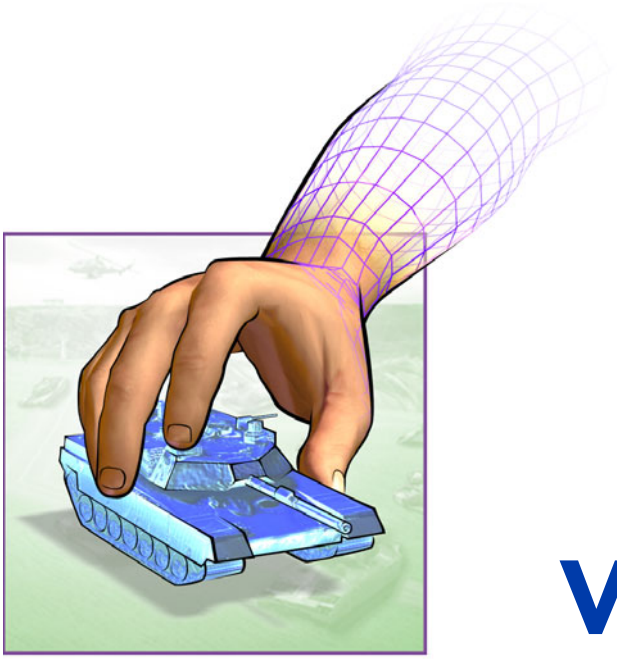
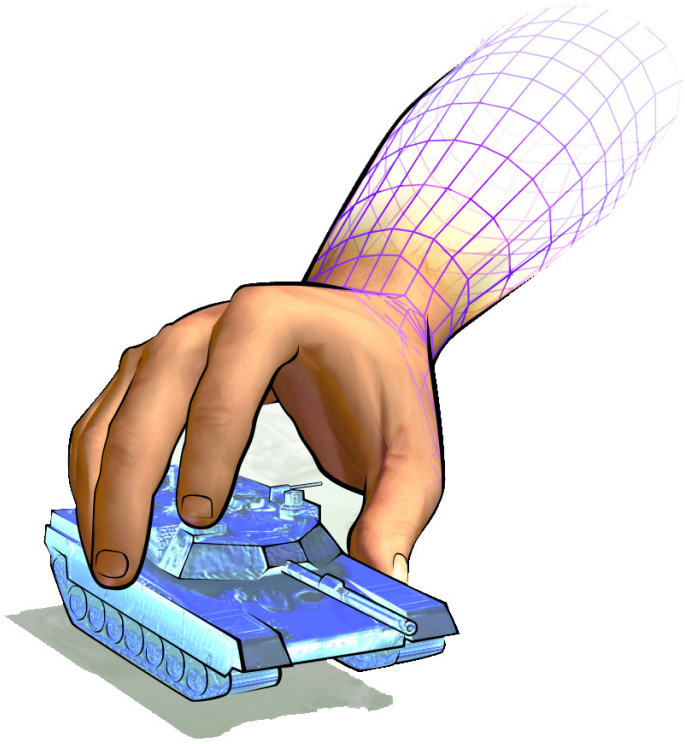


VR-Forces

Configuration Guide

VT MÄK
A company of VT Systems



VR-Forces

Configuration Guide

VT MÄK
A company of VT Systems

Copyright © 2012 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™, and VR-Vantage™ 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 xx.

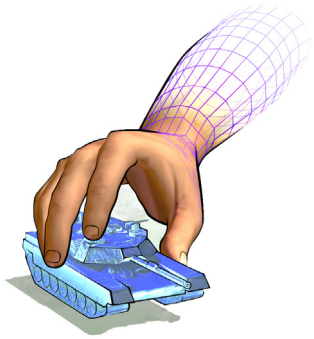
VT MÄK
68 Moulton St.
Cambridge, MA 02138 USA

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

info@mak.com

www.mak.com

Revision VRF-4.0.4-7-120613



Contents

Preface

How This Manual is Organized xiii
VR-Forces Documentation xv
MÄK Products xvi
How to Contact Us xviii
Document Conventions xix
Mouse Button Naming Conventions xx
Third Party Licenses xx
Boost License xx
libXML and libICONV xxi
pThreads Library xxi
EHS, PCRE, and PME xxi
Freefont OpenType Font Set xxi

Chapter 1 Optimizing Performance

1.1. Introduction 1-2
1.1.1. Optimizing Simulation Engine Performance 1-2
1.1.2. Optimizing Front-End Performance 1-3
1.2. Monitoring the Back-end Frame Rate 1-4
1.3. Displaying Performance Statistics 1-4
1.4. Filtering Entities Using Interest Management 1-6
1.4.1. Enabling Interest Management 1-7
1.4.2. Configuring Interest Management 1-7
1.5. Miscellaneous Front-End Performance Configuration Options 1-9
1.5.1. Limiting Use of Spot Reports 1-9
1.5.2. Using Asynchronous I/O for DIS Exercises 1-9
1.6. Setting the Tick Rate 1-10
1.6.1. Specifying the Tick Rate for Components 1-10
1.6.2. Tuning the Network Interface 1-11

1.6.3. Tuning the State Repository Tick Rate 1-11
1.7. Balancing Visual Quality Against Network Performance 1-12
1.7.1. Enabling Configuration of Performance Options 1-13
1.8. Tuning the Target Frame Rate 1-13
1.9. Configuring VSync 1-13

Chapter 2 VR-Forces Configuration Files

2.1. Introduction 2-2
2.2. The *vrfSim.mtl* Configuration File 2-2
2.3. Configuring DI-Guy Integration 2-8
2.4. Configuring Communication Models 2-9
2.4.1. Simple Radio Communication Model 2-9
2.4.2. External Communication Model 2-9
2.4.3. Custom Communication Models 2-10
2.5. MÄK Technologies Lisp (MTL) Files 2-10
2.5.1. Using Environment Variables in an MTL File 2-11
2.5.2. Specifying Lists of Lists 2-11
2.5.3. Using Conditional Statements 2-11

Chapter 3 Scenario Files

3.1. Introduction 3-2
3.2. The Scenario File 3-3
3.2.1. Scenario Parameters 3-4
3.2.2. Specifying Pathnames for Scenario Files 3-7
3.2.3. Changing the Terrain Database for a Scenario 3-8
3.3. Editing the Plan Files 3-8
3.3.1. The Structure of an Entity Plan File 3-9
3.4. The Object Map File 3-11
3.5. Editing the Order of Battle File 3-12
3.5.1. Saving Default Entity Parameters to the Order of Battle File 3-12
3.6. The Scenario Extras File 3-12
3.7. Temporary Scenario Directories 3-13

Chapter 4 Editing Entity Simulation Models

4.1. Introduction to the Entity Editor 4-3
4.1.1. Simulation Model Sets 4-4
4.1.2. Starting the Entity Editor 4-5
4.2. Creating a New Simulation Model Set 4-6
4.2.1. Creating a Completely New Simulation Model Set 4-7
4.2.2. Specifying the Default Simulation Model Set for Scenarios 4-7
4.3. Opening a Simulation Model Set 4-8
4.4. Editing an Entity Simulation Model 4-9
4.4.1. Selecting an Entity to Edit 4-10

4.4.2. Editing an Entity's Category 4-11
4.4.3. Editing an Entity's Force 4-13
4.4.4. Changing an Entity's Entity Enumeration 4-14
4.4.5. Changing an Entity's Name 4-15
4.4.6. Specifying Whether an Entity Can Be Created 4-15
4.4.7. Editing an Entity's Bounding Volume (Size) 4-16
4.4.8. Editing Behavioral Parameters 4-18
4.4.9. Editing a Lifeform's DI-Guy Character, Appearance, and Animation 4-19
4.4.10. Undoing (Reverting) Changes to an Entity 4-21
4.5. Editing an Entity's Visual Model 4-21
4.5.1. Adding a New Visual Model or Image 4-24
4.6. Editing Aggregates 4-24
4.6.1. Adding Subordinates 4-26
4.6.2. Removing Subordinates 4-27
4.6.3. Replacing Subordinates 4-27
4.6.4. Changing the Order of Subordinates 4-27
4.6.5. Editing a Subordinate Aggregate 4-28
4.6.6. Adding Formations 4-29
4.6.7. Removing Formations 4-29
4.6.8. Renaming Formations 4-29
4.6.9. Specifying the Default Formation 4-30
4.6.10. Expanding and Collapsing the Display of Formations 4-30
4.6.11. Displaying Bounding Boxes 4-31
4.6.12. Copying Formations 4-32
4.6.13. Laying Out a Formation Automatically 4-33
4.6.14. Changing a Formation's Layout by Hand 4-33
4.7. Editing an Entity's Systems 4-35
4.7.1. Adding a System 4-37
4.7.2. Selecting a Damage Model 4-37
4.7.3. Selecting a Movement Model 4-37
4.7.4. Editing a System's Properties 4-38
4.7.5. Editing a System in the OPD Editor 4-38
4.7.6. Removing a System 4-38
4.8. Creating a New Entity 4-39
4.8.1. Creating a New Entity from an Existing Entity 4-39
4.9. Deleting an Entity Model from a Model Set 4-40
4.10. Navigating Among Edited Entities 4-40
4.11. Editing the Create Menu 4-40
4.11.1. Generating the Create Menu Automatically 4-41
4.11.2. Editing the Create Menu Manually 4-41
4.11.3. Undoing (Reverting) Changes to the Create Menu 4-44
4.12. Editing the Category List 4-44
4.12.1. Adding a New Category 4-44
4.12.2. Removing a Category 4-45
4.12.3. Renaming a Category 4-45

4.12.4. Undoing Category Changes 4-45
4.13. Managing the Forces List 4-46
4.13.1. Adding a Force 4-46
4.13.2. Editing a Force 4-47
4.13.3. Removing a Force 4-48

Chapter 5 Advanced Entity Model Editing

5.1. The Object Parameter Database Editor 5-2
5.2. Starting the OPD Editor 5-3
5.3. Loading an Object Parameter Database 5-4
5.3.1. Changing the Object List View 5-5
5.4. Editing an Object's Parameters 5-5
5.4.1. Editing Individual Parameters 5-6
5.4.2. Editing Parameters for Multiple Objects 5-7
5.4.3. Deleting Parameters and Components 5-7
5.4.4. Restoring Parameters 5-7
5.5. Creating a New Object from an Existing Object 5-8
5.6. Adding New Components and Parameters 5-9
5.6.1. Adding New Components 5-9
5.6.2. Editing Components 5-10
5.6.3. Adding New Resource Parameters 5-11
5.6.4. Adding New Subordinate Object Parameters 5-12
5.6.5. Adding New Soil Type Parameters 5-12
5.7. Editing Variables 5-13
5.8. Editing Systems 5-13
5.9. Saving Changes to the Object Parameter Database 5-14

Chapter 6 The Object Parameter Database

6.1. The Object Parameter Database 6-2
6.1.1. VR-Forces Classes and Object Parameters 6-6
6.1.2. Object Types 6-6
6.1.3. Parameter Types 6-9
6.1.4. Local and Remote Subcomponents 6-10
6.2. Component Descriptors and Parameters 6-13
6.2.1. Common Elements of Component Descriptors 6-14
6.2.2. Configuring the Priority of Components 6-16
6.3. Adding a New Entity to the Object Parameter Database 6-17
6.4. System Definition Files 6-18
6.4.1. Creating a System Definition File 6-19
6.4.2. Configuring System Connections 6-19
6.4.3. Adding Variable Bindings 6-22
6.5. Configuring Aggregate Entities 6-23
6.5.1. Adding an Aggregate to the Object Parameters Database 6-23
6.6. Changing the Echelon ID Category and Echelon ID Level 6-26
6.7. Configuring Tactical Graphics 6-27

6.8. Pathname Interpretation in Simulation Model Set Files 6-27
6.9. Updating Object Parameter Database Files for New Releases 6-28

Chapter 7 Configuring Object Behaviors

7.1. Configuring Sensors 7-3
7.1.1. Configuring a Sensor in an OPE File 7-3
7.1.2. Editing a Sensor System Definition File 7-4
7.1.3. Specifying a Sensor Signature 7-5
7.1.4. Adding a New Sensor Type 7-5
7.2. Detection Probability Tables 7-8
7.3. Probability and Selection Tables for Munitions 7-9
7.3.1. Ammo Select Tables 7-9
7.3.2. Hit Probability Tables 7-10
7.3.3. Damage Probability Tables 7-12
7.3.4. Configuring an Entity to Take Damage from a Munition 7-13
7.4. Configuring Formations 7-14
7.4.1. Formation Files 7-15
7.4.2. Creating a User-Defined Formation 7-16
7.5. Configuring Emitters 7-17
7.6. Configuring Radios 7-19
7.7. Configuring Sensor Volumes 7-19
7.7.1. Configuring Sensor Volume Color 7-20
7.7.2. Controlling Sensor Volume Radius 7-21
7.7.3. Configuring Sensor Volume Segments 7-22
7.8. Configuring Task Execution Rules 7-25
7.9. Configuring Joysticks 7-26
7.10. Configuring Obstruction Avoidance 7-27
7.10.1. Configuring Object Types 7-27
7.10.2. Configuring Vector Feature Types 7-28
7.11. Configuring Embarkation 7-29
7.11.1. Editing Embarkation Parameters 7-30
7.12. Configuring VR-Forces Components on Remote Entities 7-32

Chapter 8 Using VR-Forces with an External Comms Effects Server

8.1. Introduction 8-2
8.2. Configuring VR-Forces 8-2
8.2.1. Enabling Radio Transmitter Publishing 8-3
8.2.2. Enabling the External Communication Model 8-3
8.2.3. Configuring the External Communications Model 8-4
8.3. Using VR-Forces with Qualnet 8-5
8.3.1. Running VR-Forces with Qualnet 8-5
8.4. Example Scenario 8-6
8.4.1. Running the Scenario 8-6

Chapter 9 Display Engine Configurations

9.1. The Display Engine 9-2
9.1.1. Window Types 9-2
9.1.2. Adding a Window 9-3
9.1.3. Adding a Channel to a Window 9-5
9.1.4. Removing a Window 9-6
9.1.5. Removing a Channel 9-6
9.1.6. Saving a Display Engine Configuration 9-6
9.1.7. Loading a Display Engine Configuration 9-6
9.2. Changing a Window's Attributes 9-7
9.3. Changing a Channel's Attributes 9-8
9.3.1. Setting the Clipping Planes 9-10
9.3.2. Specifying the Projection Resize Policy Attribute 9-11
9.3.3. Changing a Channel's Frustum (Field of View) 9-14
9.3.4. Changing the Viewport 9-15
9.4. Configuring Multichannel Displays 9-16
9.4.1. Changing the Camera's Position and Orientation Offset 9-19
9.5. Stereoscopic Displays 9-20
9.5.1. Configuring Anaglyphic Stereo 9-21
9.5.2. Configuring Polarized Stereo 9-22

Chapter 10 Composing Terrains

10.1. Creating a Composed Terrain 10-3
10.1.1. Considerations and Limitations for Building Terrains 10-3
10.1.2. Saving a Terrain 10-4
10.2. Adding Elevation Data (Terrain Patches) to a Terrain 10-5
10.3. Adding Images to a Terrain 10-6
10.3.1. Changing the Display Order of Raster Maps 10-9
10.4. Adding a Feature Layer 10-10
10.5. Connecting to Terrain Servers 10-11
10.5.1. Adding Terrain Server Connections 10-13
10.5.2. Editing a Terrain Server Configuration 10-15
10.6. Loading MetaFlight Terrains 10-15
10.7. Preprocessing Paged Terrains 10-15
10.8. Manually Paging-In Terrain 10-16
10.9. Adding Props to a Terrain 10-17
10.9.1. Extracting Props from a Terrain Patch 10-17
10.9.2. Adding Props from a Feature Layer 10-19
10.9.3. Viewing a List of Props 10-21
10.9.4. Selecting Props 10-21
10.9.5. Setting the Opacity of Props 10-22
10.9.6. Changing a Prop's Type 10-22
10.9.7. Changing a Prop's Position or Orientation 10-23
10.9.8. Changing a Prop's Model Definition 10-23
10.10. Configuring File Caching 10-24

10.10.1. Caching Terrain Server Data 10-24
10.10.2. Enabling Texture Compression 10-25
10.10.3. Clearing the File Cache 10-25
10.11. Displaying DDS Textures Correctly 10-25
10.11.1. Flipping DDS Textures Globally 10-26
10.12. Displaying Information About Paged and Streaming Terrains ... 10-27
10.12.1. Displaying Additional Terrain Paging Information 10-28
10.12.2. Configuring Terrain Paging for the Back-End 10-29
10.13. Building Efficient MetaFlight Terrains for VR-Forces 10-30

Chapter 11 Mapping Models and Effects

11.1. Mapping Models to Entities and other Scene Elements 11-2
11.1.1. Adding a Model Mapping 11-3
11.1.2. Editing a Model Mapping 11-4
11.1.3. Filtering the Element Definition List 11-5
11.1.4. Deleting a Model Mapping 11-5
11.1.5. How VR-Forces Maps DI-Guy Models 11-6
11.2. Clearing the Model Instancing Cache 11-6
11.3. Compressing Model Files 11-7

Chapter 12 Model and Element Definitions

12.1. Creating and Editing Schemas 12-3
12.1.1. Creating Schemas 12-3
12.1.2. Deleting Schemas 12-4
12.1.3. Copying Schemas 12-5
12.1.4. Adding a Parameter to a Schema 12-5
12.1.5. Deleting a Parameter from a Schema 12-7
12.1.6. Editing a Schema Parameter 12-7
12.2. Creating and Editing Model Definitions 12-8
12.2.1. Creating a Model Definition 12-9
12.2.2. Deleting a Model Definition 12-10
12.2.3. Copying a Model Definition 12-10
12.2.4. Filtering the List of Model Definitions 12-11
12.2.5. Editing a Model Definition 12-11
12.2.6. Saving Model Definitions 12-13
12.3. Creating and Editing Element Definitions 12-13
12.3.1. Creating an Element Definition 12-14
12.3.2. Editing an Element Definition 12-14
12.3.3. Deleting an Element Definition 12-18
12.3.4. Saving Element Definitions 12-18
12.4. Configuring 2D Icons 12-19
12.4.1. Editing Font-Based 2D Icons 12-19
12.4.2. Using Images for 2D Icons 12-21
12.4.3. Adding Images for Entities 12-26
12.5. Adding New Cockpit Display Models 12-27

12.5.1. Installing a Cockpit DLL 12-28
12.5.2. Creating a Model Definition for a Cockpit 12-29
12.6. Flipping DDS Textures for a Model 12-31
12.7. Configuring SpeedTree Trees 12-32
12.7.1. Specifying the SpeedTree Performance Profile 12-33
12.8. Best Practices for Creating Models for VR-Forces 12-34

Appendix A Editing the Create Menu

A.1. The Create Menu A-2
A.2. The *vrf.ent* Configuration File A-2
A.2.1. Object Blocks in *vrf.ent* A-4
A.2.2. Reorganizing the Create Menu A-4
A.2.3. Editing Menu Items A-4
A.3. Adding New Entity Types to VR-Forces A-5
A.3.1. Adding a New Entity or Control Object to *vrf.ent* A-5

Appendix B Object Parameters

B.1. The Structure of an Entity 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. Entity 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. Aggregate Category B-10
B.4.5. Aggregate 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 Terrain Tutorials

C.1. Composing a Terrain C-2
C.1.1. Add Elevation Data (Add a Terrain Patch) C-3
C.1.2. Add Imagery C-5
C.1.3. Add a Feature Layer C-8
C.1.4. Add Props C-9
C.2. Adding a Terrain Server C-13
C.2.1. Connect to the SanLuisObispo Terrain Server C-15
C.3. Extracting Props from a Terrain C-17

C.3.1. Create a New Terrain and Add a Terrain Patch C-17
C.3.2. Extract the Props C-18
C.3.3. Change the Model Definition for a Prop C-20
C.4. Adding Feature Data to a GDB Terrain C-21
C.4.1. Create a New GDB Terrain in the TDB Tool C-22
C.4.2. Add a Point Feature to the Terrain C-25
C.4.3. Add the GDB Terrain to VR-Forces as a Feature Layer C-28

Appendix D Model Tutorials

D.1. Adding a Model to VR-Forces D-2
D.1.1. Create a Model Definition D-2
D.1.2. Add an Element Definition D-5
D.1.3. Add a Model Mapping D-8
D.1.4. Test the Model Mapping D-10
D.2. Extracting a Point Feature as a Prop (Adding a SpeedTree) D-13

Appendix E Creating and Editing Key Mappings

E.1. Introduction E-2
E.2. The Key Mapping Editor E-2
E.2.1. Binary Key Mappings E-3
E.3. Editing a Key Map E-3
E.3.1. Adding a Key Mapping E-4
E.3.2. Changing a Key Mapping E-5
E.3.3. Deleting a Key Mapping E-6
E.4. Using Combined Key Mappings E-6
E.5. Filtering the Function List E-7
E.6. Creating a Key Map E-7
E.7. Deleting a Key Map E-7

Appendix F The WRM Specification

F.1. Introduction F-2
F.2. The OpenFlight File Format F-3
F.2.1. Node Name and Comment Fields F-3
F.2.2. External References and Instancing F-3
F.3. Modeling for DIS Interoperability F-4
F.3.1. The DIS Attribute Lexicon (DAL) F-4
F.3.2. DAL Keywords F-5
F.3.3. Model Coordinate Systems F-6
F.4. Articulated Parts F-8
F.5. Entity Appearances F-9
F.6. Animation F-12
F.7. DIS Keyword Values F-12

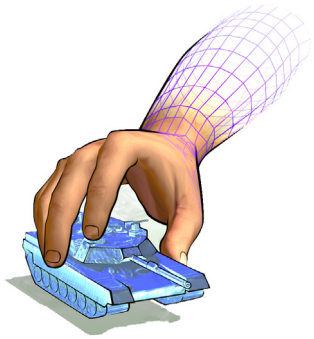
Appendix G Processing MetaFlight Files

G.1. Processing MetaFlight Files for Use in VR-Forces G-2
G.2. Splitting Datasets into Individual MetaFlight Files G-2
G.3. Convert Virtual Texture Datasets to Tiled Textures G-2
G.3.1. The mftTool G-3
G.3.2. Using the mftTool to Process MetaFlight Files G-5
G.3.3. Convert Geometry Grid Datasets with Tiled Textures G-5
G.4. Convert Source Data into MEDF Format G-6
G.5. Create an MTF File for the MetaFlight Terrain G-6

Appendix H Editing Earth Files

H.1. Earth Files H-2
H.2. A Simple Earth File H-2

Index



Preface

This manual is written for persons who want to configure or customize VR-Forces® without using the API. The manual assumes that you are familiar with the use of VR-Forces, as described in *VR-Forces Users Guide*.

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:

Chapter 1, *Optimizing Performance* describes how to monitor VR-Forces's performance in the Frame Rate Monitor and how to configure VR-Forces to improve performance.

Chapter 2, *VR-Forces Configuration Files* explains the syntax of configuration files and the parameters in the configuration files provided with VR-Forces.

Chapter 3, *Scenario Files* describes the files used to define a scenario and how to edit them.

Chapter 4, *Editing Entity Simulation Models* explains how to use the Entity Editor to edit entity parameters, systems, and symbol mappings. It also explains how to create custom Create menus.

Chapter 5, *Advanced Entity Model Editing* explains how to use the OPD Editor to edit entity and system parameters. The OPD Editor lets you edit all entity parameters, while the Entity Editor only lets you edit some parameters.

Chapter 6, *The Object Parameter Database* describes the object parameter database, which contains parameters for entities and control objects, and how to edit it.

Chapter 7, *Configuring Object Behaviors* describes additional configuration files that VR-Forces uses to manage munitions, formations, and joystick use.

Chapter 8, *Using VR-Forces with an External Comms Effects Server* explains how to configure VR-Forces to work with external communications effects servers, such as QualNet.

Chapter 9, *Display Engine Configurations* explains how to configure windows and channels and how to save and load display configurations.

Chapter 10, *Composing Terrains* explains how to build terrain databases by importing elevation data, imagery, and feature data.

Chapter 11, *Mapping Models and Effects* explains how map entity enumerations to the models configured in VR-Forces.

Chapter 12, *Model and Element Definitions* explains how to create and edit model schemas, model definitions, and element definitions.

Appendix A, *Editing the Create Menu* explains the format of the *vrf.ent* file and how to edit it to change the listings on the Create menu.

Appendix B, *Object Parameters* describes the high level parameters used in OPE files.

Appendix C, *Terrain Tutorials* provides step-by-step instructions for creating and saving a terrain database.

Appendix D, *Model Tutorials* provides step-by-step instructions for creating a model and mapping it to an entity enumeration.

Appendix E, *Creating and Editing Key Mappings* explains how to edit key mappings and create new key mappings.

Appendix F, *The WRM Specification* explains the WRM specification for 3D modeling.

Appendix G, *Processing MetaFlight Files* explains how to preprocess MetaFlight files so that you can open them in VR-Forces.

Appendix H, *Editing Earth Files* introduces the syntax for EARTH files.

