, radar mode 30-33
track history
  configuring 37-6
  displaying 37-5
  enabling and disabling for all simulation objects 37-5
  length 37-6
  terrain profile 37-8
Track History Settings page 37-7
Track mode 44-8, 45-6, 45-13
tracking
  ammunition 30-8
  entity, resources 18-30
  equipment 30-19
  resource 30-34
  weapon 30-44
tracking beam 30-17
traffic
  generating 20-2
  spawn pattern 20-5
trail, model 37-4
trailing effects 37-3, 58-2
  enabling and disabling 37-4
  model 37-4
  simulation object 37-3
trajectory, smoothing 6-16
trajectory history. *See* track history
transform, multiple 55-5
transforming, attribute 55-4
transient intervisibility
  displaying 41-6
  object 41-5
transition time
  posture, editing 66-21
translating, GUI text 2-11
translation 66-26
transmitter, publishing radio 70-13
transparency 59-22
  prop, setting 50-29
  surface 11-5, 11-15, 11-17, 73-17
transponder, IFF 30-24
trap, creating 27-14
Travel, posture 22-4
tread track 37-3
tree
  prop type 50-30
  setting avoidance 30-11
  SpeedTree 59-18
triangle, vertex label 33-8
trigger 31-7
  adding to plan 32-4
  deciding order in plan 31-15
  reactive task 25-11
  registration 31-8
  reregistering 31-16
  taskless 31-15
  unregister 31-8
Triton Ocean SDK 1-21
TrueType font, 2D icon 59-4
tuning
  components 6-12
  network performance 6-14
  performance
    component 6-12
    network interface 6-13
    state repository 6-13
  visual performance 6-14
turn rate 26-29, 26-30
Turn to Heading dialog box 26-61
Turn to Heading task 26-61
turret, attaching to 45-4
tutorial
  adding model 62-2
  composing terrain 56-2
  extract props 56-20
  starting VR-Forces 4-10
TXP file 47-3
type
  adding 66-26
  prop 50-30
type attribute 59-6

U

-u command-line option 54-4
UI file 68-23
  variable binding, adding 68-24
unconstrained 4-18
underFireDistance parameter 30-36
underFireTime parameter 30-36
underwater, visibility 11-15, 11-17
Underwater Acoustics PDU 9-17
undo, ellipse rotation 35-16

undocking, panel 74-2
undoing
  category changes 64-49
  simulation object edits in Simulation Object Editor 64-8
unexploded ordnance, destroying 27-20
unique
  buoy 52-22
  ID 15-3
unit B-8
  activity 30-6
  adding entity to 23-7
  adding simulation object to 30-41
  aggregated state 23-9
  aggregate-level 23-5
    aggregated 27-5
  ammunition, weapons, equipment 66-5, 66-8
  artillery 27-25
  assigning tasks 25-10
  attrition 22-11
    from combat engineering object 22-10
  bounding box, Simulation Object Editor 65-11
  bounding volume, color 24-11
  changing state manually 23-12
  closing formation 21-5
  collapsing 24-3
  collapsing to root 24-3
  convoy 26-8, 26-9
    behavior 25-10
  count of 18-37
  creating 23-2, 23-3, 65-4
    preconfigured 23-5
  deleting 23-13
  disaggregated and aggregated 21-2
  disaggregated state 23-9
  disaggregation area 23-12
  display 74-9
  displaying by level 24-4
  displaying ghosted icons 24-6
  echelon ID 21-3
  editing 65-4
  embark task 26-18
  entity, individual tasks and plans 23-11
  entity-level 21-2
  expanded and collapsed 21-2
  expanding 24-3
  expanding all 24-3
  expanding and collapsing 24-2
  footprint 22-4
unit (continued)
  formation 21-4, 30-22, 70-8
    adding 65-9
    automatic layout 65-13
    copying 65-12
    default 65-10
    editing 65-8
    manual layout 65-13
    removing 65-10
    renaming 65-10
    snap to grid 65-14
  information 18-3, 18-7
  introduction 23-8
  measurement 74-9
  movement 21-4, 22-6
    affected by combat engineering objects 22-9
  moving to formation 26-37
  new 66-3
  organization concepts 21-3
  parameter type 68-25
  posture 22-4
  preconfigured 23-5
  removing simulation object from 23-8, 30-41
  reorganizing 21-4, 21-5, 30-33
  restoring 23-10, 30-35
  restricting movement 55-9
  sector 18-25, 22-4
  selecting 23-6, 24-2
  sensor 22-8
  setting formation in plan 31-16
  specifying as Aggregate As option 65-3
  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 30-43
    showing in GUI 23-9
  subordinate
    adding 65-5
    changing order 65-6
    editing 65-7
    in condition 32-7
    removing 65-5
    replacing 65-6
  target, preferred 30-32
  task behavior 21-5
  template 66-3

unit (continued)
writing plan for 23-12, 32-22
Unit Display Settings page 24-7, 24-10, 24-11, 37-11, 37-17
Unit State dialog box 30-43
Unit State set data request 30-43
Universal Transverse Mercator 47-6
unloading, entities from entities 19-7
unlocking, overlay 34-4
unpublishing, tactical graphics 35-7
unregistering, trigger 31-8
unselecting
object 17-7
simulation object 17-7
update
frequency of A-4
messages 43-11
rate, intervisibility 41-9
Update_EW_Degradation, script 22-18
updater 59-10
AOA 59-12
attribute 59-10
FeetAGL 59-12
FeetMSL 59-12
FeetPerMinute 59-12
Headlight 59-13
KilometersPerHour 59-12
KnotsPerHour 59-12
MilesPerHour 59-12
Pitch 59-12
Roll 59-13
Value 59-13
Yaw 59-13
updating, platforms 63-17
upgrades lii
upgrading, SMS 63-13
Use Extracted Geometry 50-24
useAbsoluteTimeStamps parameter A-7, A-8
use-actual-message-size parameter 13-3
useAdvisories parameter A-7
useAsyncIO parameter A-8
useCustomMapDatum parameter A-6
Used By Countries list 64-8
USE-DEFAULT keyword 12-10
useDIGuy parameter A-6
useIpv6 parameter A-8
use-logger-control parameter 7-38
user
settings 4-25
task 26-62
user data, specifying directory 5-6
User Task dialog box 26-62
user-defined, formation 70-10
using
Boundary Generation Tool 52-14
navigation data 53-8
UTM 47-6
coordinate system 44-7
database 7-3
UUID 15-2
scheme 5-5
simulation object 15-3
UV, coordinates 50-9

V

-v command-line option 5-6, 5-10, 5-13, 54-4
Value updater 59-13
vapor trail 37-3
variable, tick rate 6-12
variable binding 68-21
adding to platform file or system definition file 68-22
adding to Simulation Object Editor 68-21
applying changes 63-17
platform file 68-21
Simulation Object Editor, adding new 68-23
variable frame mode 6-17
variable-frame keyword 12-6
variable-frame run-to-complete 3-11, 12-6
variance, spawn interval 20-6, 20-8
varying, heartbeat A-7
vector
geocentric 29-20
Lua 29-18, 29-20
network 25-21
vector feature
configuring avoidance of 70-16
list 48-15
Vector3D class 29-18
VectorGeoc3D class 29-20
VectorOffset3D class 29-20
vehicle, model 1-7
velocity
displaying 18-12
specifying measurement unit 74-9
vel-to-align-actual parameter 25-31
vel-to-align-desired parameter 25-31
version
displaying 5-6

version (continued)
plug-in, checking 5-3
RPR FOM 5-7, 5-11, A-7
version parameter, scenario file 12-6
vertex
adding 35-13
altitude, displaying 33-9
deleting 35-15
editing 35-14
line, terrain profile 49-3
showing and hiding 33-8
video, adding to scenario event 8-5
video card, processor speed 6-4
view
clearing list 44-31
constraint 44-27
default 44-31
deleting 44-31
dragging with mouse 44-15
floor 44-27
inset 46-2
loading saved 44-32
navigating 44-3
observer, transitioning between 44-30
recalling 44-29
restoring 44-30
saved, scenario 12-2
saved with scenario 6-1
saving 44-29, 44-30, 44-31
zooming 44-18
2D 44-20
3D 44-19
view constraint, disabling 19-4
view control, sending in plan 32-19
view control message 44-32
enabling processing of message 44-32
script 44-32
view mode
Absolute 45-7
Follow 45-7
Mimic 45-8
Mimic Location Track 45-9
Mimic Track 45-9
Space Follow 45-10
Tether 45-11
Tether Location Track 45-12
Tether Track 45-12
Track 45-13
viewing
current task 31-14
viewing (continued)
list of props 50-28
Logger files 7-42
obstructed terrain in terrain profile 41-12
other MAK 3D GUI viewers 43-12
plan 31-14
plug-in, data 4-39
viewpoint
spot report 9-4, 9-5
custom 9-6
viewport 73-8
changing 73-16
virtual texture 54-3
Virtual Texture Datasets 54-2
visibility 11-8, 11-10
modifier 22-10
observer 11-5
underwater 11-5, 11-15, 11-17
water 73-17
visible surface, configuring 48-2
Visible Surfaces page 48-3
Visit Interest Points reactive task 20-22
visual definition 57-3, 57-14, 57-16
adding 57-16
attribute 57-17
adding 57-19
editing 57-17, 57-18
Entity Image Symbol 59-2, 59-4, 59-9
Military Symbol Icon 59-2
schema 57-3
visual model
adding 64-20
details 64-21
editing 64-16
tactical graphic 64-46
Visual Model Editors dialog box 57-8
visual quality 50-17
visual quality slider, configuring 6-15
visual sensor 71-12
visualizer 57-14
visualizing, tasks 37-12
volume
sound, adjusting 42-3
volumetric, tactical graphic 35-4
vrf class 29-4
vrf-base-object-param parameter 68-26, B-3
vrf-base-object-param parameter type B-7
vrfGui 3-2
vrfGui.log file 4-40