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 Documentation Center* is your central starting point for accessing the VR-Forces manual set. It has a documentation roadmap, tables of contents for all manuals, and a master index, including the TDB Tool, to help you find references to subjects that are covered in two or more manuals or locate the manual in which a particular topic is discussed. All table of contents entries and index entries are live links to the manuals.
- *VR-Forces Getting Started Guide* is a quick introduction to VR-Forces. It covers the basics of installing VR-Forces, running a scenario, and creating a scenario. It focuses on helping new users avoid common mistakes.
- *VR-Forces Users Guide* describes how to install VR-Forces and configure license management. It explains how to use the VR-Forces graphical user interface to view simulations and how to manage aspects of VR-Forces that are not directly related to creating and running scenarios.
- *VR-Forces Scenario Management Guide* explains how to create and run scenarios.
- *VR-Forces Configuration Guide* explains advanced features for configuring performance and how to edit VR-Forces configuration files. It includes documentation of the Entity Editor, OPD Editor, and Scenario Merge tools.
- *VR-Forces Developers Guide* explains how to use the VR-Forces simulation and remote control APIs. It focuses primarily on the simulation engine. Provided in PDF format and as part of the HTML API documentation.
- *VR-Forces Front-End Developers Guide* explains how to extend or modify the VR-Forces graphical user interface (GUI). Provided in PDF format and as part of the HTML API documentation.
- *VR-Forces Migration Guide* collates API migration information for each release in the VR-Forces 3.x series.
- Online help. The VR-Forces front-end, the OPD Editor, the Entity Editor, and the TDB Tool have online help accessible from the Help menu.
- HTML API documentation. Class documentation and developers guides in linked HTML pages.
- *VR-Forces Release Notes*.

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 local entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. VR-Forces also functions as a plan view display for viewing remote entities 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 four end-user applications (VR-Vantage Stealth, VR-Vantage XR, VR-Vantage PVD, and VR-Vantage IG), the VR-Vantage Toolkit, and VR-Vantage FreeView.
  - VR-Vantage Stealth displays a realistic, 3D view of your virtual world. You can view this world from the inside of a simulated moving vehicle, or place the eyepoint at another moving or stationary location. The Stealth lets you switch rapidly among several predefined viewpoints while the simulation is underway.
  - 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 XR provides a common operational picture using the same 3D views as VR-Vantage Stealth, 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.
  - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.
  - 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.
  - VR-Vantage Free View is a terrain and 3D model viewer that introduces some of the features of VR-Vantage applications in a useful, free utility.
- 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.
- VR-inTerra. VR-inTerra is a C++ API for adding terrain agility to applications. It can load, page, or stream terrain from a wide variety of formats or sources into a single, consistent run-time representation, consisting of a collision graph and vector network.

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:	vrv-support@mak.com

Internet

MÄK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses.html
Product version and platform information:	www.mak.com/support/product-versions.html
For the free, unlicensed MÄK RTI:	www.mak.com/resources/bonus-material/cat_view/16-bonus-materials/24-mak-high-performance-rti.html
MÄK Community Forum:	www.mak.com/community-forum/1-forum.html

Post

Send postal correspondence to:	VT MÄK 68 Moulton St. Cambridge, MA, USA 02138

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.
Blue bold text	A hypertext link to class documentation.
`{ }`	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 UNIX 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**.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

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 *vrforces/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 popup 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 which 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/

pThreads Library

VR-Exchange links with the pThreads win32 library. The library is distributed under the GNU Lesser General Public License (LGPL). MÄK has made no modification to this library. For information about the pThreads win32 library please see: http://sourceware.org/pthreads-win32/

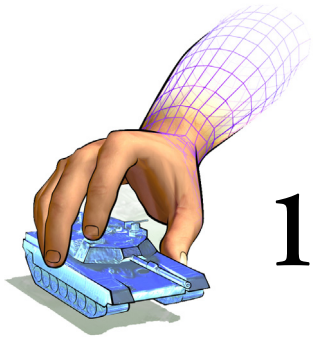
EHS, PCRE, and PME

The MÄK RTI links with the EHS, PCRE, and PME libraries. These libraries are distributed under the GNU Lesser General Public License (LGPL). MÄK has made modification to these libraries only to allow them to compile on the supported platforms. For more information about these libraries please see the following web sites:

- EHS: http://xaxxon.slackworks.com/ehs/
- PCRE: http://www.pcre.org/
- PME: http://xaxxon.slackworks.com/pme/index.html

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



1

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 1-2
Optimizing Simulation Engine Performance 1-2
Optimizing Front-End Performance 1-3
Monitoring the Back-end Frame Rate 1-4
Displaying Performance Statistics 1-4
Filtering Entities Using Interest Management 1-6
Enabling Interest Management 1-7
Configuring Interest Management 1-7
Miscellaneous Front-End Performance Configuration Options 1-9
Limiting Use of Spot Reports 1-9
Using Asynchronous I/O for DIS Exercises 1-9
Setting the Tick Rate 1-10
Specifying the Tick Rate for Components 1-10
Tuning the Network Interface 1-11
Tuning the State Repository Tick Rate 1-11
Balancing Visual Quality Against Network Performance 1-12
Enabling Configuration of Performance Options 1-13
Tuning the Target Frame Rate 1-13
Configuring VSync 1-13

1.1. 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.

1.1.1. Optimizing Simulation Engine Performance

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 (entities and tactical graphics) in the scenario and how active the entities 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.

VR-Forces supports the following methods for optimizing the performance of the simulation engine:

- Adjusting the refresh rates. ("Balancing Visual Quality Against Network Performance," on page 1-12)
- Turning off processor-intensive features. (For details, please see "Miscellaneous Front-End Performance Configuration Options," on page 1-9.)
- Adjusting the network processing time.
- Specifying the tick rate for an entity or component. ("Setting the Tick Rate," on page 1-10)
- Specifying a tick rate for the network interface. ("Tuning the Network Interface," on page 1-11)
- Specifying the VR-Forces sleep time. ("Tuning the Target Frame Rate," on page 1-13)
- Specifying the number of objects that you expect to simulate in the exercise.
- Optimizing the terrain.
- Using interest management. For details, please see "Filtering Entities Using Interest Management," on page 1-6.

1.1.2. 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.
- Number of objects in the simulation and how active they are.

You can display statistics to help you evaluate VR-Forces's performance. For details, please see "Displaying Performance Statistics," on page 1-4.

The following actions can affect performance:

- Increasing the size or density of a cloud layer.
- Displaying performance statistics.
- Use of ground clamping.
- Displaying track histories.
- Displaying height-above-terrain lines.

The following sections describe options for optimizing performance:

- "Configuring SpeedTree Trees," on page 12-32.
- "Configuring File Caching," on page 10-24.
- "Clearing the Model Instancing Cache," on page 11-6.
- "Setting the Clipping Planes," on page 9-10.
- "Enabling Texture Compression," on page 10-25.
- "Considerations and Limitations for Building Terrains," on page 10-3.
- "Filtering Entities Using Interest Management," on page 1-6.

1.2. Monitoring the Back-end Frame Rate

You can monitor frame rate statistics for each back-end in your session. The Frame Rate Monitor panel lists statistics for each back-end (Figure 1-1).

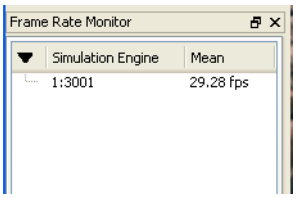


Figure 1-1. Frame Rate Monitor panel

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

1.3. Displaying Performance Statistics

VR-Forces can display two kinds of performance statistics — OSG statistics (Figure 1-2), and VR-Forces statistics (Figure 1-3).

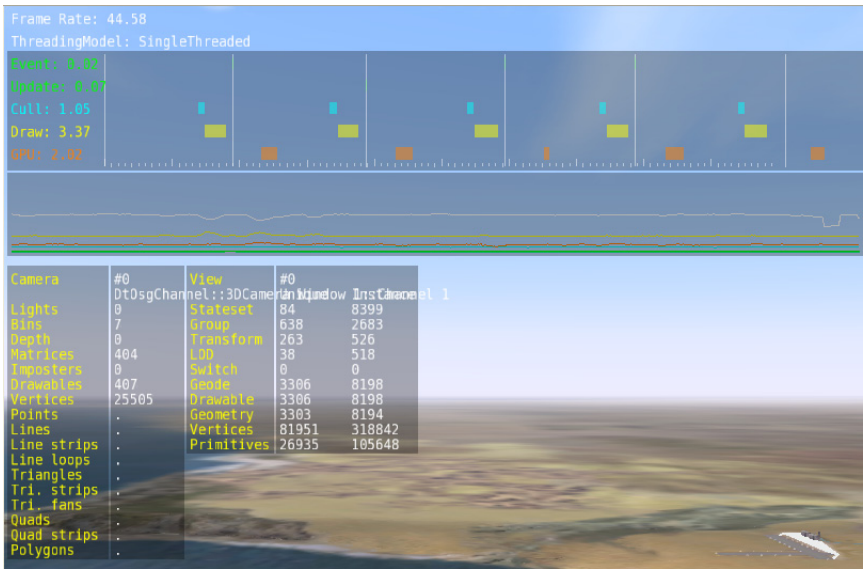


Figure 1-2. OSG performance statistics



Figure 1-3. VR-Forces performance statistics



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 1-4).

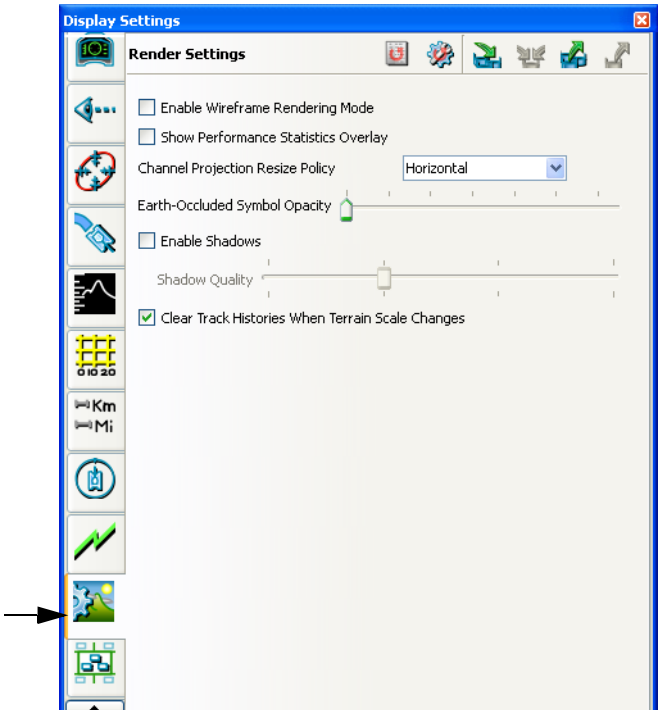


Figure 1-4. Render Settings page

3. Select the Show Performance Stats Overlay check box. A statistics display is added to the window (Figure 1-3).

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.

1.4. Filtering Entities Using Interest Management

VR-Forces can use interest management to improve front-end performance in simulations that have high entity counts. Interest management is an implementation of HLA data distribution management (DDM). When interest management is enabled, VR-Forces filters out all entities that are more than a specified distance from the observer.

1.4.1. 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 VR-Link20017-1
   --rprFomVersion 2.0017 -F VR-Link20017-1.fed
   --useGeographicDdm
```

1.4.2. Configuring Interest Management

Interest management works by filtering out all entities 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 entities 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 entities. 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 entity labels). Therefore, you may want to filter out entities 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 1-5).

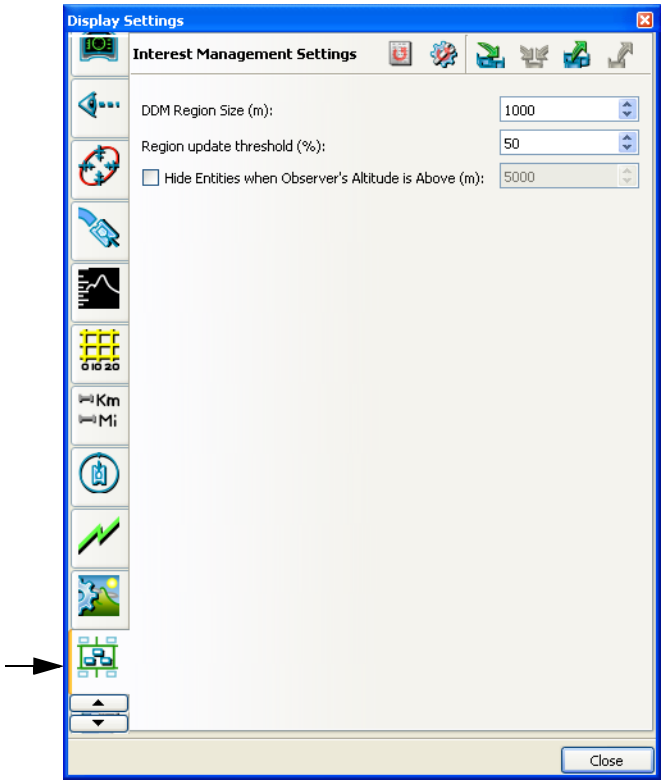


Figure 1-5. Interest Management Settings page

3. In the DDM Region Size box, type a value, in meters, for the region size.
4. In the Region Update Threshold box, type a value, as a percentage, for the update threshold.
5. Optionally, filter entities 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 entities.
6. Click OK.

1.5. Miscellaneous Front-End 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.

1.5.1. Limiting Use of Spot Reports

Use of spot reports can degrade performance. You may want to limit the number of entities for which they are generated or disable spot reports entirely.

1.5.2. 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 entities 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.

1.6. 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 entity components are ticked, you make more CPU time available to other entities. 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 an entity'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.

1.6.1. 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 entity and for individual entity components. The component-level settings, if any, override the entity-level settings. You can update these parameters in the OPD Editor.

1.6.2. 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.

1.6.3. 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.

1.7. 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 and the VR-Forces 3D view. 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 *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 1-6).

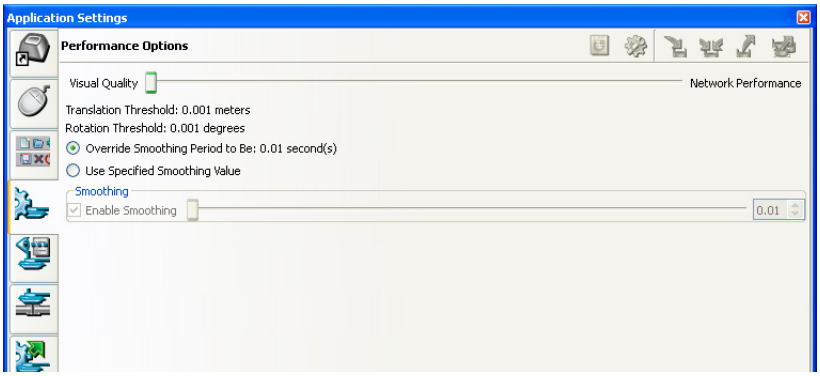


Figure 1-6. Performance Options page

3. Adjust the Visual Quality slider to your desired preference for visual quality versus network performance.
4. To override the default smoothing period and set it to a value calculated to work best with the performance setting, select Override Smoothing Period.
5. Click OK.

1.7.1. 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.

1.8. Tuning the Target Frame Rate

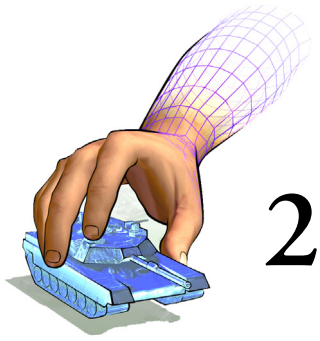
The `--targetFrameRate` command line option and *vrfSim.mtl* parameter allow you to specify the mean frame rate above which the VR-Forces back-end should begin to sleep in order to give up CPU time to other processes. This parameter is only applicable to variable frame mode; the fixed frame modes already 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.

1.9. 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.



2

VR-Forces Configuration Files

This chapter introduces the configuration files used by VR-Forces and describes the syntax of VR-Forces MTL files.

Introduction 2-2
The vrfSim.mtl Configuration File 2-2
Configuring DI-Guy Integration 2-8
Configuring Communication Models 2-9
- Simple Radio Communication Model 2-9
- External Communication Model 2-9
- Custom Communication Models 2-10

MÄK Technologies Lisp (MTL) Files 2-10
- Using Environment Variables in an MTL File 2-11
- Specifying Lists of Lists 2-11
- Using Conditional Statements 2-11

2.1. 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 2-10.

2.2. The vrfSim.mtl Configuration File

The file *../appData/settings/vrfSim/vrfSim.mtl* configures back-ends. Table 2-1 describes the parameters in *vrfSim.mtl*. The Option column lists the command-line argument equivalent for parameters that have them.

Table 2-1: *vrfSim.mtl* parameters (Sheet 1 of 6)

Parameter	Description	Option
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.	`-a | -p`

Table 2-1: *vrfSim.mtl* parameters (Sheet 2 of 6)

Parameter	Description	Option
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 will cause 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 Section 3.6, "Object Operations and Blocking Terrain Calls", in *VR-Forces Developers Guide*.	
batchScenarioFileName	Specifies a batch file to run.	`-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*.	
daemon	Runs vrfSim as a daemon process (Linux only.) vrfSim will not attempt to read characters from a console, and will fork a background process on startup. It will exit cleanly on receipt of the SIGTERM, SIGQUIT, SIGINT signals. Default: Off.	
		`--daemon`
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.	
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.	`-n`

Table 2-1: *vrfSim.mtl* parameters (Sheet 3 of 6)

Parameter	Description	Option
entVrfDataTimeThreshold	Specifies how often, in seconds, VR-Forces environmental and entity 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 OPD as the previous one (for example, in a rewind), the OPD is not reloaded.	
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 1516e.) Allowing a plug-in to load if there is a mismatch can cause VR-Forces to be unstable.	
logFrameRateStatistics	Enables (1) or disables (0) logging of frame rate statistics to a file.	
maxDrainInputTime	Sets the maximum time to wait in drainInput() before returning. For the DMSO and MÄK RTIs, 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.	
objectConsoleNotifyLevel	Specifies the notification level (0-4) for messages sent to the entity 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/*.	
publishSimulationTime-ForStealth	Specifies whether or not to publish the simulation time to the network so that MÄK Stealth 6.2 or earlier can set its simulation clock to work in conjunction with the simulation time.	
scenarioFileName	Specifies the scenario file to load.	`-L`
sendBackendLogToNetwork	Enables (1) or disables (0) sending simulation engine log messages to the front-end.	
sessionId	Specifies the session ID.	`-i`

Table 2-1: *vrfSim.mtl* parameters (Sheet 4 of 6)

Parameter	Description	Option
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.	`-s` | `-t`
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 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 1-13. Default: 22.	
terrainDatabase	Specifies the terrain database to load. This parameter is overridden by the scenario file if one is specified in scenarioFileName.	`-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 Section B.5, "Command Line Options for HLA Federations" in *VR-Forces Users Guide.*	
destroyFedExec	Specifies whether (1) or not (0) the *DtExerciseConn* destructor should try to destroy the federation execution. Default: 1.	
execName	The federation execution name.	`-x`
federateName	The federate name.	`-N`
fedFileName	Specifies the FED file name.	`-F`
fomMapperLib	A FOM Mapper library.	`-f`

Table 2-1: *vrfSim.mtl* parameters (Sheet 5 of 6)

Parameter	Description	Option
fomMapperInitData	An initialization string.	`--fomMapperInitData`
minDrainInputTime	The minimum time to stay in drainInput() before returning.	
rprFomVersion	The RPR FOM version. (0.5, 0.8, 1.0, 2.0)	`--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 Section B.6, "Command Line Options for DIS Exercises" in *VR-Forces Users Guide*.	
addMulticastAddr	The multicast address.	`-S`
appIdRange	Specifies the application ID range to use to identify entities 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.	`-A`
disPort	The port number.	`-P`
exerciseId	The exercise ID.	`-x`
maxDrainInputReads	The maximum number of reads to make in drain input before returning	
mcastTtl	Specifies the multicast time to live.	`--mcastTtl`

Table 2-1: *vrfSim.mtl* parameters (Sheet 6 of 6)

Parameter	Description	Option
recvBufferSize	Specifies the receive buffer size, as a positive value. A value of -1 means use the system default.	`--recvBufferSize`
sendBufferSize	Specifies the send buffer size, as a positive value. A value of -1 means use the system default.	`--sendBufferSize`
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.	`-I`

2.3. Configuring DI-Guy Integration

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 *VR-Forces Users Guide*.

You can specify the default character, appearance, and animation for a lifeform entity in the Entity Editor or OPD Editor. (For details, please see "Editing a Lifeform's DI-Guy Character, Appearance, and Animation," on page 4-19.) The DI-Guy mappings are stored in the entity's OPE 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 drop-down lists in the Entity 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 the DI-Guy SDK, VR-Forces will support these characters.

- *./appData/settings/vrfSim/character_digest.xml* contains information that VR-Forces needs to synchronize an entity'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.

2.4. Configuring Communication Models

Communication models are configured in *./data/simulationModelSets/<model-set>/commModelParams.mtl*. Communication models determine how radio messages travel from one entity to another. They determine which entities are connected to each other and how long it takes a message to get from one entity 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.

2.4.1. Simple Radio Communication Model

VR-Forces includes a default communication model called simple-radio-comm-model. This communication model models radio traffic in a very simple way. 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, **connection-mode**, which determines how the radio networks are set up. There are three connection-modes available:

- `aggregate` mode specifies that each aggregate has its own radio network. Messages sent from an entity can reach any entity that is in the same organization, at the same level (the entity's peers in the organization), and can also reach the entity's direct superior. All other entities are outside the network, and are unable to receive messages sent by the entity. This is the default mode.



If an entity is not in an aggregate, it is considered to be part of the force aggregate and its radio messages go to other entities that are part of the force aggregate.

- `force` mode specifies that there is a radio network for each force in VR-Forces. Messages sent from an entity can reach any other entity of the same force.
- `all` mode specifies that all entities in VR-Forces are part of one large radio network. In this case, any entity can receive messages sent by any other entity.

2.4.2. External Communication Model

VR-Forces includes an external communication model that can communicate with an external communication effects server over the network to determine how to deliver radio messages. For details, please see Chapter 8, *Using VR-Forces with an External Comms Effects Server*.

2.4.3. 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 Section 9.5.1, "Creating Custom Communication Models" in *VR-Forces Developers Guide*.

2.5. 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")
```

2.5.1. 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)))
```

2.5.2. 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))
```

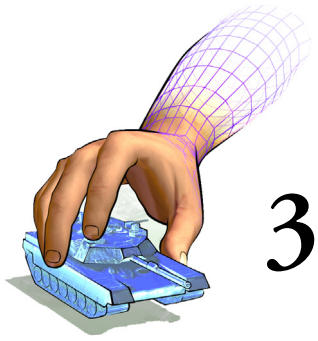
2.5.3. Using Conditional Statements

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)
   )
)
```



3

Scenario Files

This chapter describes the files that get created when you create a scenario.

Introduction 3-2
The Scenario File 3-3
Scenario Parameters 3-4
Specifying Pathnames for Scenario Files 3-7
Changing the Terrain Database for a Scenario 3-8
Editing the Plan Files 3-8
The Structure of an Entity Plan File 3-9
The Object Map File 3-11
Editing the Order of Battle File 3-12
Saving Default Entity Parameters to the Order of Battle File 3-12
The Scenario Extras File 3-12
Temporary Scenario Directories 3-13

3.1. 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 entities, control objects, and overlay objects.
- The object map file, named *scenario_name.omp*, maps objects to the back-ends on which they were created.
- The plan file, named *scenario_name.pln*, stores entity plans.
- The global plan file, names *scenario_name.gpl*, stores the scenario's global plans.
- The scenario extras file, named *scenario_name.xtr*, lists the forces that each force is hostile to. If you have installed the B-HAVE Module, it stores spawn pattern templates.
- The scenario-specific GUI settings file, named *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.

3.2. The Scenario File

When you create a scenario, you can set its parameters in the New Scenario dialog box (Figure 3-1). Thereafter, you cannot edit scenario parameters in the front-end except for the scenario description and the option to attach components to remote entities. However, you can edit them in a text editor.

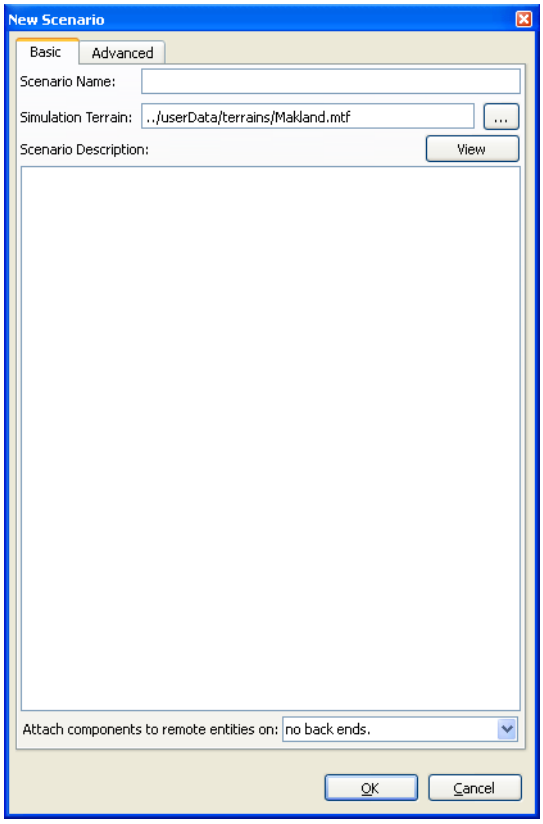


Figure 3-1. New Scenario dialog box

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.

3.2.1. Scenario Parameters

A typical scenario file looks like the following:

```
(Scenario
  (Component-Attachment  0)
  (Component-Attachment-Table  "")
  (Terrain-Database  "..\userData\terrains\Makland.mtd")
  (Gui-Terrain-Database  "..\userData\terrains\Makland.mtf")
  (Order-Of-Battle  "example.oob")
  (Plan  "example.pln")
  (Global-Plan  "example.gpl")
  (Overlay  "example.ovl")
  (SelectionGroups  "example.sgr")
  (Object-Map  "example.omp")
  (Scenario-extras  "apache_groundDB.xtr")
  (time-multiplier  1.000000)
  (auto-reorganize False)
  (random-number-seed  0)
  (frame-mode  "variable-frame")
  (frame-time  0.000000)
  (sim-time )
  (exercise-start-time )
  (additional-opds )
  (Simulation-Model-Set-Files "../data/simulationModelSets/default.sms")
  (scenario-name  "maklanddemo")
  (scenario-data
    (GuiScenarioSettings
      "..\userData\scenarios\makland\maklanddemo.gui_settings")
    (ScenarioDescription  "The Makland Demo is a moderately complex
      demonstration scenario that uses ground, lifeform, surface,
      fixed-wing, and rotary-wing entities. Entities have plans and
      friendly and opposing forces interact.")
  )
  (version  3.100000)
)
```



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

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 entities 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 entities 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.
- **Plan** specifies the plan file.
- **Global-Plan** specifies the global plan file.
- **Overlay** specifies the overlay file[1].
- **SelectionGroups** specifies the selection groups file[1].
- **Object-Map** specifies the object mapping file.
- **Scenario-Extras** specifies the force hostility relationships. If the B-HAVE Module is installed, spawn pattern templates are stored in this file. It may be used for other purposes in the future.
- **time-multiplier** specifies whether a scenario runs in real time or at a rate other than real time. A value greater than 1 runs the simulation faster than real time. A value between 0 and 1 runs the simulation slower 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.

1. Although the scenario file contains entries for an overlay file and a selection groups file, these files are not created or used in VR-Forces 4.0.4.

- **auto-reorganize** lets you specify the default behavior for aggregate reorganization. Set it to True or False. For information about reorganization, please see Section 3.6.1, "How Aggregates Are Organized" in *VR-Forces Users Guide*.
- **random-number-seed** sets the default random number seed. It provides a starting point for the generation of random numbers needed by VR-Forces. Since computer-generated random numbers are not truly random, if you run a scenario repeatedly with the same random number seed (and you run in fixed-frame-mode), you will get the same results. (For more information about repeatability, please see Section 3.13.3, "Exercise Clock Modes" in *VR-Forces Users Guide*.) 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 Section 3.13.3, "Exercise Clock Modes" in *VR-Forces Users Guide*.
- **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:
  - **GuiScenarioSettings** specifies the gui_settings file. It records scenario-specific GUI settings.
  - **ScenarioDescription** is the text of the scenario description.
- **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.

3.2.2. 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 *./bin* directory. To change pathnames, edit the scenario file in a text editor. VR-Forces will preserve 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 the *.opd* file 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 (*vrforces4.0.4/bin/vrfSim*).
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.

3.2.3. 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.

3.3. Editing the Plan Files

VR-Forces saves two plan files for each scenario, an entity plan file (*.pln*), which stores the plans for all individual entities, and the global plans file (*.gpl*), which stores global plans.

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. However, there may be occasions when it makes sense to edit a plan file by hand or programmatically. For example, if you change the name of an entity, updating each entity's plan in the Plan window to change all plan statements that refer to that entity could be tedious. It would be much quicker to open the plan file in a text editor and replace appropriate instances of the old entity name with the new entity name.

Before you edit a plan file, make a backup copy that you can revert to if necessary.

3.3.1. The Structure of an Entity Plan File

Figure 3-2 illustrates a simple entity plan that specifies one task for each of two entities (a global plan has a similar syntax):

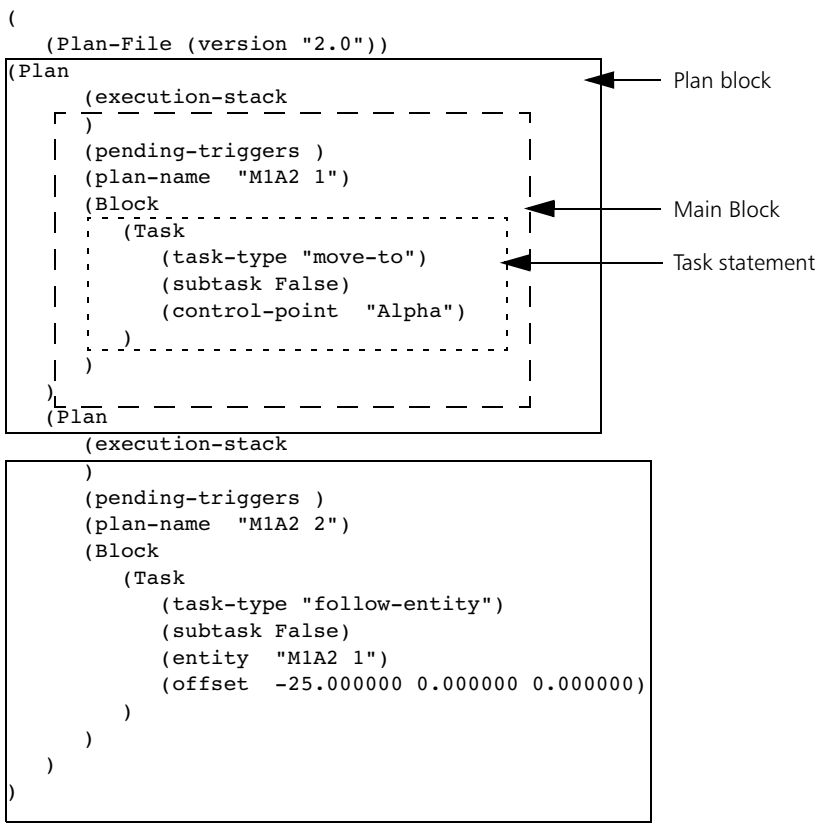


Figure 3-2. Simple plan

Plans are specified for entities by entity name. The plan is specified within a Plan block. The content of the plan is specified in the Main block. Tasks and set statements are specified in the Main block and in conditional statements. Conditional statements can be specified in the main block and can be nested in other conditional statements.

The basic syntax for a set statement in a plan is:

```
(Block
  (Set
    (set-data-request-type "set-heading")
    (heading  0.872665)
  )
)
```

Figure 3-3 illustrates the basic syntax for a conditional expression. It includes a task statement in the main block to show relative indentation.

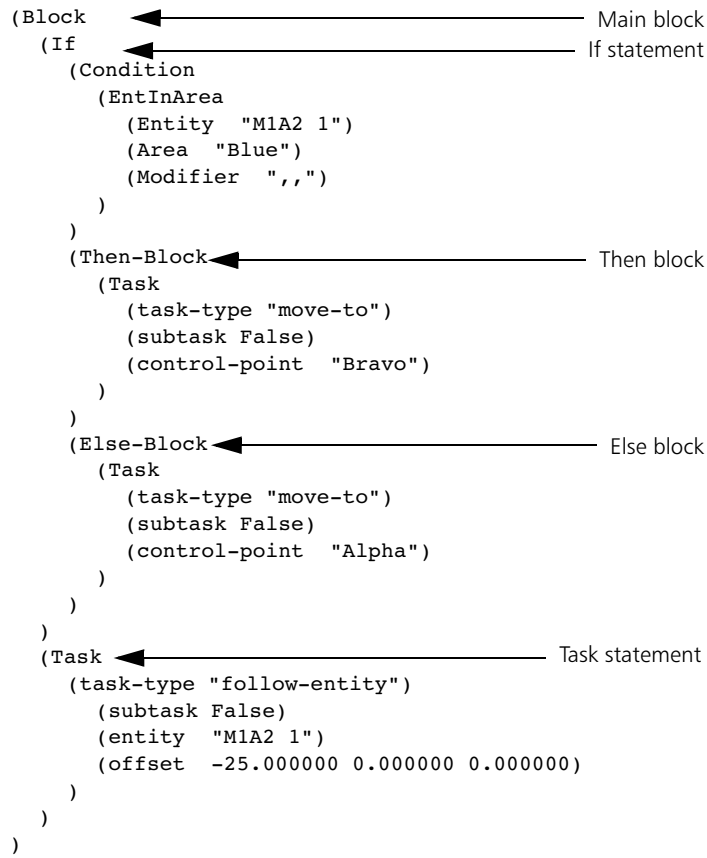


Figure 3-3. Conditional expression with subblocks

To edit a plan file by hand, add Task, Set, If, While, and When sections as needed to the Block section for each entity. For tasks or set statements, you will always have a **task-type** or **set-data-request-type** parameter that needs to contain the type of task or set data request. The presence of additional parameters depends on the task or set data request that you created. You will need to know the names of the parameters and what type of value to specify.

Make sure that you match parentheses properly.

3.4. 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 entities or 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 Section 1.2.3, "Load Balancing a Scenario" in *VR-Forces Scenario Management Guide*.

3.5. Editing the Order of Battle File

The order of battle is a description of the local objects in a simulation. When you create a scenario, you populate it with entities. When you save or checkpoint a scenario, VR-Forces saves a record of each entity, including its state information and geographic location, to the order of battle file. The next time you load this scenario, VR-Forces creates the entities 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 entity value, you might need to edit the order of battle file.

The order of battle file also describes the tactical graphics that have been defined for the scenario. Tactical graphics identify locations and areas in the terrain. You can place a basic set of objects on the terrain. For more information, please see "Tactical Graphics," on page 3-24, in *VR-Forces Users Guide* and Chapter 5, *Overlays and Tactical Graphics*, in *VR-Forces Scenario Management Guide*.

3.5.1. Saving Default Entity Parameters to the Order of Battle File

When an entity in a scenario is saved to the order of battle (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 entity's object parameter database 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 entities and objects in the scenario.

This lets you experiment with changes to entity parameters and view the effects of the changes to existing scenarios, without modifying the order of battle files.

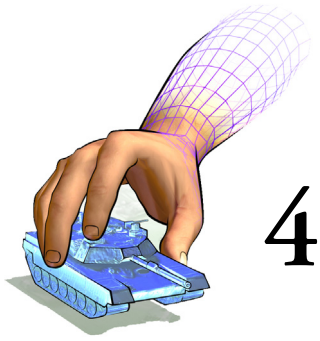
3.6. The Scenario Extras File

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*).

If you have the B-HAVE Module installed, it stores spawn patterns in the scenario extras file.

3.7. 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.



4

Editing Entity Simulation Models

The Entity Editor is a stand-alone tool that lets you configure entities, edit the Create menu, and create new simulation model sets.

Introduction to the Entity Editor 4-3
Simulation Model Sets 4-4
Starting the Entity Editor 4-5
Creating a New Simulation Model Set 4-6
Creating a Completely New Simulation Model Set 4-7
Specifying the Default Simulation Model Set for Scenarios 4-7
Opening a Simulation Model Set 4-8
Editing an Entity Simulation Model 4-9
Selecting an Entity to Edit 4-10
Editing an Entity's Category 4-11
Editing an Entity's Force 4-13
Changing an Entity's Entity Enumeration 4-14
Changing an Entity's Name 4-15
Specifying Whether an Entity Can Be Created 4-15
Editing an Entity's Bounding Volume (Size) 4-16
Editing Behavioral Parameters 4-18
Editing a Lifeform's DI-Guy Character, Appearance, and Animation 4-19
Undoing (Reverting) Changes to an Entity 4-21
Editing an Entity's Visual Model 4-21
Adding a New Visual Model or Image 4-24
Editing Aggregates 4-24
Adding Subordinates 4-26
Removing Subordinates 4-27
Replacing Subordinates 4-27

Changing the Order of Subordinates 4-27
Editing a Subordinate Aggregate 4-28
Adding Formations 4-29
Removing Formations 4-29
Renaming Formations 4-29
Specifying the Default Formation 4-30
Expanding and Collapsing the Display of Formations 4-30
Displaying Bounding Boxes 4-31
Copying Formations 4-32
Laying Out a Formation Automatically 4-33
Changing a Formation's Layout by Hand 4-33
Editing an Entity's Systems 4-35
Adding a System 4-37
Selecting a Damage Model 4-37
Selecting a Movement Model 4-37
Editing a System's Properties 4-38
Editing a System in the OPD Editor 4-38
Removing a System 4-38
Creating a New Entity 4-39
Creating a New Entity from an Existing Entity 4-39
Deleting an Entity Model from a Model Set 4-40
Navigating Among Edited Entities 4-40
Editing the Create Menu 4-40
Generating the Create Menu Automatically 4-41
Editing the Create Menu Manually 4-41
Undoing (Reverting) Changes to the Create Menu 4-44
Editing the Category List 4-44
Adding a New Category 4-44
Removing a Category 4-45
Renaming a Category 4-45
Undoing Category Changes 4-45
Managing the Forces List 4-46
Adding a Force 4-46
Editing a Force 4-47
Removing a Force 4-48

4.1. Introduction to the Entity Editor

Many VR-Forces users want to edit some or all of the following entity simulation characteristics:

- Weapons systems
- Basic parameters
- Sensors
- Force.

Users also want to add new entities and rearrange how entities are organized on the Create menu. The Entity Editor allows you to edit most of the aspects of an entity that a typical VR-Forces user will want to change. For example, you can change the force associated with an entity type. You can easily add and remove sensors, weapons systems, and other major component systems, and can edit their properties.



The Entity Editor primarily lets you edit entity 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 Entity Editor also lets you configure the 3D models and 2D icons used to display entities. For more information, please see Chapter 11, *Mapping Models and Effects* and Chapter 12, *Model and Element Definitions*.

The Entity 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 5, *Advanced Entity Model Editing*, or edit specific configuration files, which are described throughout this manual.

If you want to add new entities or customize the entities supplied with VR-Forces, start with the Entity Editor. Use the OPD Editor or a text editor only if the Entity Editor does not let you make the changes you need to make.

Do not try to edit an entity model in the Entity Editor and the OPD Editor at the same time, except as described in "Editing a System in the OPD Editor," on page 4-38. Use one tool, save the changes, and close the tool before opening the other.

4.1.1. Simulation Model Sets

A simulation model set (SMS) defines all of the entity and 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 entities and other simulation objects. These files include the object parameter database, damage files, and formation files. Every scenario must reference one or more simulation model sets.

If you use the Entity Editor to edit entity 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 Entity Editor.

When you use the Entity Editor, you are editing a simulation model set. The default SMS provided with VR-Forces is *default.sms*. We recommend that you keep the default SMS as a reference model set and not edit it. If you want to edit entity models, create a new SMS based on *default.sms* and use it as the default SMS for your scenarios. For details about creating a new SMS and changing the default SMS, please see "Creating a New Simulation Model Set," on page 4-6.

4.1.2. Starting the Entity Editor

➤ To start the Entity Editor in Windows, choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.0.4** → **Tools** → **Entity Editor**. The Entity Editor opens (Figure 4-1).

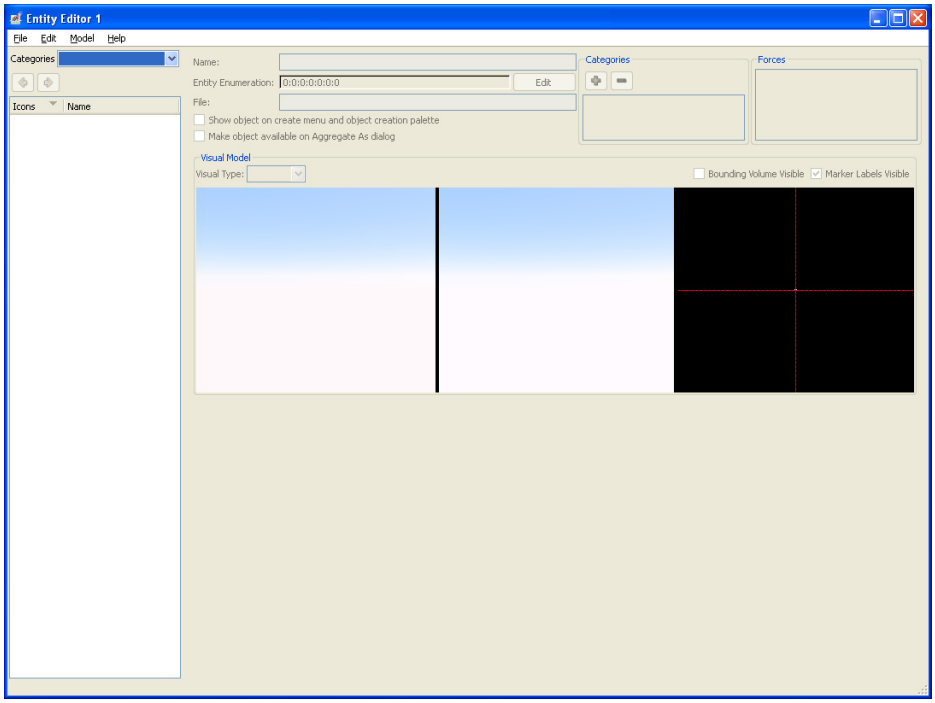


Figure 4-1. Entity Editor

To start the Entity Editor from the command-line in Windows or Linux:

1. Change directory to *./bin*.
2. Run:

```
entityEditor
```

4.2. Creating a New Simulation Model Set

You can create new SMSs by copying and editing an existing model set or you can create a completely new model set and create the entities that you want to include in it.

To create a new simulation model set from an existing model set:

1. Choose **File** → **New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 4-2).

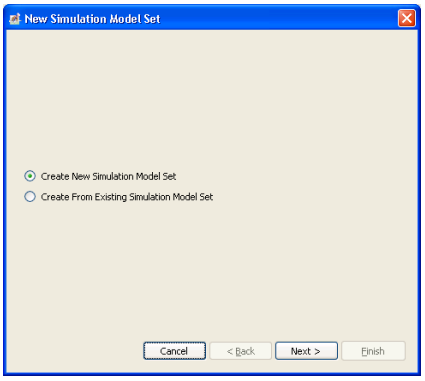


Figure 4-2. New Simulation Model Set dialog box

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 New Model Set Name dialog box opens.
5. In the File name box, type a name for the new model set.
6. Click Finish. The Entity Editor loads information from the source model set.
7. Make any changes you want to the model set.
8. Choose **File** → **Save Simulation Model Set**.

4.2.1. 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 4-2).
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 Entity Editor loads systems information from the default model set.
6. Create the entities that you want to have in the model set, as described in "Creating a New Entity," on page 4-39.
7. Choose **File** → **Save Simulation Model Set**.

4.2.2. Specifying the Default Simulation Model Set for Scenarios

When you create a new scenario, the New Scenario dialog box lists *default.sms* as the SMS to use. You can change the default SMS by editing the **simulationModelSet** parameter in *./appData/settings/vrfSim/vrfSim.mtl*.

4.3. Opening a Simulation Model Set

To edit entity models in the Entity 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 entities is displayed in the Entity Editor.

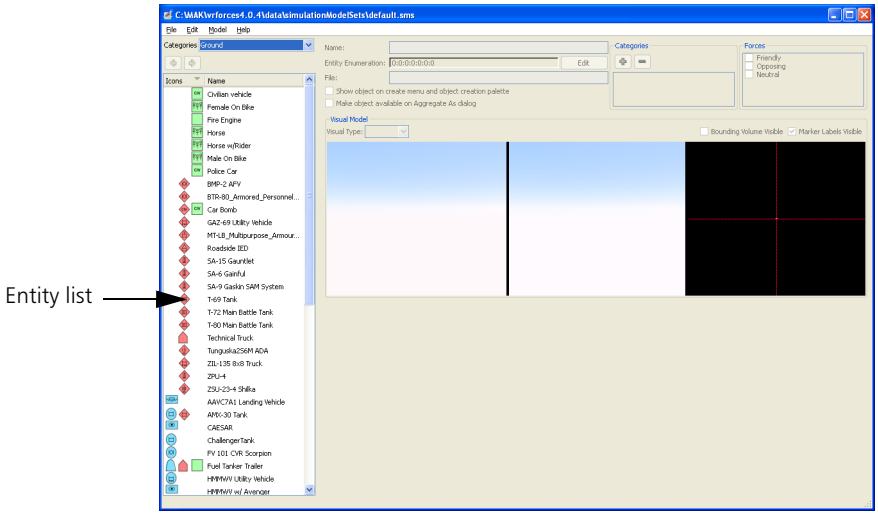


Figure 4-3. Entity Editor with model set loaded

4.4. Editing an Entity Simulation Model

This section explains how to edit an existing entity model. You can:

- Change the force assignments.
- Add and remove systems.
- Edit the properties of systems.
- Edit some entity parameters.
- Change the entity's category.
- Change the entity's name.
- Specify whether or not the entity can be created from the GUI
- Specify an entity's 3D model and 2D icon.



The 3D models and 2D icons used to display entities are also configured as part of model definitions and entity definitions. For information about changing models and icons, please see Chapter 11, *Mapping Models and Effects*.

When you edit an entity model in the Entity Editor, you are essentially editing the OPE file that it uses. In the default SMS, some OPE files are used by more than one entity. If you edit a particular entity in the Entity Editor, the tool creates a unique OPE file for that entity, so the changes do not affect other entities that might be using the original OPE file.

4.4.1. Selecting an Entity to Edit

Most of the Entity Editor procedures assume that you have selected an entity in the entity list to edit.

To select an entity to edit:

1. In the entity list, select the category for the type of entity you want to edit. The Entity Editor loads the information it needs for the category and the list redisplays.
2. Select the entity in the list.



When you select a human or animal entity in the entity list, the Entity Editor marks the entity as changed even though you have not made any changes. This is due to some anomalies in how DI-Guy characters are mapped to entity types. The Entity Editor automatically determines the correct model to display in the Visual Models panels and marks the entity as changed. This is expected behavior.

Even if you do not make any changes to the entity, save the SMS to ensure that correct visual models are used if you use the entity in a scenario. To save the SMS, choose **File** → **Save Simulation Model Set**.

4.4.2. Editing an Entity's Category

Entities are organized by categories on the Create menu and the Entity Palette. You can change the category that an entity is listed in. You can also list it in more than one category. For details about adding new categories to the list, please see "Adding a New Category," on page 4-44.

To change an entity's category:

1. In the Categories group box (Figure 4-4), click the category that you want to change. The field becomes editable.

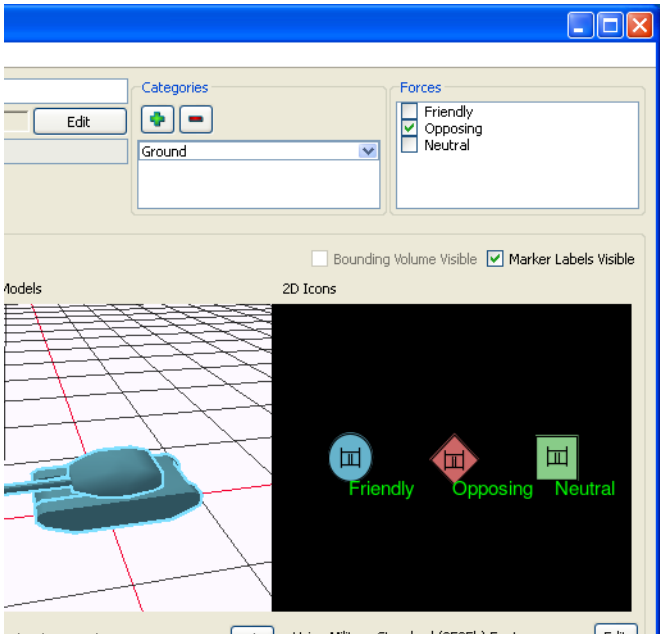


Figure 4-4. Categories group box

2. Select a different category from the drop-down list.

To add a category:

1. In the Categories group box, click the Add icon (). A new entry is added to the Categories list.

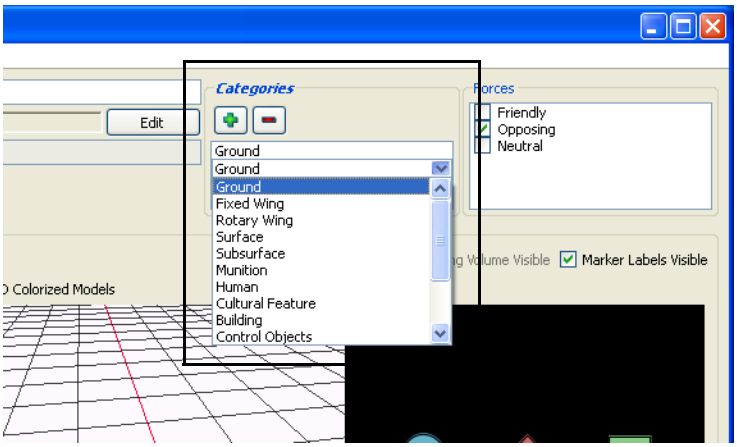


Figure 4-5. Adding a category

2. Select a category from the drop-down list.

4.4.3. Editing an Entity's Force

Each entity can be created as a member of one or more forces. You can specify which force an object can be created for. When you add an entity to a force, an icon for that force is assigned. If you remove an entity from a force, the icon is removed. You can quickly see which forces an entity is associated with by scanning the entity list in the Entity Editor (Figure 4-6). For details about adding new forces, please see "Adding a Force," on page 4-46.

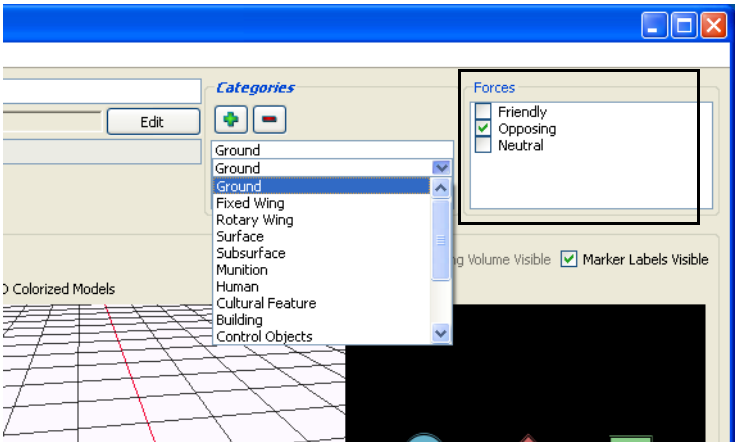


Figure 4-6. Entity (Leclerc Tank) assigned to multiple forces

➤ To make or remove force assignments for an entity, in the Forces list, select or clear the check boxes for the forces that you want the entity to be assigned to.

4.4.4. Changing an Entity's Entity Enumeration

The Entity Editor lets you change an entity's enumeration. 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.
- If you create a new model from an existing model, the Entity 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 Entity Editor generates.
- You might want to change an existing entity's enumeration.



When you change an entity's enumeration and save the change, the Entity Editor creates a new OPE file with the new enumeration in the file name. It does not delete or change the OPE file with the entity's previous enumeration.

To change an entity's entity enumeration:

1. Select the entity in the entity list.
2. Click Edit next to the Entity Enumeration text box (Figure 4-7).

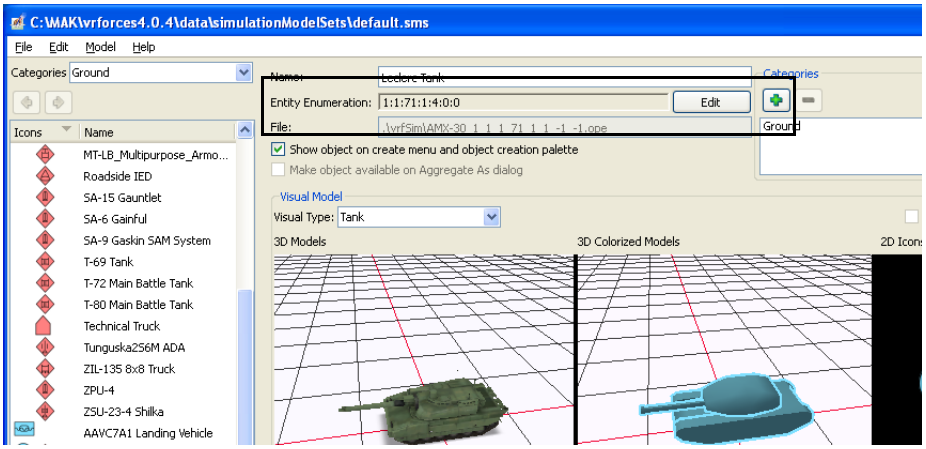


Figure 4-7. Entity enumeration

The Change Enumeration dialog box opens (Figure 4-8).

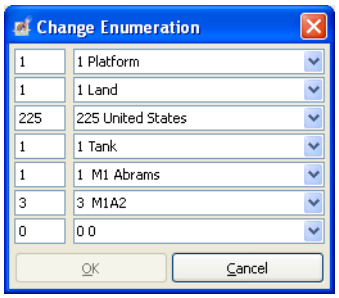


Figure 4-8. Change Enumeration dialog box

3. Edit the enumeration by typing new enumerations in the left column or selecting new entries in the right column.
4. Click OK. The enumeration changes and the OPE filename changes.

4.4.5. Changing an Entity's Name

To change an entity's name:

1. Select the entity in the entity list.
2. Type a new name in the Name text box.

4.4.6. Specifying Whether an Entity Can Be Created

You can control whether or not an entity is listed in the Create menu and Entity Palette. It may not make sense to be able to create some types of entities, such as missiles, which are normally added to a scenario and removed dynamically.

➤ To enable or disable creation of an entity from the Create menu and Entity Palette, select or clear the Show Object on Create Menu and Object Creation Palette check box (Figure 4-9).

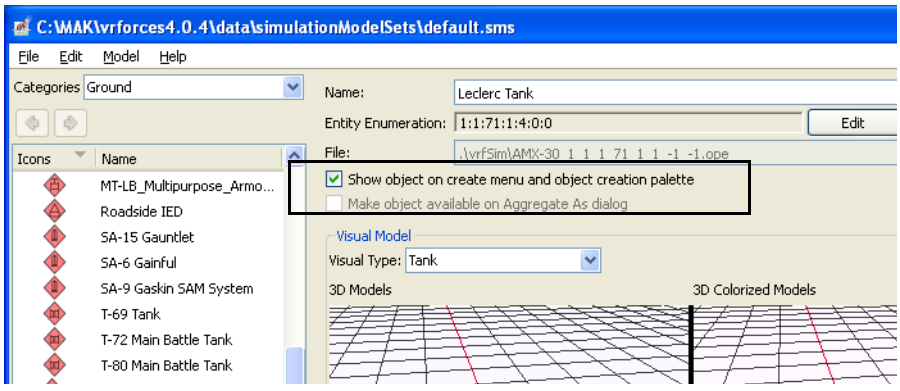


Figure 4-9. Show Object on Create Menu and Object Creation Palette check box

4.4.7. Editing an Entity's Bounding Volume (Size)

VR-Forces uses an entity's bounding volume to do intersection tests. The bounding volume is specified by the values in the Size group box. You can display an entity's bounding volume in teh Visual Model group box. If you change size values, the bounding volume changes accordingly.



The bounding volume is not the same thing as the selection box that is displayed around an entity when you select it. The selection box is just part of the front-end display. The bounding volume is used by the simulation engine.

To change an entity's bounding volume:

1. Select the entity you want to edit in the entity list.
2. Change any of the values in the Size group box (Figure 4-10).

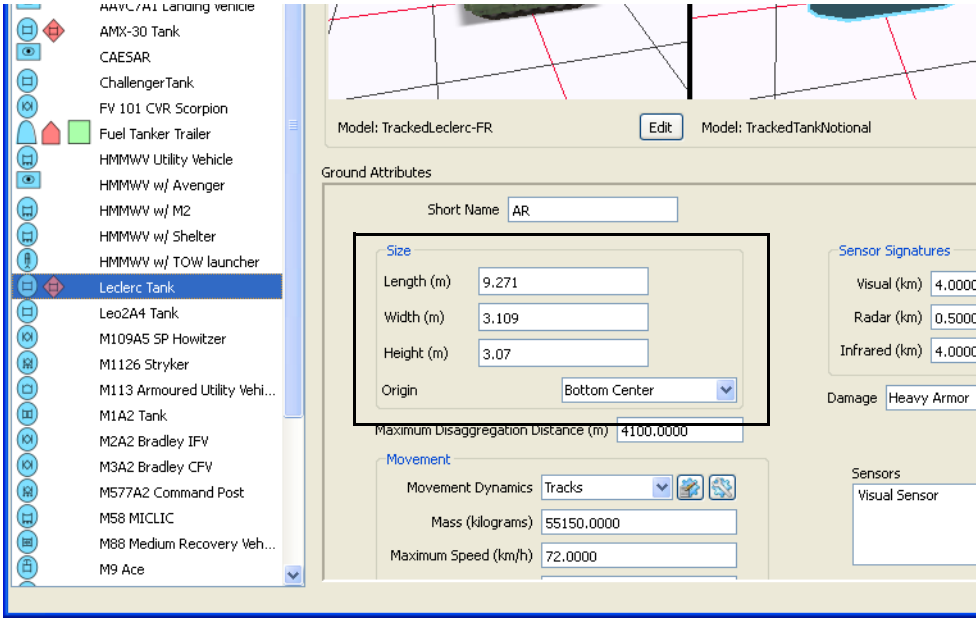


Figure 4-10. Size group box

Displaying an Entity's Bounding Volume



You must run VR-Forces at least once before you can display bounding volumes in the Entity Editor.

To display an entity's bounding volume:

1. Select the entity in the entity list.
2. In the Visual Model group box, select the Bounding Volume Visible check box. The bounding volume is displayed in the 3D Models panel (Figure 4-11).

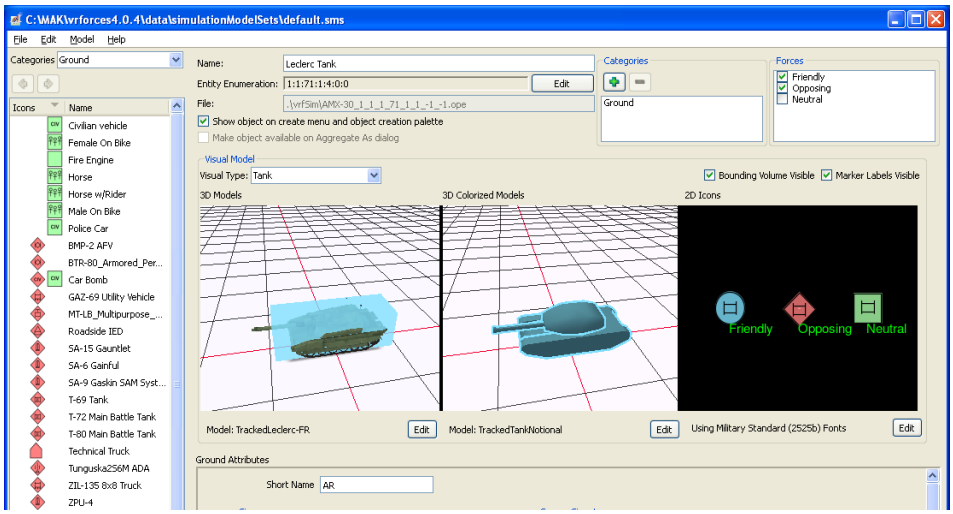


Figure 4-11. Bounding volume

4.4.8. Editing Behavioral Parameters

The Entity Editor lets you edit some of the entity parameters that affect behavior. For detailed information about entity parameters, please see the parameter descriptions in the OPD Editor online help.



The Sensor Signature parameters specify the distance at which other entities can sense the entity you are editing. By contrast, the sensor systems on the entity control its ability to sense other entities. For more information, please see "Editing an Entity's Systems," on page 4-35.

➤ To edit entity parameters, edit any of the parameters in the Attributes panel (Figure 4-12). (The panel reconfigures itself as you select different types of entities.)

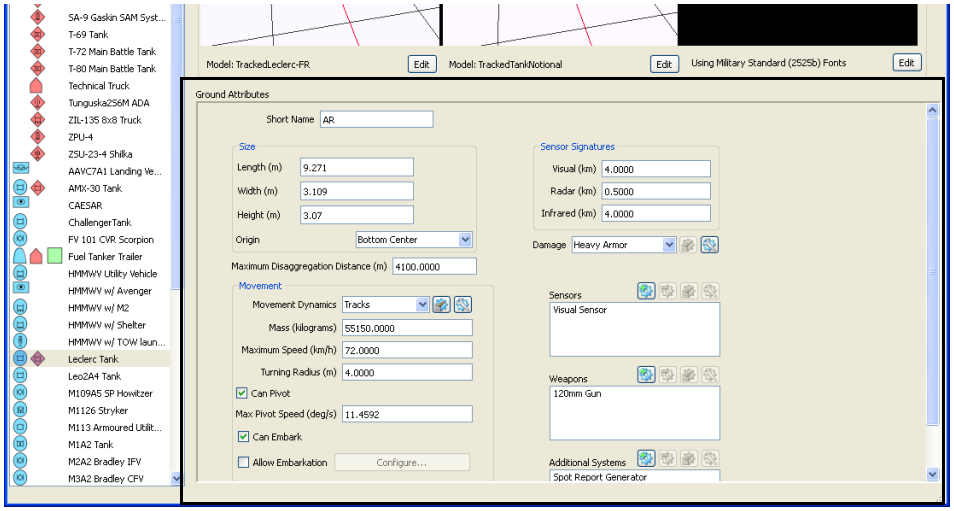


Figure 4-12. Attributes panel

4.4.9. 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 drop-down list, select the Human or Animal category.
2. Select an entity in the entity list.
3. In the Visual Model group box, click the Edit button for the 3D Models panel. The Modify Visual dialog box opens (Figure 4-13). The Use DIGuy option should be selected. If it is not selected, select it.

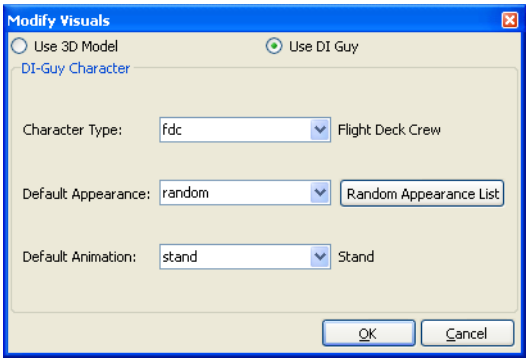


Figure 4-13. DI-Guy Character group box

4. Select the DI-Guy character you want to use from the Character Type drop-down list.
5. Select the default appearance from the Default Appearance drop-down list. One of the default options is to use a random appearance.
6. 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 4-20.
7. Select the default animation from the Default Animation drop-down 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 drop-down list, select the Human category (or Animal category).
2. Select an entity in the entity list.
3. In the Visual Model group box, click the Edit button for the 3D Models panel. The Modify Visual dialog box opens (Figure 4-13). The Use DI-Guy option should be selected. If it is not selected, select it.

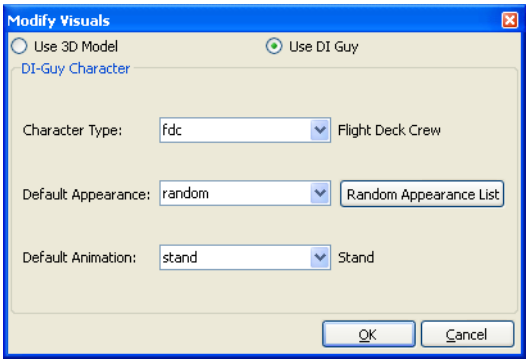


Figure 4-14. DI-Guy Character group box

4. Click the Random Appearance List button. The Random Appearance List dialog box opens (Figure 4-15). The appearances available for the character type are listed.

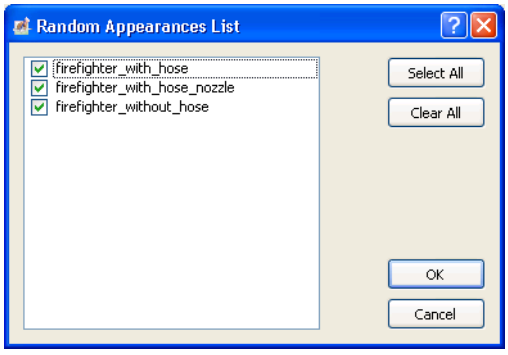


Figure 4-15. Random Appearances List dialog box

5. Select the appearances that you want to include in the random appearance list.

6. Clear any appearances that you do not want to include in the random appearance list.
7. Click OK.
8. Click OK to close the Modify Visuals dialog box.

4.4.10. Undoing (Reverting) Changes to an Entity

If you have made changes to an entity and have not yet saved the simulation model set, you can revert to the original entity configuration. You can revert an entity at any time prior to saving the SMS, regardless of how many other entities you may have changed since you edited the entity you want to revert.

To undo changes to an entity:

1. Select the entity in the entity list.
2. Choose **Model** → **Revert Model**.

4.5. Editing an Entity'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 12, *Model and Element Definitions*.) However, the Entity Editor lets you quickly and easily change the 3D model or 2D icon used to represent an entity. When you change a model in the Entity Editor, the entity's model definition also gets changed.

To change the visual model for an entity:

1. In the entities list, select the entity whose model you want to change. Its 3D models and 2D icon are displayed in the panels in the Visual Model group box.



The panels are live windows. If you move the cursor over a panel, you can manipulate the view using the same keyboard controls that are available in the front-end.

2. Optionally, select a visual type in the Visual Type drop-down list. This choice changes the 3D model for any entity that is using a generic model. If an entity has a specific model, changing the visual type does not affect it.
3. To change a 3D model:
   a. Click the Edit button for the 3D model you want to change (3D or 3D Colorized). The Modify Visuals dialog box opens.
   b. 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 4-16). (You can add new models. For details, please see "Adding a New Visual Model or Image," on page 4-24.)

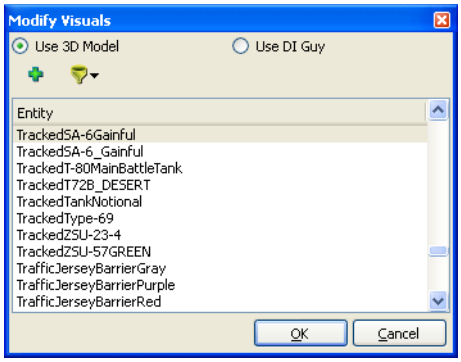


Figure 4-16. Modify Visuals dialog box (3D)

c. Select the entity model that you want to use. The new model is displayed in the panel.

d. Click OK to close the Modify Visuals dialog box.

4. To change a 2D icon using the MIL-STD font:

   a. Click the Edit button for the 2D Icons panel. The Modify Visuals dialog box opens (Figure 4-17).

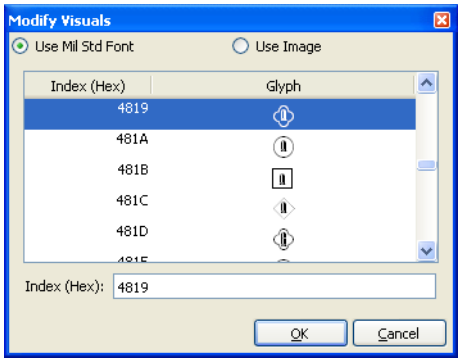


Figure 4-17. Modify Visuals dialog box (2D)

   b. Select the Use MilSTD Font option. The dialog box lists index values and associated glyphs.

   c. Select the icon glyph that you want to use. The new icon is displayed in the panel

   d. Click OK to close the Modify Visuals dialog box.

5. To change a 2D icon using an image:

   a. Click the Edit button for the 2D Icons panel. The Modify Visuals dialog box opens (Figure 4-17).

   b. Select the Use Image option. The dialog box lists the forces that are configured for this simulation model set (Figure 4-18).

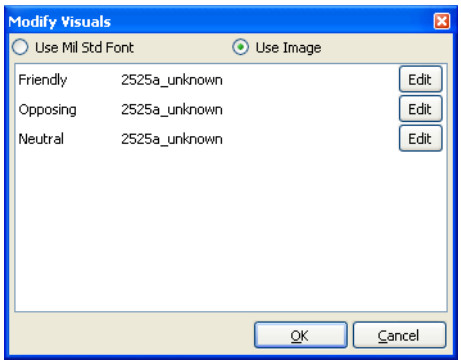


Figure 4-18. Modify Visuals dialog box (2D image)

c. Click the Edit button for the force whose icon you want to change. The Select Model Definition dialog box opens . 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 4-24.)

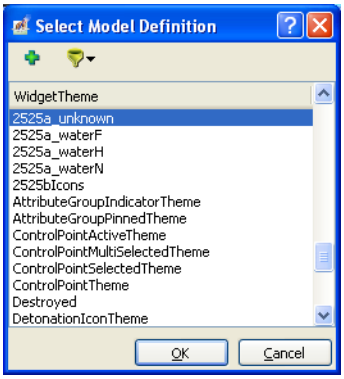


Figure 4-19. Select Model Definition dialog box

d. Select the image that you want to use. The new image is displayed in the panel.

e. Click OK to close the Modify Visuals dialog box.

4.5.1. Adding a New Visual Model or Image

When you change a model or image in the Entity 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. For 3D models, click the Edit button for the model type you want to edit. The Modify Visuals dialog box opens (Figure 4-16). For 2D icons, do the following:
   a. Click the Edit button for the 2D Icons panel. The Modify Visuals dialog box opens (Figure 4-17).
   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 4-18).
2. Click the Add icon (). The Model Definition dialog box opens. This is a standard file choose dialog box.
3. Select the file for the model that you want to add to the list. It must be a model or image type that VR-Forces supports.
4. Click OK.

4.6. Editing Aggregates

The procedures for creating aggregate entities and editing their generic parameters are the same as for individual entities (described elsewhere in this chapter). This section describes how to edit aggregate composition and formations.



The Convoy aggregate provided with VR-Forces is an exception to the discussion of aggregates in this section. It does not have any formations because there is no preconceived membership in a convoy. It does not have an Auto Aggregation Controller, which allows an aggregate to aggregate and disaggregate. Although you can assign this controller, it is not recommended, because the resulting behavior would be undefined.

Each aggregate entity provided with VR-Forces has a pre-defined set of subordinates and the following defined formations:

- Column
- Line
- Vee
- Wedge.

You can easily edit subordinate lists and formations in the Entity Editor. You can add, remove, and replace subordinates and you can change their order in the subordinate list. (The order in the subordinate list determines which aggregate is considered the leader. It also affects aggregate reorganization.)

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 aggregates.
- Zoom in and out on the display.
- Display subordinate bounding boxes.

Figure 4-20 illustrates the Entity Editor with an aggregate selected. Area 1 is the Composition Editor. Area 2 is the Formation Editor.

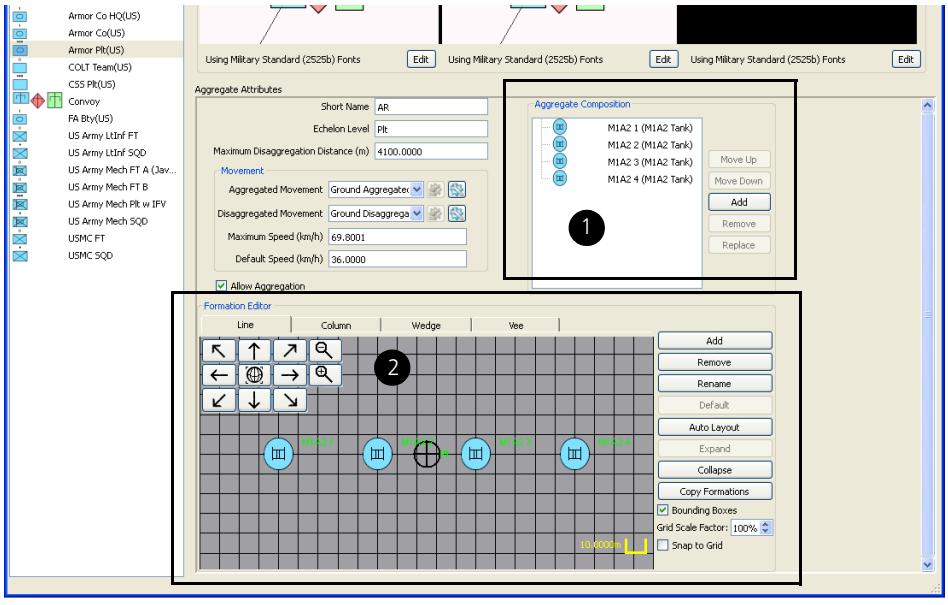


Figure 4-20. Aggregate page in Entity Editor

When you select an aggregate or individual entity in the Composition Editor or the Formation Editor, it is highlighted in both editors.

4.6.1. Adding Subordinates

When you create a new aggregate without copying an existing entity, you must specify its subordinates. You can also add subordinates to existing aggregates.

To add a subordinate to an aggregate:

1. In the Aggregate Composition group box (#2 in Figure 4-20), click the Add button. The Add to Hierarchy dialog box opens (Figure 4-21).

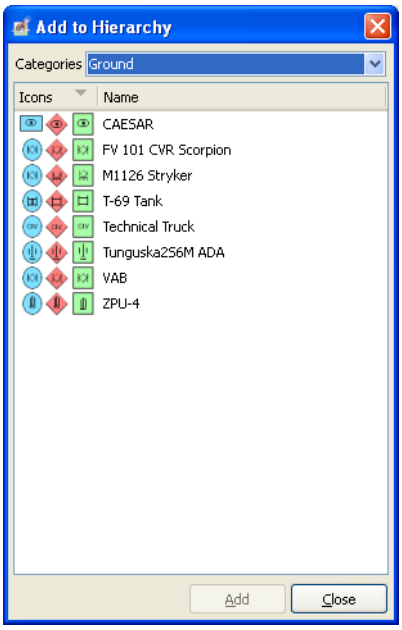


Figure 4-21. Add to Hierarchy dialog box

2. In the Categories drop-down list, select the category of the entity that you want to add as a subordinate.
3. In the list of entities, select the entity that you want to add.
4. Click Add. The entity is added to the list in the Aggregate Composition window. It is also added to each formation using a default layout algorithm.
5. Continue to add as many subordinates as desired.
6. Click Close.

4.6.2. Removing Subordinates

To remove a subordinate from an aggregate:

1. In the Aggregate Composition window, select the subordinate that you want to remove.
2. In the Aggregate Composition group box, click the Remove button. The subordinate is removed from the aggregate.

4.6.3. 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 Aggregate Composition window, select the subordinate that you want to replace.
2. In the Aggregate Composition group box, click the Replace button. The Add to Hierarchy dialog box opens (similar to Figure 4-21).
3. In the Categories drop-down list, select the category of the entity that you want to add as a replacement subordinate.
4. In the list of entities, select the entity that you want to add as a replacement.
5. Click OK. The entity replaces the selected entity in the Aggregate Composition window. It is also added to each formation in the location of the replaced entity.

4.6.4. Changing the Order of Subordinates

The order of subordinates in the Aggregate Composition window determines which entity is the leader, the order in which aggregates are placed into formation, and the succession when aggregates are reorganized.

To change a subordinate's position in the aggregate composition:

1. In the Aggregate Composition window, select the subordinate that you want to move.
2. In the Aggregate Composition group box, click Move Up or Move Down.

4.6.5. Editing a Subordinate Aggregate

Aggregates can contain individual entities and other aggregates as subordinates. When you are editing an aggregate, 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 entities.

! If you edit a subordinate aggregate, the changes affect all uses of that subordinate, not just the one in this particular aggregate.

To edit a subordinate:

1. In the Aggregate 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 Entity Editor redisplays to show the selected entity.

Navigating the Subordinate Hierarchy

If you jump to a subordinate in the Entity Editor, you can quickly back up to its superior aggregate by clicking the back arrow above the entity list (Figure 4-22).

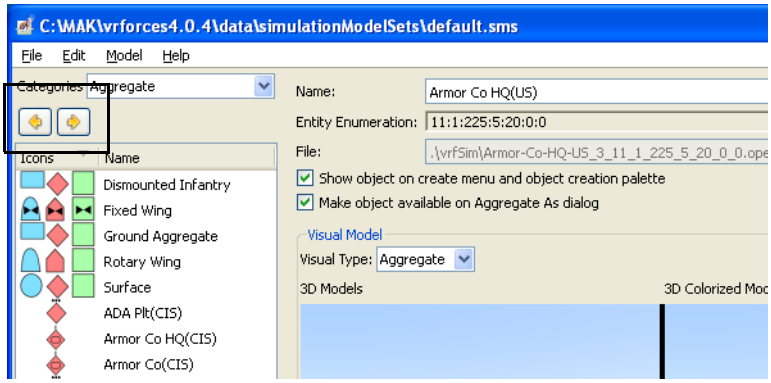


Figure 4-22. Forward and Back arrows

4.6.6. Adding Formations

To add a formation:

1. In the Formation Editor group box, click Add. The Add Formation dialog box opens (Figure 4-23).

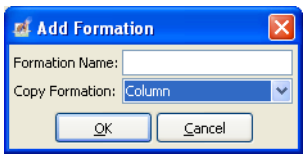


Figure 4-23. Add Formation dialog box

2. In the Formation Name box, type a name for the formation.
3. In the Copy Formation drop-down list, select an existing formation to copy to the new formation.
4. Click OK.
5. Edit the layout of the formation as described in "Changing a Formation's Layout by Hand," on page 4-33.

4.6.7. Removing Formations

To remove a formation:

1. In the Formation Editor group box, select the tab for the formation that you want to remove.
2. Click the Remove button. The tab is removed.

4.6.8. Renaming Formations

To rename a formation:

1. In the Formation Editor group box, select the tab for the formation that you want to rename.
2. Click the Rename button. The New Name dialog box opens.
3. Type a name for the formation.
4. Click OK.

4.6.9. Specifying the Default Formation

The default formation is the formation assigned to an aggregate 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 Formation Editor group box, select the tab of the formation that you want to specify as the default formation.
2. Click the Default button. The selected tab gets moved to the leftmost position.

4.6.10. 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 entities 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 in the Formation Editor group box.

4.6.11. Displaying Bounding Boxes

You can display bounding boxes for the subordinate aggregates in an aggregate. This may be useful when you collapse the views of individual aggregates, but you still want to have a sense for the area that they occupy. Figure 4-24 illustrates an aggregate with the display of bounding boxes off and on.

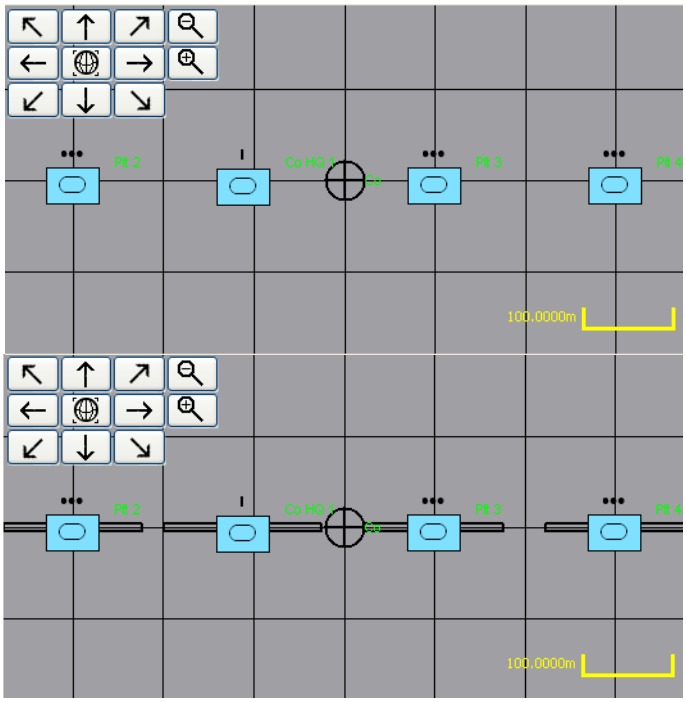


Figure 4-24. Aggregate bounding boxes in formation editor

➤ To display or hide bounding boxes, select or clear the Bounding Boxes check box.

4.6.12. Copying Formations

You can copy the formations from one aggregate to another. When you copy formations, the subordinates in the aggregate are laid out in each formation according to their order in the Aggregate Composition window.

To copy a formation:

1. In the entity list, select the aggregate that you want to copy formations to.
2. In the Formation Editor group box, click Copy Formations. The Select formations dialog box opens (Figure 4-25).

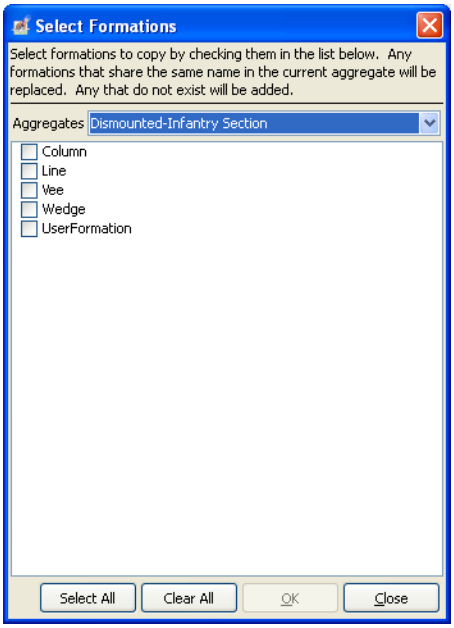


Figure 4-25. Select Formations dialog box

3. In the Aggregates drop-down list, select the aggregate whose formations you want to copy.
4. In the window, select the formations that you want to copy.
5. Click OK. The formations are copied from the selected aggregate to the currently displayed aggregate.

4.6.13. Laying Out a Formation Automatically

You can automatically distribute the entities in an aggregate 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 Formation Editor group box, select the tab for the formation that you want to lay out.
2. Click Auto Layout. The Auto Layout dialog box opens.
3. In the drop-down list, select a formation type.
4. Specify the distance, in meters, between each entity in the formation.
5. Click OK. The entities in the aggregate are reorganized to the new layout.

4.6.14. Changing a Formation's Layout by Hand

You can lay out a formation by hand. The crosshair in the formation editor indicates the center of the formation (Figure 4-26). This is where the aggregate icon is placed when the aggregate is collapsed or aggregated.

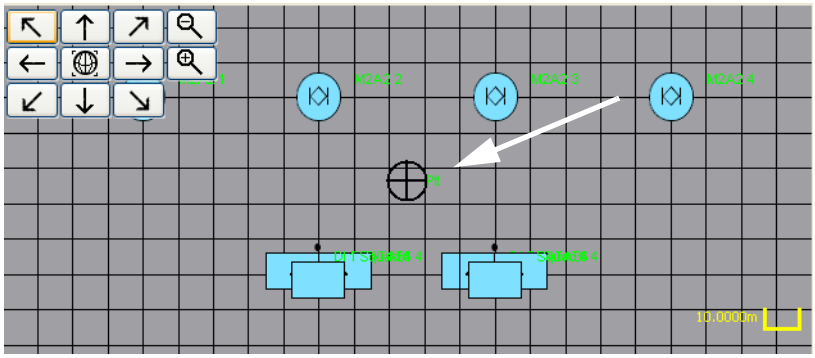


Figure 4-26. Aggregate formation editor crosshair

To change a formation's layout:

1. In the Formation Editor group box, select the tab for the formation that you want to lay out.
2. 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 entities 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 entities 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 Scale Factor

The grid scale factor determines the distance, in meters, between grid lines. The default is 10 meters. When you change the scale factor, the distance between lines is changed by the specified percentage.

➤ To change the grid scale factor, in the Grid Scale Factor box, type a value or change the value using the spinner controls. The grid is resized to the new scale factor and the scale in the editing window shows the new distance between grid lines.

4.7. Editing an Entity's Systems

Entities have complex capabilities, such as the ability to move, to take damage, to sense other entities, and to fire munitions. These capabilities are organized as systems. You can quickly change how an entity functions by adding, removing, and changing its systems. The Entity Editor lists the systems available in each category and only lists systems that are appropriate for the entity type (Figure 4-27). For example, a lifeform can have an M-16 or an RPG, but not a cruise missile.

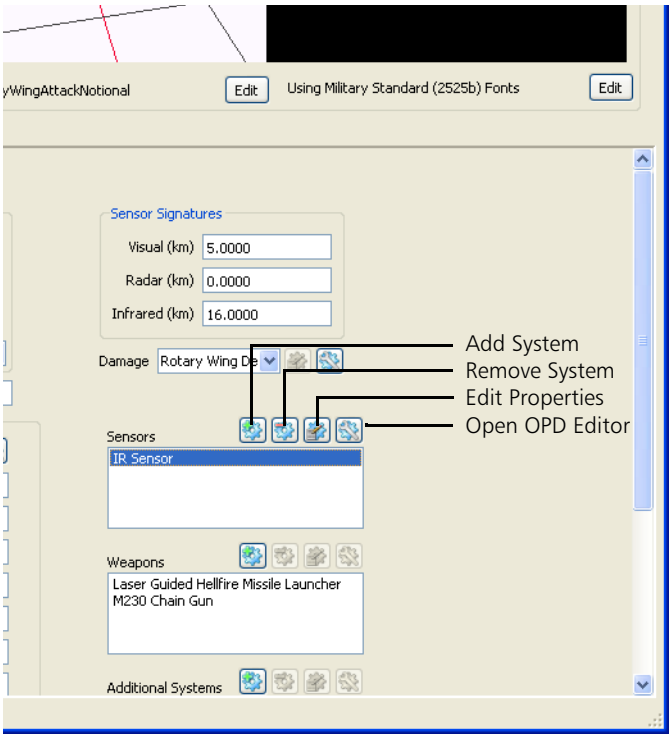


Figure 4-27. Entity systems

Some systems are interdependent. For example, if you add a weapon system to an entity, it cannot use it unless it also has a sensor system so that it can detect hostile entities. 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 Entity Editor lets you edit the following types of systems:

- Damage. You can specify the type of damage model that an entity uses. When you select a damage model, the Entity Editor configures the damage files that are appropriate for the entity type.
- Sensors. You can specify the sensors that an entity uses to locate other entities in the simulation. (By contrast, sensor signatures, which you can configure in the parameter section of the window, specify the distance at which other entity sensors can locate this entity.) When you add a weapon, the Entity Editor configures the components and connections necessary for the sensor to work.
- Weapons. The Entity Editor lists the weapons systems that are appropriate for the vehicle type. When you add a weapon, the Entity Editor configures the components, targets, ammunition, and connections necessary for the weapon to work.
- Movement. You can specify the type of movement system that an entity uses, for example, tracked, wheeled, pedestrian, and so on. When you select a movement system, the Entity Editor configures all of the actuators and controllers that the entity needs to implement the system.
- Other systems. This category includes spot reports and IFF transponders.

4.7.1. Adding a System

There is no specific limit to the number of sensors, weapons, or other systems that you can configure for an entity.

To add a system:

1. Click the Add System button for the system type you want to add (Figure 4-27). The Add System dialog box opens (Figure 4-28). It lists the systems of this type that are available for the entity you are editing. If you select a system in the window, the dialog box displays a brief description.

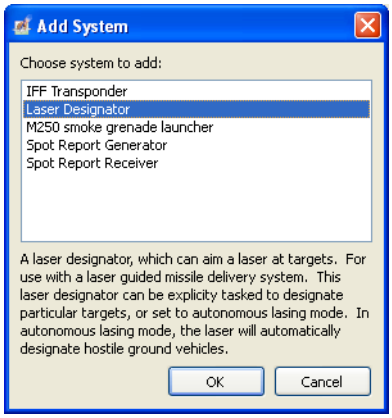


Figure 4-28. Add System dialog box

2. Select the system you want to add.
3. Click OK. The system is added to the list in the Entity Editor.

4.7.2. Selecting a Damage Model

You can only have one damage model specified for an entity.

➤ To specify a damage model, select an option from the Damage drop-down list.

4.7.3. Selecting a Movement Model

You can only have one movement model specified for an entity.

➤ To specify a movement model, in the Movement group box, select an option from the Movement Dynamics drop-down list.

4.7.4. Editing a System's Properties

Some systems have properties that you can edit.

To edit a system's properties:

1. Select the entity 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 Properties button (Figure 4-27). A Properties dialog box opens. The 3D model in the Visual Models group box displays a green dot on the entity where the system is located. In some cases there are multiple locations.
4. Optionally, in the Visual Models group box, select the Marker Labels Visible check box. A label is added to the green dot to identify it.
5. Edit the properties as desired.
6. Click OK.

4.7.5. Editing a System in the OPD Editor

Systems may have attributes that you cannot edit in the Entity Editor. You can jump directly to the OPD Editor from the Entity Editor to edit these attributes. When you open the OPD Editor from the Entity Editor, only the Systems page is displayed. You cannot edit other entity attributes.

Before you jump to the OPD Editor, you should save any changes that you have made to the simulation model set. If you make any changes in the OPD Editor, the Entity Editor is automatically updated after you save the changes in the OPD Editor.

To jump to a system in the OPD Editor:

1. Select the system that you want to edit.
2. Click the Open OPD Editor icon next to the system (Figure 4-27).

4.7.6. Removing a System

You can remove sensor, weapons, 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 (Figure 4-27).

4.8. 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.

To create a completely new entity:

1. Choose **Model** → **New Model**. The Create New Entity dialog box opens.
2. Select an entity category from the Entity to Create drop-down list.
3. Type a name for the entity.
4. Click OK. The Entity Editor adds a new entry to the entity list. The editor shows default values for all the fields.
5. Specify an enumeration for the entity.
6. Specify force values for the new entity.
7. Specify appropriate values for the entity parameters.
8. Type a name to be used in the entity's label in the Short Name text box.
9. Specify a movement system.
10. Specify a damage model.
11. Optionally, add weapons systems.
12. Optionally, add sensors.
13. Save the simulation model set.

4.8.1. 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.

To create a new entity from an existing entity:

1. In the entity 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. Click OK. The new entity is added to the entity list. The enumeration is incremented and an OPE file name is created based on the entity name.
5. Edit the parameters to provide the new entity the characteristics you want for it.
6. Save the simulation model set.

4.9. Deleting an Entity Model from a Model Set

To delete an entity model from a simulation model set:

1. Select the entity in the entity list.
2. Choose **Model** → **Delete Model** or press **`Delete`**.

4.10. Navigating Among Edited Entities

If you have viewed or edited several entities and want to return to a previous entity without finding it in the entities list, you can use the Forward and Backward arrows to navigate among previously edited entities. The arrows act like the forward and back buttons in a web browser or online help viewer.

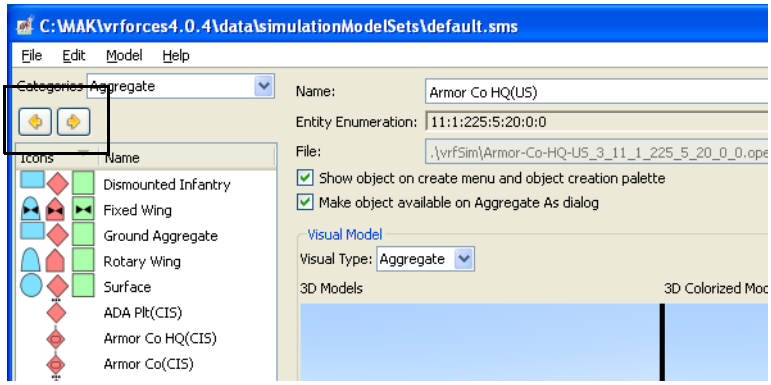


Figure 4-29. Forward and Back arrows

4.11. Editing the Create Menu

By default, VR-Forces automatically generates the Create menu based on the entity category and force assignments in the simulation model set. When you change the category or force assignments in the Entity Editor, these changes affect the automatically generated Create menu and the Entity Palette. Inclusion of an entity in these menus is controlled by the Show Object On Create Menu and Object Creation Palette check box. However, if you want to, you can configure a custom Create menu.



- This procedure only affects the Create menu. It does not affect the organization of the Entity Palette.
- The organization of the Create menu is a characteristic of a simulation model set. Therefore, if you use different SMSs for different scenarios, the Create menu could be different from scenario to scenario.

4.11.1. Generating the Create Menu Automatically

To generate the Create menu automatically:

1. Choose **Edit** → **Edit Create Menu**. The Menu Editor dialog box opens (Figure 4-30).

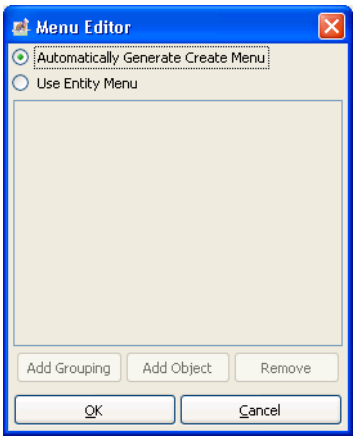


Figure 4-30. Menu Editor

2. Select Automatically Generate Create Menu.
3. Click OK.

4.11.2. Editing the Create Menu Manually

To edit the Create menu manually:

1. Choose **Edit** → **Edit Create Menu**. The Menu Editor dialog box opens.
2. Select Use Entity Menu. The window displays a tree list representing the current Create menu (Figure 4-31).

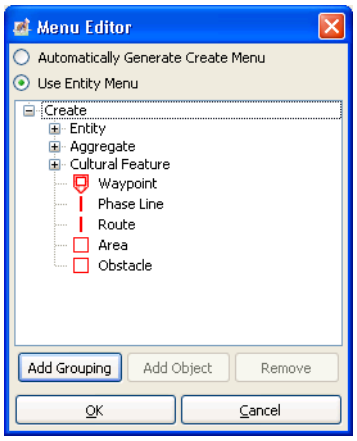


Figure 4-31. Menu Editor

3. To add a grouping:
   a. Select a current grouping.
   b. Click Add Grouping. The Add Group dialog box opens.
   c. Select the location for adding the group: as a sibling, above or below the selection, or as a child.
   d. Click OK. An entry called New Group is added (Figure 4-32).

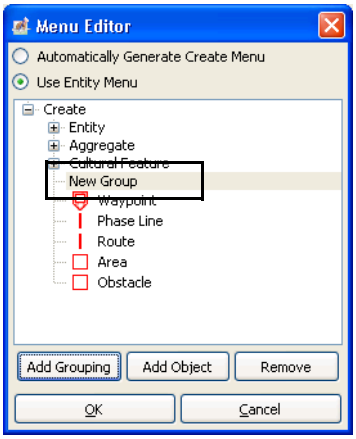


Figure 4-32. Menu Editor with new group added

   e. Double-click New Group to make it editable.
   f. Type a name for the group.
4. To add an object to the Create menu:
   a. Select a group title or an object in one of the groups.
   b. Click Add Object. The Add Object dialog box opens (Figure 4-33).

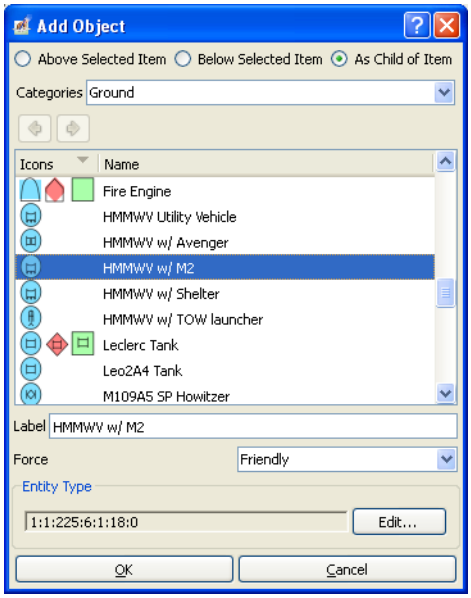


Figure 4-33. Add Object dialog box

c. Select the location where you want to add the object. If you selected a group title, the options are above or below it, or as a child. If you selected an object, the options are above and below it.

d. Select an object to add from the list. Filter the list as necessary to display the categories of objects that you want to add.

e. Optionally, edit the label and force for the object.

f. Click OK. The object is added to the menu (Figure 4-34).

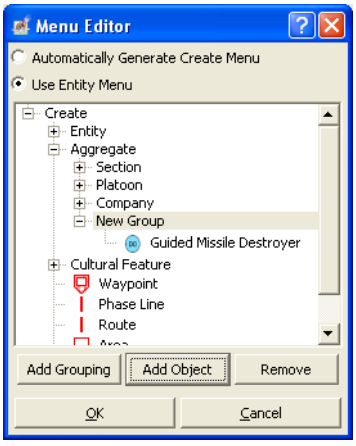


Figure 4-34. Menu Editor with new object added

5. You can edit any of the group or object labels.

6. You can drag an object to another group by dragging it onto the group title.

4.11.3. Undoing (Reverting) Changes to the Create Menu

If you have made changes to the Create menu, but have not saved the simulation model set, you can revert to the original menu.

➤ To undo changes to the Create menu, choose **Edit** → **Revert Create Menu**.

4.12. Editing the Category List

If you want to organize entities into categories that are not included in the default category list, you can add new categories. You can also remove and rename categories.

4.12.1. Adding a New Category

To add a new category:

1. Choose **Edit** → **Edit Categories**. The Categories Editor opens (Figure 4-35).

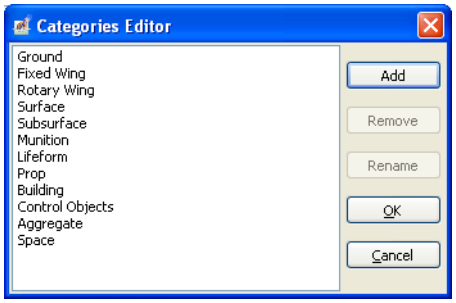


Figure 4-35. Categories Editor

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 an entity to it, it does not get saved when you save the simulation model set.

4.12.2. 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** → **Edit Categories**. The Categories Editor opens.
2. Select the category that you want to remove.
3. Click Remove.
4. Click OK.

4.12.3. Renaming a Category

When you rename a category, all entities that are assigned to the old name are reassigned to the new name. The old name remains in the Categories drop-down list until you save the simulation model set, but it is not listed in the Categories Editor. If you assign an entity 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** → **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.

4.12.4. 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.

4.13. Managing the Forces List

Each entity configuration page has a list of forces. You specify the forces for which an entity can be created. You can add new forces and removing existing forces. You can also change the color assigned to a force.

4.13.1. 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** → **Edit Forces**. The Forces Editor opens (Figure 4-36).

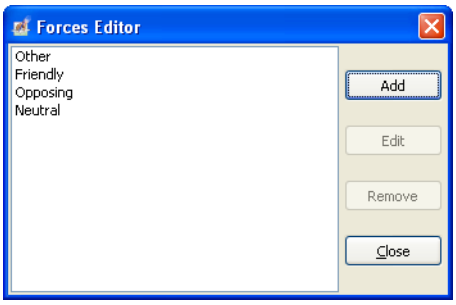


Figure 4-36. Forces Editor

2. Click Add. The Force Hostility Editor opens (Figure 4-37). The new force is assigned a force ID. You cannot edit the force ID.

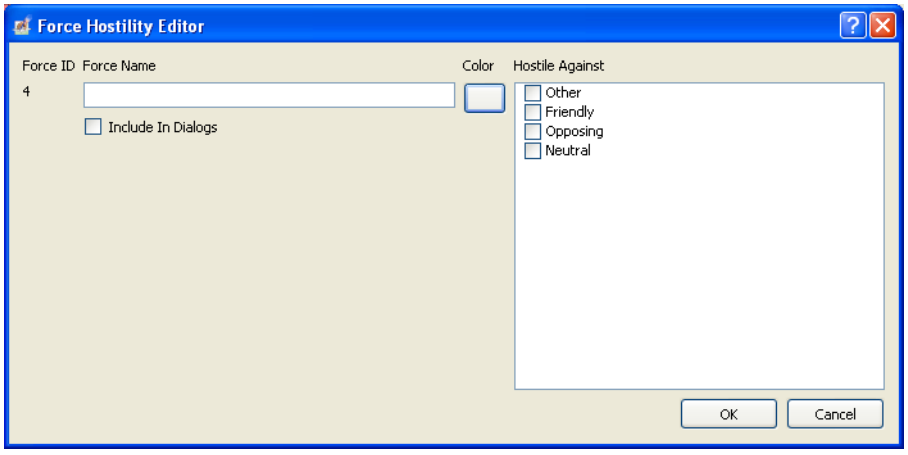


Figure 4-37. Force Hostility Editor

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 Section 9.2, "Managing Force Hostility Relationships", in *VR-Forces Users Guide*.
7. If you want the force to be listed in dialog boxes that list forces (including the Forces list in the Entity Editor), select the Include in Dialogs check box.
8. Click OK.
9. Click Close.

4.13.2. 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:

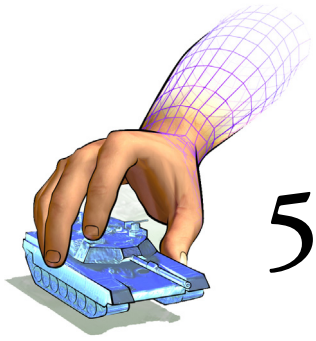
1. Choose **Edit** → **Edit Forces**. The Forces Editor opens (Figure 4-36).
2. Select the force that you want to edit.
3. Click Edit. The Force Hostility Editor opens (Figure 4-37).
4. To edit the name, change the text in the Force Name box.
5. To edit the color associated with the force, click the 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 (including the Forces list in the Entity Editor), select or clear the Include in Dialogs check box.
8. Click OK.
9. Click Close.

4.13.3. Removing a Force

If you create a scenario that has entities 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 entity. 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 entities with the removed force.

To remove a force:

1. Choose **Edit** → **Edit Forces**. The Forces Editor opens (Figure 4-36).
2. Select the force that you want to remove.
3. Click Remove. The force is removed from the list.
4. Click Close.



5

Advanced Entity Model Editing

This chapter describes how to use the OPD Editor to edit the object parameter database.

The Object Parameter Database Editor 5-2
Starting the OPD Editor 5-3
Loading an Object Parameter Database 5-4
- Changing the Object List View 5-5

Editing an Object's Parameters 5-5
- Editing Individual Parameters 5-6
- Editing Parameters for Multiple Objects 5-7
- Deleting Parameters and Components 5-7
- Restoring Parameters 5-7

Creating a New Object from an Existing Object 5-8
Adding New Components and Parameters 5-9
- Adding New Components 5-9
- Editing Components 5-10
- Adding New Resource Parameters 5-11
- Adding New Subordinate Object Parameters 5-12
- Adding New Soil Type Parameters 5-12

Editing Variables 5-13
Editing Systems 5-13
Saving Changes to the Object Parameter Database 5-14

5.1. The Object Parameter Database Editor

The Object Parameter Database Editor (OPD Editor) is a graphical user interface for editing the object parameter database. It gives you access to all the components and entity parameters in 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.



If you want to add new entities or customize the entities supplied with VR-Forces, start with the Entity Editor, which is easier to use than the OPD Editor. Use the OPD Editor only if the Entity Editor does not let you make the changes you need to make.

- When you edit an entity's parameters in the OPD Editor, you are essentially editing the OPE file used by that entity. If more than one entity uses that OPE file, the change will apply to all of the entities.
- Do not try to edit an entity model in the Entity Editor and the OPD Editor at the same time, except as described in "Editing a System in the OPD Editor," on page 4-38. Use one tool, save the changes, and close the tool before opening the other.

5.2. Starting the OPD Editor

➤ To start the OPD Editor, choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.0.4** → **Tools** → **OPD Editor**. The OPD Editor opens (Figure 5-1).

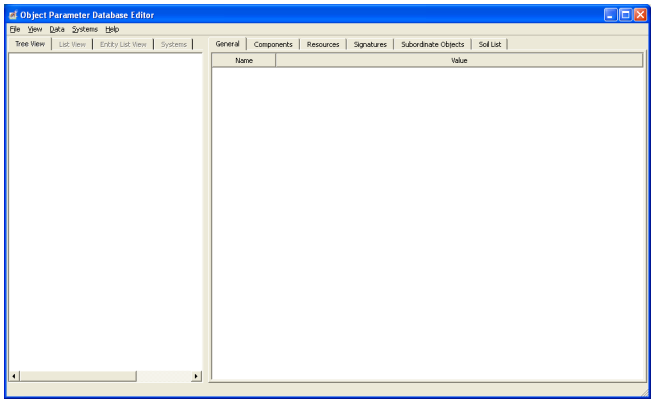


Figure 5-1. OPD Editor

To start the OPD Editor from the Windows or Linux command line:

1. Change directory to *./bin*.
2. Run *vrfOpdEditor*.

5.3. Loading an Object Parameter Database

To edit an object parameter database, you must load it into the editor.

To load an object parameter database:

1. Choose **File** → **Open SMS File**. An Open dialog box opens. By default it opens to *./data/simulationModelSets*.
2. Select the object parameter database file you want to load, for example, *default.sms*.
3. Click Open. The file is loaded and is listed as a tree hierarchy in the list window.

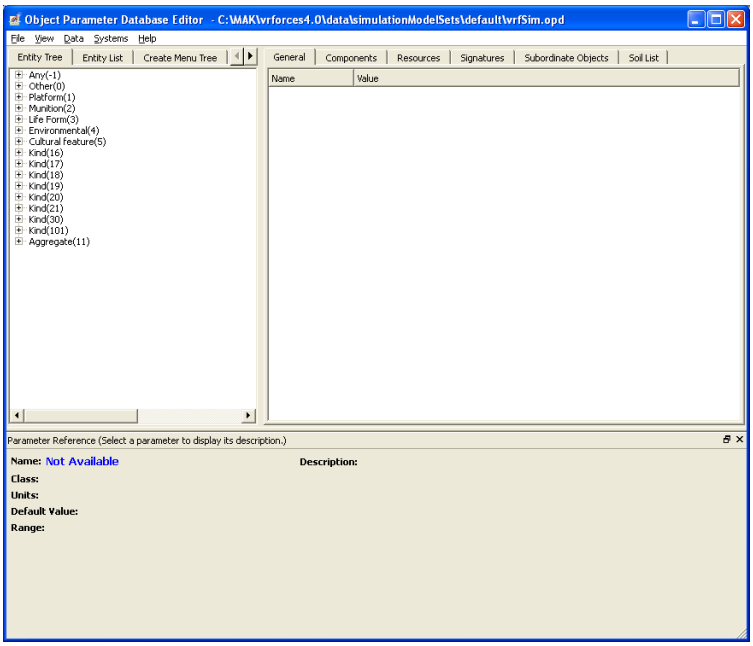


Figure 5-2. OPD Editor with SMS file loaded

5.3.1. Changing the Object List View

The object list has the following views:

- Entity Tree: The hierarchical tree view allows you to configure any enumeration within the DIS/RPR FOM entity enumeration scheme.
- Entity List: The list view lists all objects in the object parameter database.
- Create Menu Tree: The entity list view lists entities and objects organized by categories as they are on the Create Menu.
- Systems List: The systems view lists the systems configured in system definition files.

➤ To change the list view, click the tab for the list you want to view.

5.4. Editing an Object's Parameters

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.

5.4.1. 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 parents is displayed in bold (Figure 5-3).

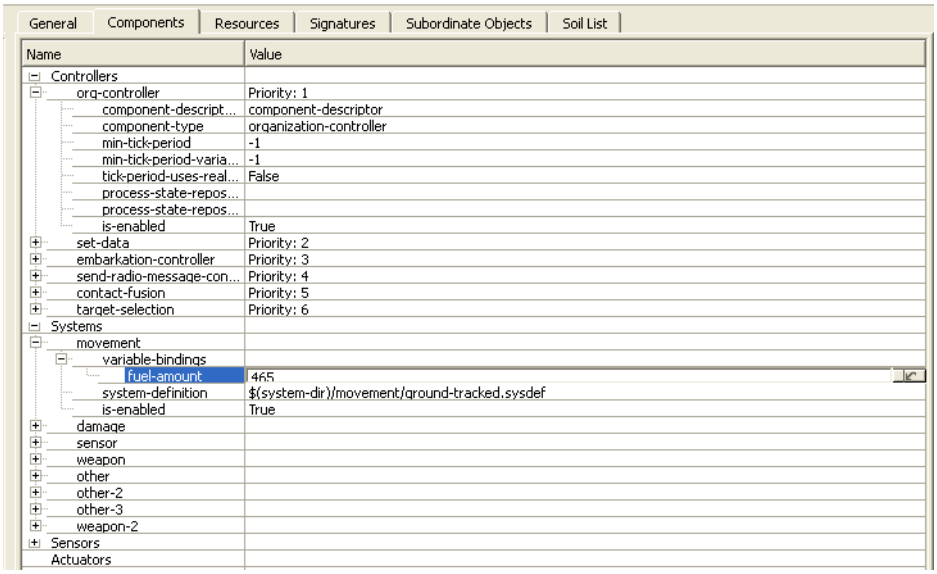


Figure 5-3. Parameter hierarchy

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 icon (Figure 5-4), click the icon to open a File Chooser dialog box. If the value has an edit icon, click the icon to open a window that allows you to edit the file.

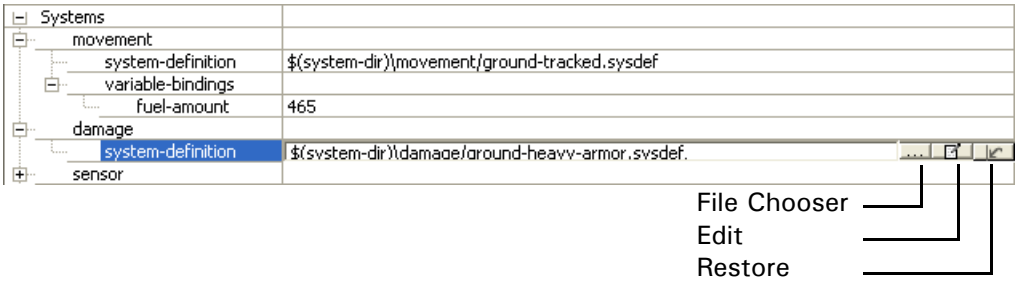


Figure 5-4. Parameter value icons

5.4.2. Editing Parameters for Multiple Objects

To edit parameters for multiple objects:

1. Select the objects that you want to edit by clicking them while holding down **`Ctrl`**. The OPD Editor enters multiple mode. It displays one tab, labeled Multiple. The selected objects do not have to be of the same platform type. If a particular node on the hierarchy has multiple leaf entries, selecting it puts the OPD Editor into multiple edit mode. If there is just one leaf entry, selecting a higher level node leaves the editor in individual edit mode.
2. Edit the parameters on the Multiple tab as you would edit an individual parameter. The changes are applied to all selected objects.

5.4.3. 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. Choose **Data** → **Delete Parameter Entry** or click the Delete icon at the right end of the parameter entry (Figure 5-5).

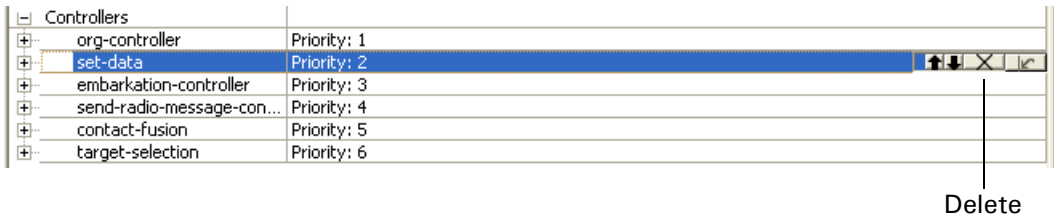


Figure 5-5. Delete parameter icon

5.4.4. 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 5-3). 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 **Data** → **Restore Parameter Entry**, or click the Restore icon (Figure 5-4).

5.5. Creating a New Object from an Existing Object

You can add new entity types to the object parameter database using an existing object as a template.

To add a new object to the object parameter database:

1. Expand the Entity Tree to the leaf level and select the object on which you want to base the new object. (The other views do not support this procedure.)
2. Choose **Data** → **Create From Existing**. The Create Entry dialog box opens (Figure 5-6). It lists the object enumeration for the base object in the Base Entry box.

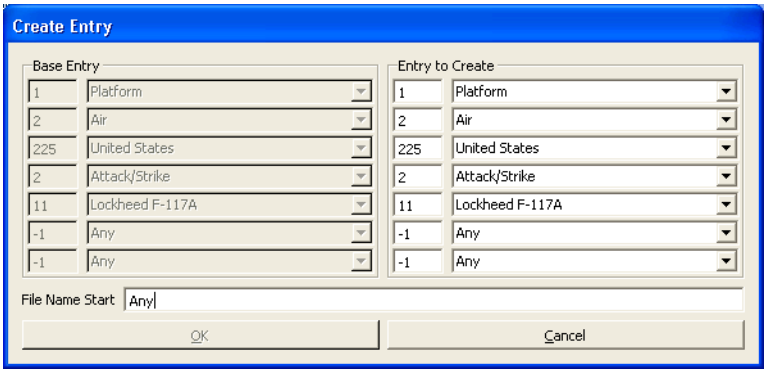


Figure 5-6. Create Entry dialog box

3. In the Entry to Create box, change the enumeration to the one for the object you want to create. You can type the enumeration directly, or build it by selecting options from the lists for each value.
4. Type a filename start value. This is the prefix used for automatically generating entity names.
5. Click OK. The new object is added to the Entity Tree in the appropriate location.
6. Edit the parameters for the new object to give it the proper behaviors.
7. Save the object parameter database.

5.6. Adding New Components and Parameters

You cannot add new parameters to the General tab. You can add new components, resources, subordinates, and soil types if an object supports them.



For most entities, 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 entity. If you do not see a component that you are looking for in an entity's component list, look at the component list for the relevant system in the Systems List.

5.6.1. Adding New Components



Before you add a component to an entity, consider whether or not the component should be added as part of a new or existing system, rather than being added directly to the entity.

To add a component to an object:

1. Select the Components tab.
2. Select the component type (Actuator, Controller, Sensor) you want to add.
3. Choose **Data** → **Create Parameter**, or click the Add icon (Figure 5-7). The Create New Component dialog box opens. The Component list lists all components that are available for this object.

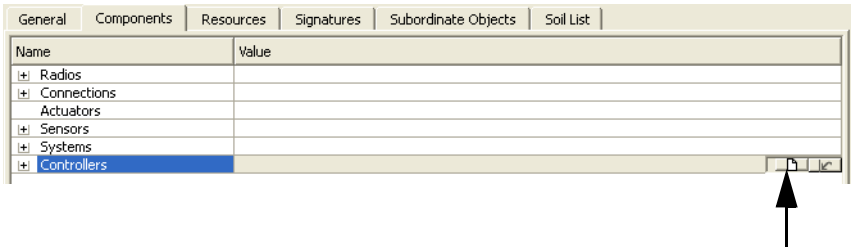


Figure 5-7. Add icon

4. From the Component list, select the component you want to add.
5. Click OK. The component is added to the object.

5.6.2. Editing Components

The Components tab lets you edit actuators, controllers, sensors, and connections. You can edit the individual parameters for components, as described in the other sections of this chapter. You can add and delete components and you can change their priorities.



If you do not see a component that you are looking for in an entity's component list, look at the component list for the relevant system in the Systems List.

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 icon on the value line (Figure 5-8).

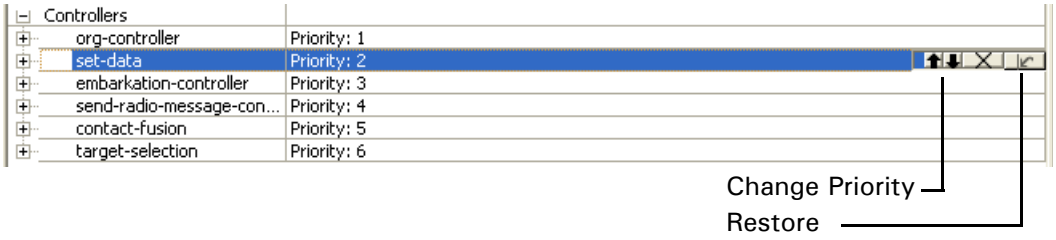


Figure 5-8. Change Priority and Delete Parameter icons

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. Choose **Data** → **Create Parameter**, or click the Add icon (Figure 5-7). The Create Connection dialog box opens (Figure 5-9).

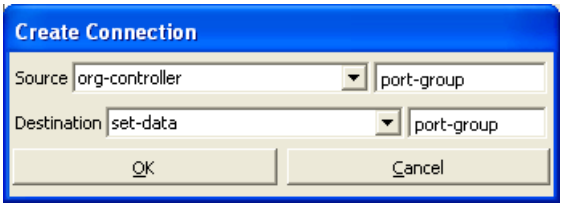


Figure 5-9. Create Connection dialog box

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.

5.6.3. Adding New Resource Parameters

To add a resource parameter to an object:

1. Select the Resources tab.
2. Select Resources.
3. Choose **Data** → **Create Parameter**, or click the Add icon (Figure 5-7). The Create Resource dialog box opens.

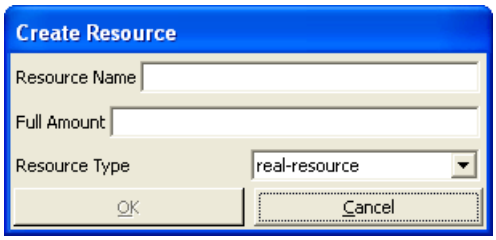


Figure 5-10. Create Resource dialog box

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.

5.6.4. Adding New Subordinate Object Parameters

You are responsible for knowing whether or not a subordinate object is appropriate for a given object.

To add a subordinate object parameter:

1. Select the Subordinate Objects tab.
2. Select Subordinate Objects.
3. Choose **Data** → **Create Parameter**, or click the Add icon (Figure 5-7). A subordinate parameter is added.
4. Expand the Subordinate Objects list and edit the parameters for the new subordinate object.

5.6.5. Adding New Soil Type Parameters

To add a soil type parameter to an object:

1. Select the Soil List tab.
2. Expand the Soil List to see which soil types are configured.
3. Choose **Data** → **Create Parameter**, or click the Add icon (Figure 5-7). The Add Soil Type dialog box opens (Figure 5-11).

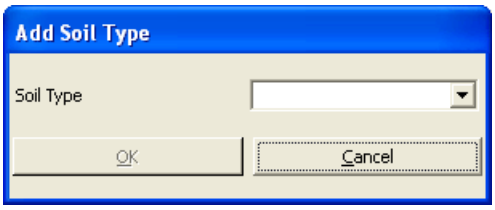


Figure 5-11. Add Soil Type dialog box

4. From the Soil Type list, select the soil type you want to add. The list of soil types is fixed. The Soil Type list contains only the supported soil types that are not already configured for the object. If the list is empty, it means that all supported soil types have already been configured.
5. Click OK. The soil type is added to the object.

5.7. 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," on page 6-22.)

To edit variables:

1. Choose **Data** → **Edit Variables**. The Edit Variables dialog box opens.
2. Change the variable values as desired.
3. Click OK.

5.8. Editing Systems

Systems are packages of components that get incorporated into entity models by reference. Internally, a system is just a list of components that could just as easily be part of the main OPE file for an entity. Consequently, within any selected system you edit components and parameters exactly the same way that you edit components that are listed for a specific entity. The only difference between a system and other component definitions is that systems contain metadata that enables them to be used in the Entity Editor. You edit a system's metadata in the OPD Editor just as you would any other parameters.

An object's systems are listed on its Components tab (Figure 5-12). Systems are also listed on the Systems List tab of the OPD Editor window.

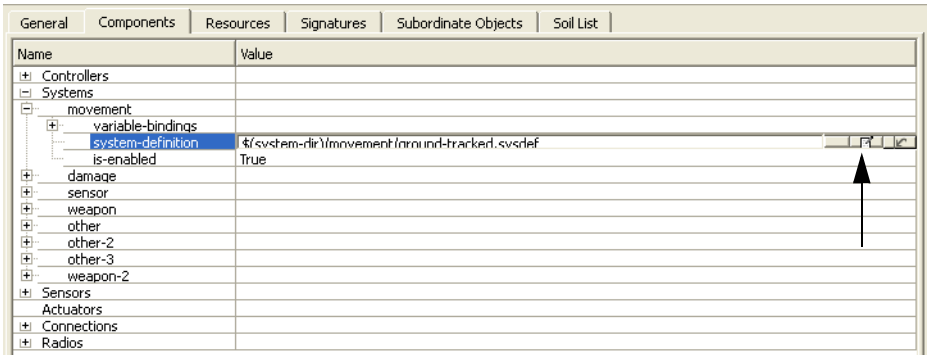


Figure 5-12. Systems list on a Components tab

If you want to edit a system and you have selected it on a Components tab, click the edit button. The OPD Editor changes the view to show the system in the Systems List.

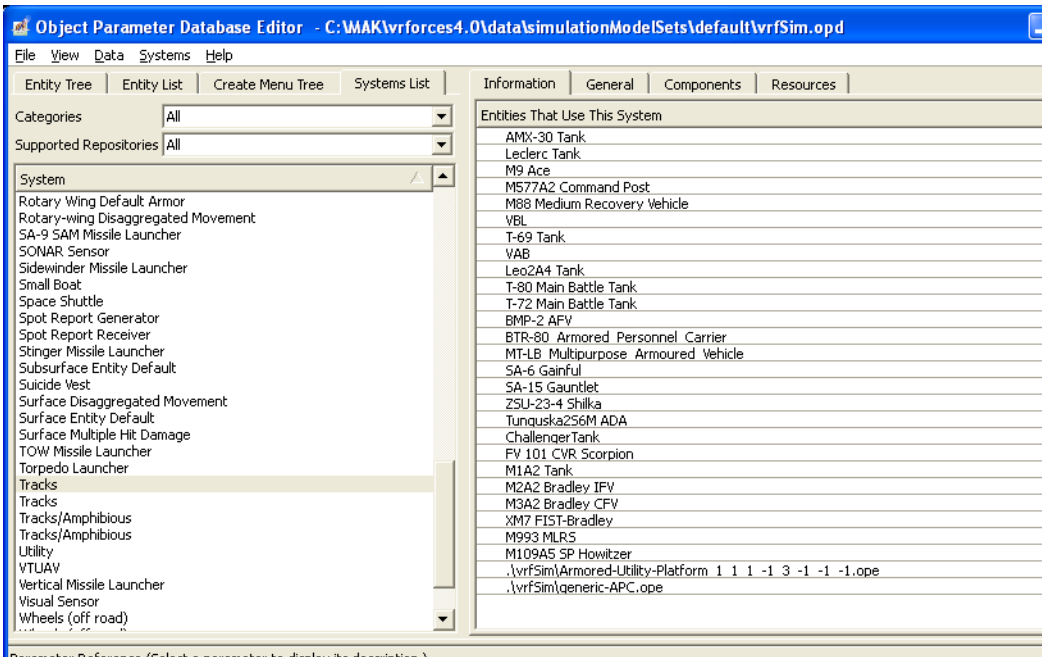


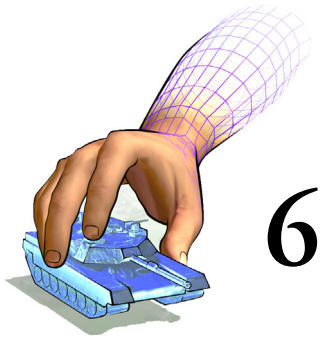
Figure 5-13. System entry in Systems List

If you click on an entry in the Shared OPE Entries tab for a system, it takes you to the OPE entry for that entity.

You can create new systems in the OPD Editor, either by copying an existing system or starting a brand new system. These procedures are the same as those described in "Creating a New Object from an Existing Object," on page 5-8 and "Adding New Components and Parameters," on page 5-9.

5.9. Saving Changes to the Object Parameter Database

➤ To save an edited object parameter database, choose **File** → **Save**.



6

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 6-2
- VR-Forces Classes and Object Parameters 6-6
- Object Types 6-6
- Parameter Types 6-9
- Local and Remote Subcomponents 6-10

Component Descriptors and Parameters 6-13
- Common Elements of Component Descriptors 6-14
- Configuring the Priority of Components 6-16

Adding a New Entity to the Object Parameter Database 6-17

System Definition Files 6-18
- Creating a System Definition File 6-19
- Configuring System Connections 6-19
- Adding Variable Bindings 6-22

Configuring Aggregate Entities 6-23
- Adding an Aggregate to the Object Parameters Database 6-23

Changing the Echelon ID Category and Echelon ID Level 6-26

Configuring Tactical Graphics 6-27

Pathname Interpretation in Simulation Model Set Files 6-27

Updating Object Parameter Database Files for New Releases 6-28

6.1. The Object Parameter Database

VR-Forces entities and objects have many parameters. Parameter values are stored in a set of files 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 an entity and starting entity state values. You can modify the behavior of an entity by changing the parameters for that type of entity. You can add new types of entities 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.

The object parameter database has an object parameter entry (OPE) file for each entity type configured for use by VR-Forces. When you load a scenario, the back-end reads the file *vrfSim.opd*. This file indexes the individual OPE files. When you create an object (either directly or by loading a scenario), VR-Forces reads the parameters for the object from its OPE file. OPE files may reference other files, in particular system definition files.

System definition files group related component and connection specifications into convenient packages that you can reference from OPE files. This makes OPE files easier to read. The system definition files also make it possible to add systems to entities in the Entity Editor.



By default, if successive scenarios use the same object parameter database, VR-Forces does not reload the object parameter database before running each scenario. With the default behavior, if you are making changes to the object parameter database, you must close VR-Forces and restart it to put the changes into effect. However, if you set **forceParameterDbReload** to true, VR-Forces reloads the object parameter database each time you load a scenario.

Example 6.1 is a highly edited example of an OPE file that includes the main features discussed in this manual. The relevant segments of the example (indicated by arrows) are cross referenced to the sections that discuss them.

Example 6.1