vrfLauncher
  combined mode 5-14
  special command-line arguments 5-14
vrf-moving-object-state-repository 68-12
vrfMsgDump 1-19
vrf-object-param parameter B-7
vrf-object-param parameter 68-12
vrf-object-state-repository 68-12
VR-Forces
  entity or object 3-9
  Launcher 4-4
  starting 4-3
  window 4-12, 4-21
VR-Forces window 7-7
vrf-overlay-object-state-repository 68-12
vrfSim
  log file 5-9
  running as daemon A-3
vrfSim 3-2
  daemon, starting as 5-8
vrfsim
  enabled 52-18
  layer 55-2
vrfsim: 52-17
vrfSim.log file 4-40
vrfSim.mtl file A-3
vrfSim.opd
  remote-configurations block 70-19
  required in every SMS 63-9
vrfSimSettings.xml file 6-14
vrfutil, functions 29-6
VR-TheWorldOnline, database 48-19
VR-Vantage, view control 32-19
VR-Vantage Control Toolkit 44-32
VR-Village, database 48-19
VSync 6-19
VTDS 54-2
vulnerability 22-3
  modifier 22-10

W

-w command-line option 54-5
wading 30-31
Wait Duration dialog box 26-63
Wait Duration task 26-63
Wait Elapsed dialog box 26-63
Wait Elapsed task 26-63
Wait task 26-62
wake 1-15, 1-21, 37-3
  enabling and disabling 39-5
  model definition 11-18
  ship, configuring 59-14
  tidal stream 11-18, 59-16
wakeLength parameter 11-18
walking 30-31
wall, not walking through 44-27
wall-clock time 4-19
wander, crowd 26-64
Wander In Area task 26-64
Wander task 26-63
warfare, electronic 22-16, 22-17, 22-18, 22-19
warfare model, aggregate 22-2
warhead parameter 70-3
warning message A-4
water
  pacing and tracking 30-20
  splash 39-2
  texture 50-13
    extracting 50-15
  visibility 73-17
wave
  breaking 11-19
  choppiness 11-15
  effect 39-2
waypoint 33-2, 33-4
  altitude, displaying 33-9
  ordering entity to 26-42
  patrolling between 26-46
weapon
  anti-air 27-7
  fire, sound mapping 42-8
  laser guided 26-32
  pacing and tracking 30-44
  status 18-4
  system 64-22, 64-23
  system definition file 68-16
  unit 66-5, 66-8
Weapons Pacing/Tracking dialog box 30-44
Weapons Pacing/Tracking set data request 30-44
weather 11-5
  setting 11-8
Weather dialog box 11-14
Weather page 11-6, 11-7, 11-8, 11-9, 11-10, 11-11, 11-16, 73-17
weather zone 11-12
  priority 11-12
When statement 31-4, 31-7
  adding to plan 32-4

When statement (continued)
building 32-4
While statement 31-4, 31-9
WidgetTheme, attribute 59-6
width
changing for line 35-17
sensor contact line 37-15
wildcard 68-7
wind 1-15
direction 11-5
moving flags and windsocks 59-16
offset 10-2
speed 11-5
speed and direction 11-8, 11-9
WindDirPart 59-17
window
Add Visualizer Definition Attribute 57-19
adding 73-3
attributes, editing 73-7
channel, adding 73-5
Event 8-10, 8-12
location, changing 73-7
Multiple Object Information 18-10
new 4-14
observer 43-6
printing 4-14
property 73-7
removing 73-6
secondary 4-14
selecting simulation objects in 17-5
size, changing 73-7
specifying 5-4
Terrain Profile 49-2
types 73-2
Window Name attribute 73-7
Window Type attribute 73-7
Windows, installing on 2-2
windsock, movement in wind 59-16
WindSpeed parameter 59-17
WindState parameter 59-17
WindSwitchPart 59-17
wireframe mode 48-11
world coordinates 44-3
writing
entity plan 32-2
global plan 32-11
plan
for multiple simulation objects 32-22
for unit 23-12
plans, remote entities 32-22
WRM Entity Model Specification 61-1, 61-9
and meta file 59-16