```
1    ;; Entry for M1A2
2    (M1A2
3      (parameter-type "ground-vehicle-param");; “Parameter Types,” on
       page 6-9
4      (bounding-geometry-type Volume)
5      (bounding-volume
6        (local-bvol  3.994358 1.929996 0.896801)
7        (offset  0.000000 0.000000 -0.896801)
```

```
8       )
9       (formation "Other")
10      (force ForceOther)
11      (category  "M1A2");; “Changing the Echelon ID Category and Echelon
        ID Level,” on page 6-26
12      (echelon-level  "")
13      (is-organized True)
14      (min-tick-period  0.000000);; 15 Hz ;; “Specifying the Tick Rate
        for Components,” on page 1-10
15      (min-tick-period-variance  0.000000)
16      (dr-algorithm Rvw)
17      (radios
18        (main-radio
19          (radio-descriptor-type "radio-descriptor")
20          (radio-type  "default-radio")
21          (comm-model-name  "default-radio-model")
22          (publish-transmitter False)
23          (transmitter-params
24            (radio-type  7 1 225 2 1 20)
25            (initial-frequency  3000000.000000)
26            (power  30.000000)
27          )
28        )
29      )
30      (systems
31        (movement
32          (variable-bindings
33            (DtRwReal
34              (fuel-amount  465.000000
35              )
36            )
37          )
38          (system-definition ;; “System Definition Files,” on page 6-18
39            (filename  "$(system-dir)/movement/ground-tracked.sysdef")
40          )
41          (is-enabled True)
42          )
43        (sensor
44          (variable-bindings
45            (DtRwVector
46              (sensor-position  -1.140000 0.570000 -2.900000
47              )
48            )
49            (DtRwReal
50              (max-range  4000.000000
51              )
52            )
53          )
54          (system-definition
55            (filename  "$(system-dir)/sensors/visual-sensor.sysdef")
56          )
57          (is-enabled True)
58        )
59        (weapon
60          (variable-bindings
61            (DtRwInt
62              (num-ap-rounds  20
63            )
64            )
65            (DtRwInt
```

```
66              (num-heat-rounds  20
67              )
68            )
69            ...
70          )
71          (system-definition
72            (filename  "$(system-dir)/weapons/120mm-gun.sysdef")
73          )
74          (is-enabled True)
75        )
76        (other
77          (system-definition
78            (filename  "$(system-dir)/other/spot-report-
      receiver.sysdef")
79          )
80          (is-enabled True)
81        )
82      )
83      (sensors
84        (under-fire-determination-sensor ;; “Configuring Sensors,” on
      page 7-3
85          (component-descriptor-type "under-fire-determination-sensor-
      descriptor")
86          (component-type  "under-fire-determination-sensor")
87          (min-tick-period  -1.000000)
88          (min-tick-period-variance  -1.000000)
89          (tick-period-uses-real-time False)
90          (process-state-repository-name  "")
91          (process-state-repository-type  "")
92          (is-enabled True)
93          (under-fire-time  120.000000)
94          (under-fire-distance  1000.000000)
95          (under-fire-from-friendly True)
96        )
97      )
98      (controllers ;; “Component Descriptors and Parameters,” on
      page 6-13
99        ...
100       (set-data
101         (component-descriptor-type "component-descriptor")
102         (component-type  "local-entity-set-controller")
103         (min-tick-period  -1.000000)
104         (min-tick-period-variance  -1.000000)
105         (tick-period-uses-real-time False)
106         (process-state-repository-name  "")
107         (process-state-repository-type  "")
108         (is-enabled True)
109       )
110        (embarkation-controller
111         (component-descriptor-type "embarkation-descriptor")
112         (component-type  "embarkation-controller")
113         (min-tick-period  -1.000000)
114         (min-tick-period-variance  -1.000000)
115         (tick-period-uses-real-time False)
116         (process-state-repository-name  "")
117         (process-state-repository-type  "")
118         (is-enabled True)
119         (response-timeout  1.000000)
120         (response-retries  2)
121       )
```

```
122       ...
123     )
124     (actuators )
125     (connections ;; “Common Elements of Component Descriptors,” on
        page 6-14
126       (connect target-selection:weapon weapon:target-list)
127       (connect target-selection:object-types-to-detect sensor:object-
        types-to-detect)
128       (connect sensor:detected-objects contact-fusion:detected-
        objects)
129       (connect target-selection:weapon-2 weapon-2:target-list)
130       (connect other:detected-objects contact-fusion:detected-objects)
131     )
132     (state-repository-extension-type  "default-vrf-state-repository-
        user-extension")
133   ...
134     (max-acceleration  1.270000) ;; Appendix B, Object Parameters
135     (max-deceleration  4.000000)
136     (fuel-efficiency  1000.000000)
137     (object-geometry-file
138       (filename  "")
139     )
140     (soil-list
141       (paved-road
142         (acceleration-factor  1.000000)
143         (stopping-factor  1.000000)
144         (max-speed-factor  1.000000)
145       )
146       (hard-packed
147         (acceleration-factor  0.720000)
148         (stopping-factor  0.720000)
149         (max-speed-factor  0.720000)
150       )
151     )
152     (local-objects ;; “Local and Remote Subcomponents,” on page 6-10
153       (state-repository  "ground-entity-state-repository")
154       (state-repository-min-tick-period  -1.000000)
155       (state-repository-min-tick-period-variance  -1.000000)
156       (net-interface  "individual-local-entity-net-interface")
157       (net-interface-min-tick-period  -1.000000);; “Tuning the Network
        Interface,” on page 1-11
158       (net-interface-min-tick-period-variance  -1.000000)
159       (task-manager  "local-task-manager")
160       (component-manager  "component-manager")
161       (plan-manager  "vrfobject-plan-manager")
162     )
163     (remote-objects
164       ... )
165     (subordinate-objects )
166     (attached-objects )
167     (user-extension )
168   )
```

6.1.1. VR-Forces Classes and Object Parameters

When you create an object in VR-Forces, it creates an instance of an object defined by a VR-Forces class. A VR-Forces object class has relationships with other classes that are instantiated along with the object. For example, in the VR-Forces API, each *DtVrfObject* "has a" *DtVrfObjectBaseStateRepository*.

The entries for each object in the object parameter database correspond to VR-Forces classes and tell the simulation engine what type of object to create. For example, if you specify the **parameter-type** "ground-vehicle-param" for an entity, VR-Forces knows to use the *DtGroundVehicleParameters* class when it creates the entity.

6.1.2. 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 aggregate objects.

This enumeration scheme describes a tree that goes from a more general specification to a more detailed one. Figure 6-1 illustrates a partial entity type tree. To build an entity 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 entity enumeration scheme, please see the DIS Enumerations Document (*Enumeration and Bit Encoded Values for Use with Protocols for DIS Applications*).

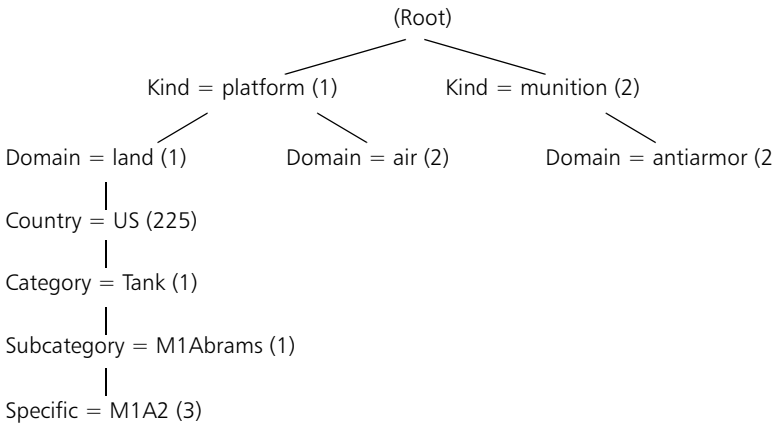


Figure 6-1. Entity type enumeration tree

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 aggregate entities and tactical graphics. We enclose the seven-digit enumeration in parentheses and preface it with the object super-type (Figure 6-2). For example, the entity enumeration for an M1A2, described in the previous paragraph, becomes the object type: 1 (1 1 225 1 1 3 0).

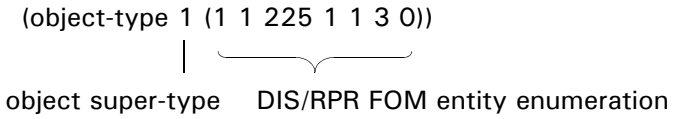


Figure 6-2. VR-Forces object-type parameter

Table 6-1 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 6-1: Object super-type field values

Object super-type	Meaning
0	Other
1	Individual (platform entities, individual tactical graphics)
2	Aggregate
3	Pseudo-aggregate
4	True Aggregate

The Object Type Hierarchy

Each simulated object on the Create menu is associated with an **object-type** (for details, please see "Adding a New Entity or Control Object to vrf.ent," on page A-5.) When you create an entity, the front-end sends this information to the back-end. The back-end searches through the object parameter database and instantiates the entity with the proper set of values for the **object-type**.

The **object-type** sent by the front-end does not need to exactly match an **object-type** in the object parameter database. Objects in the object parameter database are specified in a hierarchy in which entries at the top 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.)

An object parameter database might have the following entries:

```
(parameter-entry
  (object-type  -1 (-1 -1 -1 -1 -1 -1 -1))
  (filename ".\vrfSim\base-sim-object_-1_-1_-1_-1_-1_-1_-1_-1.ope")
)
(parameter-entry
  (object-type  1 (3 1 -1 -1 -1 -1 -1))
    (filename
    ".\vrfSim\generic-lifeform-platform.ope")
)
(parameter-entry
  (object-type  1 (3 1 225 1 1 -1 -1))
  (filename
    ".\vrfSim\friendly-dismounted-infantry.ope")
)
```

As you replace each wildcard value (-1) with a specific enumeration, the scope of the object type narrows. With the illustrated hierarchy, an object with an object type of 1(3 1 225 -1 -1 -1 -1) would have a best match at the generic-lifeform-platform entry. The friendly-dismounted-infantry entry would be too specific for this example enumeration. The order in which parameters are specified in the object parameter database does not affect the hierarchy.

6.1.3. 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 6-3 illustrates the object parameter type hierarchy.

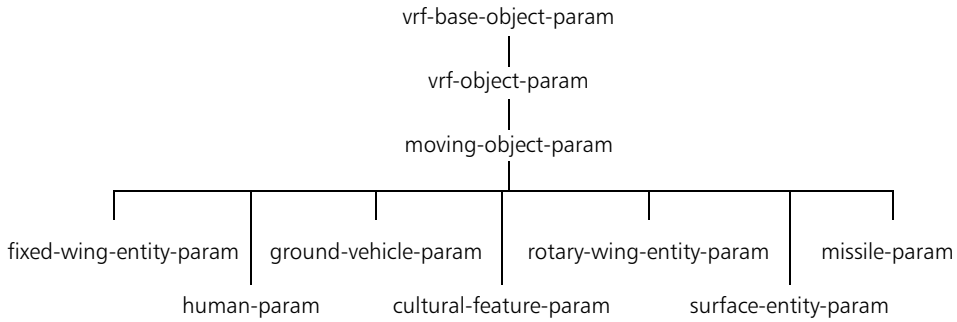


Figure 6-3. Object parameter hierarchy

The **parameter-type** specifies the set of parameters appropriate to an entity, and the entity state repository maintains state information for an entity, so the **parameter-type** and state repository must match. For more information, please see "The State Repository Hierarchy," on page 6-12.

6.1.4. Local and Remote Subcomponents

As explained in Section 3.4.2, "Local Objects and Remote Objects", in *VR-Forces Users Guide*, 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 6-2 maps the subcomponent classes to the parameters that specify them.

Table 6-2: 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 6-3 lists the permissible entries for each parameter listed in Table 6-2.

Table 6-3: 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.
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. Most objects need a net interface. Other subcomponents are optional. However, for most entities, you will want to specify a task manager, plan manager, and component manager so that you can control the entity and give it tasks.

Use the existing entries in the object parameter database as guidance if you create a new entity or change an existing entry. A typical entry for local and remote objects is:

```
(local-objects
  (state-repository "ground-entity-state-repository")
  (net-interface "individual-local-entity-net-interface")
  (task-manager "local-task-manager")
  (component-manager "component-manager")
  (plan-manager  "vrfobject-plan-manager")
)
(remote-objects
  (state-repository "vrf-moving-object-state-repository")
  (net-interface "vrf-individual-remote-entity-net-interface")
  (task-manager "")
  (plan-manager "")
  (component-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, in the object parameter database, each object must specify a state repository within its parameter block and the state repository must match the parameter type for the object.

Table 6-4 lists parameter types and the state repository type required.

Table 6-4: 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
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 entities 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 entities provided with VR-Forces.

6.2. Component Descriptors and Parameters

The parameters for an entity include a list of all the sensor, controller, and actuator components it uses. (For details, please see Section 3.5.4, "VR-Forces Uses a Component Architecture" in *VR-Forces Users Guide*.) These components might be in the OPE file or in a system definition file.

A component descriptor is a text block in the object parameter database that describes a component (see Example 6.2). 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. 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.) "Common Elements of Component Descriptors," on page 6-14 describes the construction of a component descriptor.

Components can receive data from, and convey data to, other components through ports. In an entity's entry in the object parameter database, you specify which ports a component sends output to and receives data from.

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 Section 5.9, "Ports and Port Groups", in *VR-Forces Developers Guide*.

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 6-16.

6.2.1. 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 entity. It gets used in the connections section of the entity's parameters. Other components may use this name to find the specified component. In Example 6.2, 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 6.2, 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 6.2, 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 6.2, line 48) 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 an entity 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 6.2 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.

All of the entity components that you can use in the object parameter database are described in the online help for the OPD Editor.

Example 6.2 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 6.2

```
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       (stay-in-lane False)
15       (road-center-offset  2.000000)
16     )
17     (patrol-between-points
18       (component-descriptor-type "ground-move-to-descriptor")
19       (component-type  "ground-auto-patrol-between-controller")
20       (min-tick-period  -1.000000)
21       (min-tick-period-variance  -1.000000)
22       (tick-period-uses-real-time False)
23       (process-state-repository-name  "")
24       (process-state-repository-type  "")
25       (is-enabled True)
26       (near-distance  25.000000)
27       (at-distance  1.000000)
28       (approach-speed  2.000000)
29       (stay-in-lane False)
30       (road-center-offset  2.000000)
31     )
32   ) ;; end controllers
33   (actuators
34     (powertrain
```

```
35        (component-descriptor-type "automotive-component-descriptor")
36        (component-type  "automotive-actuator")
37        (min-tick-period  -1.000000)
38        (min-tick-period-variance  -1.000000)
39        (tick-period-uses-real-time False)
40        (process-state-repository-name  "")
41        (process-state-repository-type  "")
42        (is-enabled True)
43        (art-part-list )
44        (check-terrain-collisions False)
45        (use-high-fidelity-ground-clamping True)
46      )
47    ) ;; end actuators
48    (connections
49      (connect move-along:automotive-control powertrain:automotive-
          control)
50      (connect patrol-between-points:automotive-control
          powertrain:automotive-control)
51    ) ;; end connections
```

The connection in line 46 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.

6.2.2. Configuring the Priority of Components

Each simulation frame, VR-Forces ticks each entity's components in the order in which they are listed in the object parameter database. 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).

For more information about prioritizing components, please see Section 5.4.6, "Setting the Priority of Components" in *VR-Forces Developers Guide*.

6.3. Adding a New Entity to the Object Parameter Database

This section explains how to manually add new entries to the object parameter database. The easiest way is to copy the OPE file for a similar entity and change parameters to give it the characteristics you want for the new entity.



The recommended way to add new entities to the object parameters database is to use the Entity Editor (see "Creating a New Entity," on page 4-39) or the OPD Editor (see "Creating a New Object from an Existing Object," on page 5-8.)

Before making changes to a configuration file, be sure to make a backup copy of the original file.

The object parameter database has an entry for an M1A2 tank, but not an M1A1. If you want to add an M1A1 you could:

1. Copy the *M1A2_1_1_1_225_1_1_3_-1.ope* file and rename it *M1A1_1_1_1_225_1_1_2_-1.ope*.
2. Open *M1A1_1_1_1_225_1_1_2_-1.ope*.
3. Change the vehicle name in the very first line of the block.
4. Change the **category** string to M1A1. (For more information, please see "Changing the Echelon ID Category and Echelon ID Level," on page 6-26.)
5. Change other parameters as desired and add or delete controllers and actuators to give the new vehicle the characteristics of a M1A1.
6. If you add controllers or actuators, add any necessary connections.
7. Save the *M1A1_1_1_1_225_1_1_2_-1.ope* file.
8. In the OPD file, add a reference to the new OPE file.

The following code snippet shows the start and end of the parameter block for the M1A2 vehicle. The minimum changes that you have to make to turn it into an M1A1 (name, object-type, and category) are all in the first few lines of the parameter block.

```
(M1A2 ;; Change to M1A1
  (parameter-type "ground-vehicle-param")
  (bounding-geometry-type Volume)
  (bounding-volume
    (local-bvol  3.994358 1.929996 0.896801)
    (offset  0.000000 0.000000 -0.896801)
  )
  (formation "Other")
  (force ForceOther)
  (category "M1A2") ;; Change to M1A1
```

```
  (echelon-level "")
  ...
```

The entry in the OPD file would be:

```
(parameter-entry
  (object-type  1 (1 1 225 1 1 2 -1))
  (filename  ".\vrfSim\M1A1_1_1_1_225_1_1_2_-1.ope")
)
```

6.4. System Definition Files

The Entity Editor allows VR-Forces users to easily add component systems, such as weapons, sensors, and movement, to entities without needing to understand the components, configuration files, and connections required to make these systems work. Editing of systems in the Entity Editor is facilitated by system definition files (*.sysdef*) in the object parameter database. System definition files are referenced in an entity's OPE file. So, as a byproduct of the use of system definition files, OPE files can be shorter and easier to read. System definition files 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 an entity. The data could just as easily be written into the OPE file and have the same effect.

The metadata explains to the Entity Editor how to present the system to the user and how to integrate it into the entity model.

Given this information about system definition files, if you are customizing the object parameter database by adding newly developed components or by using the components provided with VR-Forces, you have the following options:

- Add the new component configuration directly into OPE files. In this case, the new components will not be accessible from the Entity Editor, but could be edited in the OPD Editor.
- Package the new component information in a system definition file without metadata and reference the system definition file from the OPE file. Again, the system will not be accessible from the Entity Editor, but can be edited in the OPD Editor.
- Package the new component information in a system definition file and include metadata. The system will be accessible from the Entity Editor.

6.4.1. 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, the format is similar to that of the OPE file and other configuration files that use the MTL syntax. 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 list the connection among the components.

6.4.2. Configuring System Connections

Within each system definition file, you list the connections among the components in the file. (For general information about connections, please see "Common Elements of Component Descriptors," on page 6-14.) 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, weapons systems must have access to a target list. This section explains how to make these out-of-system connections. (The sample code is from the *M1A2_1_1_1_225_1_1_3_-1.ope* file and the *120mm-gun.sysdef* file.)

To understand how system definition files and OPE files are integrated, we will look at:

1. A weapon definition in the OPE file.
2. The related system definition file.
3. The connection section in the OPE file.

The M1A2 is configured with a 120mm gun, using a system definition file.

```
(weapon
  (variable-bindings
    (DtRwInt
      (num-ap-rounds  20
      )
    )
    ...
    (DtRwInt
      (gun-art-part-type  4416
      )
    )
    ...
  (system-definition
    (filename  "$(system-dir)\weapons/120mm-gun.sysdef")
  )
)
```

This block is named weapon. It contains several variable bindings. The values attributed to the variables are used in the system definition file. The file itself is specified in the system-definition block. For more information, please see "Adding Variable Bindings," on page 6-22.

The system definition file has the following relevant code:

```
(weapon-120mm-gun
  (controllers
    ...
  )
  (actuators
    (main-gun-joy
      (component-descriptor-type "ballistic-gun-descriptor")
      ...
      (is-enabled True)
      (art-part-list
        (primary-gun
          (art-part-type $gun-art-part-type)
          (parent-part-type $turret-art-part-type)
          (attach-point $turret-attach-point)
          (art-part-param-list
            (part-type  13)
            (part-type  14)
          )
        )
      )
    ...
  )
  (connections
    (connect system:target-list turret-acquire:target-list)
    (connect main-gun-control-joy:gun-control main-gun-joy:gun-
  control)
    ...
  )
  (meta-data
    ...
    (parameter-data-list
      (art-part-parameter-data
        (parameter-name "gun-art-part-type")
        (variable-type "DtRwInt")
```

```
34            (display-label "Gun Part Number")
35            (display-units “”)
36            (source-units “”)
37            (art-part-category "gun")
38         )
39      )
40   )
```

The M1A2 OPE entry for weapon creates a variable called gun-art-part-type (line 9). The variable is used in line 12 of the system definition file. Other variables specified in the OPE file are used throughout the system definition file.

The system definition file defines connections for its components. The first connection in the list (line 24) maps system to target-list. If you look at the complete system definition file, you will see that there is no component called system. Nor is there one called target-list.

The system component in the first connection (line 24) is a self reference. It links to an equivalent connection in the main OPE file.

The use of system in a connection in a system definition file creates a port into the system so that components outside the system can connect to it. Line 24 indicates that a target-list component needs to connect to the weapon system through the system port.

The connections in the M1A2 OPE file are as follows:

```
(connections
     (connect target-selection:weapon weapon:target-list)
     (connect target-selection:object-types-to-detect
       sensor:object-types-to-detect)
     ...
)
```

In the first line in the list of connections, we see that the target-selection component sends output to its weapon output port. (The weapon output port is defined in the target-selection component’s block, which is not shown in this section.) The weapon component gets input through its target-list input port. The weapon component is really a reference to the 120mm-gun system. The first connection in the 120mm-gun.sysdef file is:

```
(connect system:target-list turret-acquire:target-list)
```

In this line, “system” is a self-reference that is equivalent to the component name in the OPE file. Therefore, this line could be read as the weapon component sends output (which it received through its target-list input port in the OPE file connection) to its target-list output port. Then the turret-acquire component in the system definition file, takes input through its target-list port. (Figure 6-4) illustrates these connections.

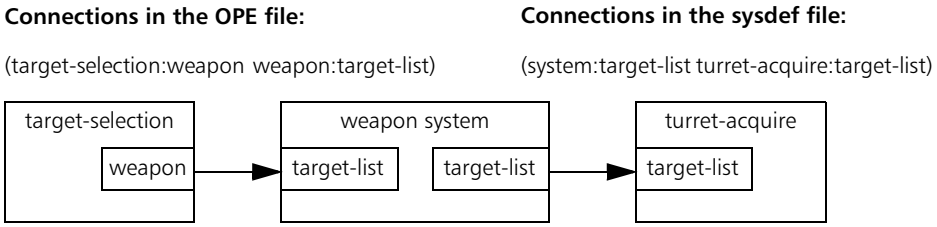


Figure 6-4. Flow of data in component connection

6.4.3. Adding Variable Bindings

Most system definition files are used by multiple entities. However, the values of some parameters in a system definition file (SDF) may need to vary for the different entities that use it. As previously noted, this is handled by using variables for the values in the system definition and specifying the values for variables in the OPE files. VR-Forces uses variable bindings for the parameters that you are most likely to want to change from entity to entity. If there are other parameters that you want to vary, you can add your own variable bindings.