X-Y-Z

-x command-line option 5-7, 5-11, 5-12, 5-16, 5-20, 54-5
XML 52-3
include files 52-5
RTI configuration file 2-10
XML file
plug-in 4-36
deleting 4-39
XR mode 43-10
XR scaling 19-24
xtr file 12-10
yaw 44-3
Yaw updater 59-13
yellow icon 9-13
Z Far 73-8
Z Near 73-8
Z-fighting 45-6, 50-13, 73-11, 73-17
zone
time 7-12
weather 11-12
zoom
scale factor, LOD 44-20
scenario, on load 44-21
SpeedTree LOD 5-6
switching from 3D to 2D 43-2
to entity extents 44-21
to extents 44-20
zooming
observer 44-18
2D 44-20
3D 44-19
sensors 30-38

VR-Forces Quick Reference Card

Table B-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.
--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.
--disableCallbackQueue	Disables all main thread parallel algorithms.
--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 SISO enumerations.
(-n | --notifyLevel) *notification_level*	Sets the level for messages written to the console or the VR-Forces log file (*vrf.log*).
--noRtiCompilerCheck	Disable RTI Compiler version check.
--nonVrfObjectsUUIDScheme *scheme*	Sets the UUID scheme for non vrf objects. Values are `entity-identifier`, `global-identifier`, `marking-text`.

Copyright © 2016 VT MÄK. All rights reserved. Document ID: VRF-4.4.1-QRC-160609

Table B-1: vrfSim Command-Line Options

Option	Description
`--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.
(`-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.
(`-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.
`--disableParallelPreprocess`	Disables parallel processing for components that use it.
(`-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)
(`-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)
`--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.
`--defaultObjectTimeout` *timeout*	Specifies the timeout interval for objects.

Table B-1: vrfSim Command-Line Options

Option	Description
`--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 © 2016 VT MÄK. All rights reserved. Document ID: VRF-4.4.1-QRC-160609

Table B-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.
--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.
--entDispSetting *filename*	Specifies the entity display setting to load.
--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	Set the root directory used to restore settings to factory defaults.
--fileTransporterReceivePort *port*	Port for the file transporter.
--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.
--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.

Table B-2: vrfGui Command-Line Options

Option	Description
`--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.
(`-n` | `--notifyLevel`) *notify_level*	Specifies the notification level for application messages. Range: 0-4.
`--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-Forces front-end with only 2D modes available.
(`-S` | `--suppressHatIntersectOnAttach`)	Suppress the periodic HAT intersection when attached.
`--scenarioFile` *filename*	Specifies a scenario to load.
`--stealth`	Start the VR-Forces 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` | `--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.

Copyright © 2016 VT MÄK. All rights reserved. Document ID: VRF-4.4.1-QRC-160609

Table B-2: vrfGui Command-Line Options

Option	Description
`--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.
`--useGeographicDdm`	Specifies use of geographic DDM if you are using the MÄK 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 B-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.

Table B-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.

K = numeric keypad (K8 means press Keypad 8). To use the numeric keypad for navigation, NumLock must be on.

Table B-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**

Copyright © 2016 VT MÄK. All rights reserved. Document ID: VRF-4.4.1-QRC-160609

Table B-4: Observer Keyboard Mappings

Observer Movement	Observer	Level Observer	Laptop	2D Level Observer
Move up level observer.	unmapped	**q**, **Ctrl**+**q**	unmapped	unmapped
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

Table B-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

Copyright © 2016 VT MÄK. All rights reserved. Document ID: VRF-4.4.1-QRC-160609

Table B-4: Observer Keyboard Mappings

Observer Movement	Observer	Level Observer	Laptop	2D Level Observer
Toggle Tactical Graphics HAT Lines	unmapped	unmapped	unmapped	unmapped
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**+**[**

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.
`(-h | --help)`	Displays help information in a console window.
`--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--`.



Link - Simulate - Visualize

VRF-4.4.1-1-160616

150 CAMBRIDGE PARK DRIVE, 3RD FLOOR CAMBRIDGE, MA 02140 617.876.8085 www.mak.com