The following example shows a variable binding for a movement system:

```
(movement
  (variable-bindings
    (DtRwReal
      (fuel-amount  462.000000
      )
    )
  )
  (system-definition
    (filename  "$(system-dir)\movement/ground-tracked.sysdef")
  )
)
```

The variable is used in the SDF as follows:

```
(resources
  (fuel
    (resource-type "real-resource")
    (subresources )
    (amount  $fuel-amount)
    (full-amount  $fuel-amount)
  )
)
```

As you can see, the variable defined as fuel-amount in the OPE file is referenced as $fuel-amount in the SDF file.

To add a variable binding to an 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.
2. In each OPE file that uses this SDF, add a variable binding to the system block that references the SDF you edited. The variable binding must specify the type of the value, the name of the variable, and the value for the variable as shown in the example earlier in this section.

If you do not add a variable binding to an OPE that references the modified SDF, a default value will be used, which may result in unexpected behavior.

6.5. Configuring Aggregate Entities

All aggregate entities have the **parameter-type** "moving-object-param". Aggregate entities have relatively few individual parameters. The principle function of the various parameter blocks is to specify controllers and formations, and to specify the **echelon-level** for the different levels of aggregates.

6.5.1. Adding an Aggregate to the Object Parameters Database

This section explains how to add a preconfigured aggregate to the object parameter database by directly editing the object parameter database. You can also add new entries with the OPD Editor (see Chapter 5, *Advanced Entity Model Editing*.) The procedure for adding a preconfigured aggregate to the object parameter database is similar to that of adding an entity, except that you must ensure that any subordinate units required by your new aggregate exist in the object parameter database. For example, VR-Forces provides preconfigured squads, platoons, and companies. If you wanted to add a preconfigured brigade, which had several battalions, you would have to add both brigades and battalions to the object parameter database, because the default object parameter database does not have any battalions.

The preconfigured aggregates in the object parameter database are based on the aggregate entries used when you aggregate entities using the Aggregate As command. Although they are not all available through the Aggregate As command, the object parameter database has entries for aggregates up through Corps.

The format for a preconfigured aggregate is:

```
(aggregate-local-entity
  (parameter-type "moving-object-param")
  ...
  (category "")
  (echelon-level "echelon_level")
  (subordinate-objects
    (sub1
      (object-type 3 (11 1 225 14 3 0 0))
      (position-offset 0.00000 0.000000 0.00000)
      (heading-offset 0)
    )
    ...
    (subn)
  )
)
```

where:

- **parameter-type** must be moving-object-param.
- **category** should be left blank.
- **echelon-level** describes the hierarchical unit for this aggregate.
- The subordinate-objects block lists the entities that you want to be created as subordinates of this aggregate entity. In each subordinate block:
  - **object-type** specifies the object type of an entity defined elsewhere in the object parameter database
  - **position-offset** specifies the location of the entity in aggregate body coordinates
  - **heading-offset** specifies the heading of the entity relative to the aggregate.

Editing a Preconfigured Aggregate

If you want to change the configuration of a preconfigured aggregate or add a variation of a provided type to the object parameter database, edit one of the current object parameter database entries as follows:

1. Copy the aggregate you want to edit to a new OPE file.
2. Change the subordinate list as desired.
3. Change the entity type enumeration to distinguish the entity from other aggregates of its type. The best way to do this is to change the last number in the enumeration.
4. Add a new entry to the *entity.mst* file so that the aggregate will be editable from the Entity Editor and be added to the Entity Palette. If you are using a customized Create menu (as opposed to having it generated automatically), add the new aggregate to *vrf.ent*.

Adding a New Type of Preconfigured Aggregate

This section explains how to add a new type of preconfigured aggregate, such as a battalion, to the object parameter database.

To add a new preconfigured aggregate to the object parameter database:

1. Copy the existing object parameter database entry for the desired hierarchical unit, in this case, the battalion, which looks like:

```
(aggregate-local-entity
  (parameter-type "moving-object-param")
  ...
  (echelon-level "Bn")
)
```

2. Save the copied file to a new OPE file.
3. Add a category parameter.

```
(category "")
```

4. Add a subordinate-objects block.
5. Add entries for each subordinate entity, for example, four companies. Using the existing preconfigured company as a subordinate, an entry would look like:

```
(sub1
  (object-type 3 (11 1 225 5 -1 -1 1))
  (position-offset 0.00000 0.000000 0.00000)
  (heading-offset 0)
)
```

   The resulting entry might look like:

```
(aggregate-local-entity
  (parameter-type "moving-object-param")
  (object-type  3 (11 -1 -1 6 -1 -1 -1))
  (category "")
  (echelon-level "Bn")
  (subordinate-objects
    (sub1
      (object-type 3 (11 1 225 5 -1 -1 1))
      (position-offset 0.00000 0.000000 0.00000)
      (heading-offset 0)
    )
    (sub2
      (object-type 3 (11 1 225 5 -1 -1 1))
      (position-offset 1000.00000 0.000000 0.00000)
      (heading-offset 0)
    )
    ...
  )
)
```

6. Add a new entry to the *vrfSim.opd* file with an appropriate object type value.

7. Add a new entry to the *entity.mst* file so that the aggregate will be editable from the Entity Editor and be added to the Entity Palette. If you are using a customized Create menu (as opposed to having it generated automatically), add the new aggregate to *vrf.ent*.

6.6. Changing the Echelon ID Category and Echelon ID Level

You can change the category and echelon ID level used in echelon IDs by editing the object parameter database. Changes to the object parameter database take effect the next time you start VR-Forces.

! Before making changes to a configuration file, be sure to make a backup copy of the original file.

To change the category or echelon name used for a platform, such as a tank:

1. Open the OPE file for the entity in a text editor.
2. Edit the value for the **category** or **echelon-level** parameter.
3. Save the file.

---

i You can also add new entries with the OPD Editor. For details, please see Chapter 5, *Advanced Entity Model Editing*.

---

i

- Entity names have the following restrictions on their length:
  - Entities – 11 characters
  - Aggregates – 10 characters in HLA exercises, 32 characters in DIS.
- Since the category is used to build the default entity name, make sure that it is short enough, in combination with the incremental entity number (for example, `M1A2 23`), that it will not exceed the character limits for entity names. If the default entity name is too long, it gets truncated and will not work properly.
- If you change the **category** or **echelon-level** parameters (or both) for any entry, existing scenarios that reference this type of object will not work unless you edit the order of battle file, object map file, and plan file so that the object names use the new **category** and **echelon-level** strings.

6.7. 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.

6.8. Pathname Interpretation in Simulation Model Set Files

The files that make up a simulation model set, such as the SMS file, the OPD file, and OPE 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 *default.sms* file show a binding (model-set-directory) and its use:

```
(simulation-model-set
  (variable-bindings
    (DtRwString
      (model-set-directory  "default")
    )
  )
  (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 OPE files, as shown.

From *vrfSim.opd*:

```
(variable-bindings
  (DtRwString
    (system-dir  "../data/simulationModelSets/default/vrfSim/systems")
    )
    (DtRwString
      (ammoselect-dir
  "../data/simulationModelSets/default/vrfSim/ammoselect")
    )
    (DtRwString
      (damage-dir  "../data/simulationModelSets/default/vrfSim/damage")
    )
    ...
)
```

From *BMP2_1_1_1_222_2_2_1_-1.ope* (system-dir variable):

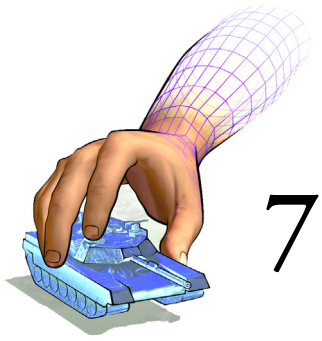
```
(systems
  (tracked-movement
    (variable-bindings
      (DtRwReal
```

```
            (fuel-amount  462.000000)
          )
        )
        (system-definition
          (filename  "$(system-dir)\tracked-movement.sysdef")
        )
      )
      ...
)
```

6.9. 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 object parameter database 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. We recommend using one or a combination of the following methods:

- Start with the object parameter database for the latest release and add your customizations to it.
- Use a text comparison tool (diff tool) to merge your customized files with the release version of the object parameter database.
- To merge in new parameters for existing components, in the OPD Editor, select the specific entities that you have customized, then save the object parameter database. The new parameters will be added with default values.
- The only way to add new components to an object parameter database or replace outmoded components (for example, to replace the target detection sensor with the new sensor mechanism in an older object parameter database) is to add them by hand in individual files or with the OPD Editor.
- Use multiple simulation model sets. VR-Forces can load multiple simulation model sets (for details, please see Section 1.1.1, "Specifying Multiple Simulation Model Sets", in *VR-Forces Scenario Management Guide*.) 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.



7

Configuring Object Behaviors

This chapter describes configuration files that affect entity behavior and the effects of munitions.

Configuring Sensors 7-3
- Configuring a Sensor in an OPE File 7-3
- Editing a Sensor System Definition File 7-4
- Specifying a Sensor Signature 7-5
- Adding a New Sensor Type 7-5

Detection Probability Tables 7-8

Probability and Selection Tables for Munitions 7-9
- Ammo Select Tables 7-9
- Hit Probability Tables 7-10
- Damage Probability Tables 7-12
- Configuring an Entity to Take Damage from a Munition 7-13

Configuring Formations 7-14
- Formation Files 7-15
- Creating a User-Defined Formation 7-16

Configuring Emitters 7-17

Configuring Radios 7-19

Configuring Sensor Volumes 7-19
- Configuring Sensor Volume Color 7-20
- Controlling Sensor Volume Radius 7-21
- Configuring Sensor Volume Segments 7-22

Configuring Task Execution Rules 7-25

Configuring Joysticks 7-26

Configuring Obstruction Avoidance 7-27

Configuring Object Types 7-27
Configuring Vector Feature Types 7-28
Configuring Embarkation 7-29
Editing Embarkation Parameters 7-30
Configuring VR-Forces Components on Remote Entities 7-32

7.1. Configuring Sensors

VR-Forces includes several sensor systems and you can add them to entities using the Entity Editor or the OPD Editor. This section provides additional details about how sensors are configured in the OPE file and system definition file. A complete sensor configuration requires the following:

- One or more system definition entries in the sensor block in the sensing entity's OPE file.
- An entry in the connections section of the entity's OPE file.
- A sensor signature in the OPE file for each entity that can be sensed by a sensor type.
- A signature propagator entry in the *./data/simulationModelSets/<model_set>/physicalWorldParam.mtl* file (one per sensor type.)

VR-Forces includes radar, infrared, visual, and sonar sensors. The entities 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 Section 5.2.6, "Sensors", in *VR-Forces Developers Guide*.

7.1.1. Configuring a Sensor in an OPE File

All sensors provided with VR-Forces are signature-object-sensors. They are distinguished by the domain in which they operate (visual, radar, infrared, or sonar.)

To add a sensor to an entity:

1. Copy an existing sensor block in the OPE file and paste it into the sensors section.
2. Change the name of the system definition file to point to the system you want to add.
3. Change the values for the variables that must be specified for the system.

Connecting a Sensor

If you add a 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 following is from the connections block for an M1A2. The sensor gets input through its object-types-to-detect input port. It sends that data out through its detected-objects output port:

```
(connect target-selection:object-types-to-detect sensor:object-types-to-
  detect)
(connect sensor:detected-objects contact-fusion:detected-objects)
```

If you were to add, for example, an infrared sensor to an entity and you called the sensor block infrared-sensor, you would add the following line:

```
(connect infrared-sensor:detected-objects target-selection:detected-
  objects)
```

7.1.2. 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 that are set in the variable binding section of the OPE file.

7.1.3. Specifying a Sensor Signature

A sensor signature indicates that an entity can be detected by other entities that have a sensor for that signature. The value of the signature using the standard VR-Forces propagator represents the maximum distance, in kilometers, at which the entity can be detected. However, this value is affected by modifiers and the calculations made by the propagator.

The modifiers specify multipliers for that base value based on the current state of the object. For example, an entity 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 will be 7.5 when moving and 5.0 when not moving.

The following is the sensor signature block from an M1A2:

```
(sensor-signatures
  (visual
    (base-signature  4.000000)
    (modifiers )
  )
  (radar
    (base-signature  27.000000)
    (modifiers )
  )
  (infrared
    (base-signature  4.000000)
    (modifiers )
  )
)
```

This sensor-signature block shows that an M1A2 can be detected by visual, infrared, and radar sensors. If you do not want an entity to be detectable by a particular type of sensor, remove it from the OPE file. If you want it to be detectable by an additional sensor type, add it to the OPE file. You can edit the values of sensor signatures in the Entity Editor. You cannot add sensor signatures in the Entity Editor. You can edit the values of sensor-signatures and add new sensor signatures in the OPD Editor.

7.1.4. Adding a New Sensor Type

You can easily add new sensor types to your entities. 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 block to each entity that can detect targets acoustically and connect the sensor to the detected-objects port.
3. Add a sensor signature to each entity 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 an entity 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 "Configuring a Sensor in an OPE File," on page 7-3. 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)
)
```

The connection for the acoustic sensor would look like:

```
(connect target-selection:object-types-to-detect sensor:object-types-to-
  detect)
(connect sensor:detected-objects contact-fusion:detected-objects)
```

Adding a Sensor Signature

If you add a new sensor, you need to add a signature for it to each entity 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 entities at a greater distance than you can hear them, so we have a lower base-signature value.

7.2. Detection Probability Tables

The detection probability tables determine the likelihood that an entity will be detected by another entity. The detection probability is used in the Detect condition in plans.

Detection probability tables are comma-delimited files that are most easily read in a spreadsheet. As illustrated in Figure 7-1, 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 an entity's absolute signature after being modified by factors such as terrain and distance. For details, please see Section 7.1.1, "Signature Sensor Concepts", in *VR-Forces Developers Guide*.)

	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 7-1. Detection table displayed in a spreadsheet

The detection probability is determined by finding the intersection of the apparent signature and the detection interval. For example, if an entity with an apparent signature of 3 has been detected for 40 seconds, its detection certainty level is 4 (full knowledge).

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 Section 9.3, "Detecting Targets", in *VR-Forces Scenario Management Guide*.

7.3. Probability and Selection Tables for Munitions

When the entities in a 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).

7.3.1. 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 wildcards.
- **ammo** – list of ammunition entries. The names of the ammunition entries are assumed to be the resource names. If the resource name is found in the Object Resource Manager, the amount of the ammunition is decremented appropriately, and is not used if the resource amount is 0. In this case, the next ammunition entry (if any) in the ammunition list is used. Each ammunition entry has the following fields:
  - **munition-type** – 7 digit entity type enumeration. This is the entity 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)
         )
      )
   )
)
```

7.3.2. 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)
        )
      )
    )
  )
)
```

7.3.3. 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 7-10. The damage actuator provided by MÄK looks for a probability entry with the name **catastrophic-kill**.



The range for missiles is the range from the missile 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 entities can be damaged.

7.3.4. 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.

7.4. 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 Entity Editor.

The object parameter database lists the formations available to an aggregate entity in the **aggregate-formation-controller** block in its OPE file. Each formation is identified by its name and the relative path to the formation file, for example:

```
(aggregate-formation-controller
  (component-descriptor-type "formation-controller-descriptor")
  (component-type  "aggregate-formation-controller")
  ...
  (formation-list
    (column-formation
      (formation Column)
      (formation-file
        (filename ".\formation\platoonColumn.frm")
    )
    (line-formation
      (formation Line)
      (formation-file
        (filename ".\formation\platoonLine.frm")
      )
    )
...
```

7.4.1. Formation Files

A formation file specifies the name of the formation and has an entry for each entity 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 entities are promoted to aggregate leader if the leader is incapacitated.
- The **leader-promotion-id** specifies the aggregate 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 entity follows the entity in front of it by 25 meters. Therefore, the **leader-promotion-id** for each entry is the **promotion-id** of the entity 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 entities into an aggregate 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 entity 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 entities.

If you want to change the layout of the formations provided with VR-Forces, you can edit the formation files.

7.4.2. 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 Entity Editor. If you use the Entity Editor, you do not have to edit any formation files or OPE 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 OPE file for each aggregate 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:

```
(my-formation-1
  (formation myFormation)
  (formation-file
    (filename ".\formation\myformation.frm")
  )
)
```

We recommend that you configure your formations so that entities 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.

7.5. 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 (*./vrforces4.0.4/data/simulationModel-Sets/default/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
      ...
    )
  )
)
```

7.6. Configuring Radios

All radios are *DtVrfRadio* classes or subclasses. The radio is responsible for managing incoming and outgoing radio messages on a single entity. Each radio is associated with a specific communication model, and uses this communication model to send and receive messages.

Radios are configured in an entity's OPE file. The radios block in the OPE file lists all the radios on the entity 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 OPE radio entry is as follows:

```
(main-radio
  (radio-descriptor-type "radio-descriptor")
  (radio-type  "default-radio")
  (comm-model-name  "default-radio-model")
  (publish-transmitter False)
  (transmitter-params
    (radio-type  7 1 225 2 1 20)
    (initial-frequency  3000000.000000)
    (power  30.000000)
  )
)
```

7.7. Configuring Sensor Volumes

You can configure how VR-Forces colors sensor volumes, how they are drawn, and how they are scaled for visibility. You can configure sensor volumes by editing the DefaultEllipsoidArc model definition or by creating customized ellipsoid arc model definitions and applying them to different entity models.

7.7.1. Configuring Sensor Volume Color

By default, sensor volumes are displayed using fixed colors for high frequency and low frequency. You can configure sensor volumes to be colored based on the emission frequency or pulse rate. When sensor 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 sensor volume is red. As frequency rises above the specified low value, sensor 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 sensor volume is violet.

To change coloring from the default, you must add the appropriate attributes to the sensor volume visualizer for the entities whose model definitions you want to change. Table 7-1 describes the sensor volume visualizer attributes that affect sensor volume color.

Table 7-1: Sensor 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 sensor 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.

7.7.2. Controlling Sensor Volume Radius

A sensor 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 sensor 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 sensor volume visualizer for the entities whose model definitions you want to change:

- linearScale. The amount of linear scaling applied to all sensor volumes. Default: 100.
- exponentialScale. The amount of exponential scaling applied to sensor volumes: Default: 0.1.

How VR-Forces Calculates the Radius

VR-Forces calculates sensor 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 );
```

7.7.3. Configuring Sensor Volume Segments

VR-Forces displays a sensor volume as a group of spherical sections projecting from the sensing device. By default, for each 10 degrees (or portion thereof) in the sensor's field of view, VR-Forces draws one segment. The figure shows segments set to 10° (the default) and 30°.

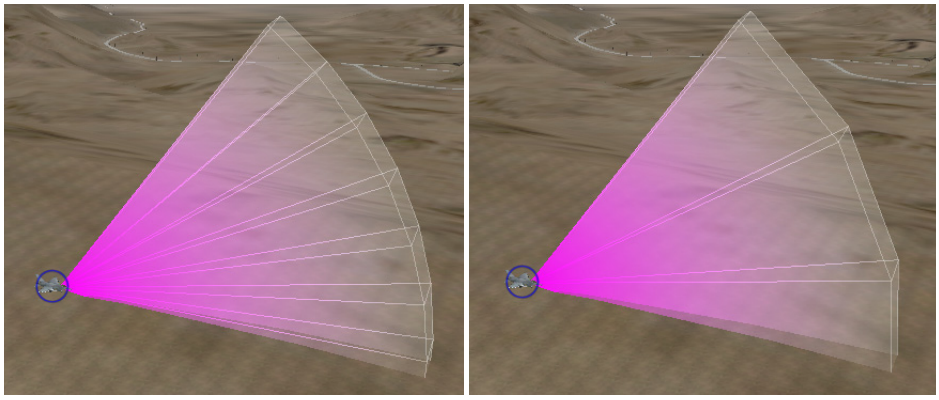


Figure 7-2. Sensor volume segments

Sensor 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 sensor 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 icon (▾) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
4. Select DefaultEllipsoidArc.
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 sensor volume.

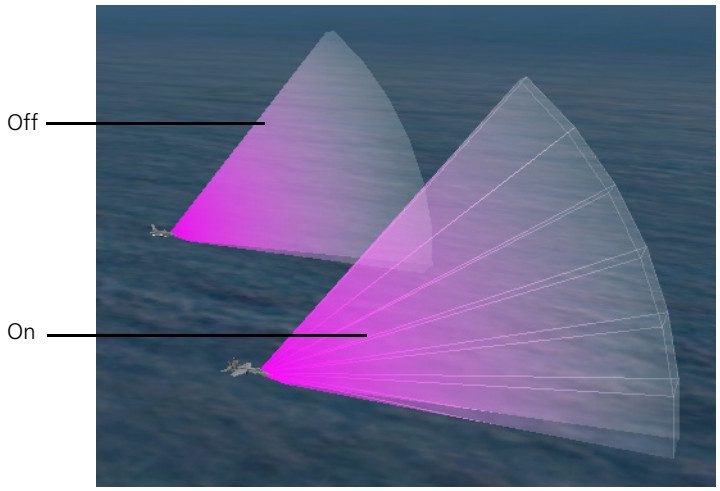


Figure 7-3. Segment outlines

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 icon ( ) 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 icon (▾) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
4. Select DefaultEllipsoidArc.
5. Click the Duplicate Model Definition icon (). 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 sensor 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 drop-down 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.

7.8. Configuring Task Execution Rules

Each entity OPE file contains an entry for a task execution rule configuration file, named **task-execution-rules**. (The default rules file is *./data/simulationModel-Sets/default/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** 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** 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 entity.

7.9. Configuring Joysticks

The *./appData/settings/vrfSim/joyParam.dat* file specifies the default values for joystick parameters. The parameters are:

- **joy-device-type** – Possible values are:
  - `direct-input-joydevice` – used for direct-input joysticks on Windows
  - `linux-joydevice` – used for joysticks on Linux.

  Default: `direct-input-joydevice`.
- **joy-trans-sensitivity** and **joy-rot-sensitivity**- affect the translation and rotation displacement required to make VR-Forces respond. The value should be between 0 and 1.0, with 0 being not sensitive at all, and 1.0 being the most sensitive it can be.
- **joy-trans-gain** and **joy-rot-gain** – affect the velocity of motion for a given displacement. The value is a multiplier, so the default value of 1.0 has no effect on the gain.
- **joy-trans-invert-x** (y, z) and **joy-rot-invert-x** (y, z) – allow you to invert axes on an individual basis.



- The joystick must be attached to the computer on which the entity is being simulated.
- Joystick configuration parameters are documented in the online help for the OPD Editor and in the context sensitive parameter descriptions for joystick controllers in the OPD Editor.

You can override the default values for joystick parameters for specific entity types in the object parameter database. Change the values in the joystick parameter block for the system definition file.

```
(joystick
  (component-descriptor-type "base-joy-controller-descriptor")
  (component-type "ground-move-joystick-controller")
  (joy-trans-sensitivity 0.3)
  (joy-rot-sensitivity 0.3)
  (joy-trans-gain 0.1)
  (joy-rot-gain 0.2)
)
(turret-joystick
  (component-descriptor-type "base-joy-controller-descriptor")
  (component-type  "ground-turret-joystick-controller")
  (joy-trans-sensitivity 0.8)
  (joy-rot-sensitivity 0.8)
  (joy-trans-gain 0.1)
  (joy-rot-gain 0.2)
  )
)
```

The rotary wing controller has some extra values that you can set, for example:

```
(joystick
  (component-descriptor-type "rotary-wing-joy-controller-descriptor")
```

```
  (component-type "rotary-wing-joy-flight-controller")

  ;; controller rise time. Default: 1.5
  ;; rotary-wing-joy-controller-descriptor only
  (controller-rise-time 1.6)

  ;; pitch feedback constant. Default: 1.0
  ;; rotary-wing-joy-controller-descriptor only
  (pitch-control-feedback-constant 1.0)

  ;; roll feedback constant. Default: 2.2
  ;; rotary-wing-joy-controller-descriptor only
  (roll-control-feedback-constant 2.265)

  ;; yaw feedback constant. Default: 2
  ;; rotary-wing-joy-controller-descriptor only
  (yaw-control-feedback-constant 2.1)
)
```

7.10. 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.

7.10.1. 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
)
```

7.10.2. Configuring Vector Feature Types

Configuring avoidance of vector features relies on feature specifications in terrain databases. To configure a vector feature type for obstruction avoidance, modify the appropriate **vector-type** parameter in the **vector-obstruction-type** parameter block for the obstruction sensor in an entity's object parameter database entry.

The vector obstruction types are used to create *DtSpecifications*, which are then used by the vector network intersection metrics to determine if obstructions are the correct types or not. Any attribute specified for avoidance must be in the vector feature for it to be considered an obstruction. Because the vector feature may contain additional properties, simple vector obstruction types can map to a large number of features.

For example, to specify all types of buildings, add a **vector-type** parameter with the single attribute:

```
(vector-obstruction-type
  (vector-type
    (string-attributes
      (DtFeatureGroup "MAK000")
    )
  )
)
```



For details about the MÄK specification model, please see Appendix A, *The MÄK Specification Model* in *MAK Terrain Database Tool Users Guide*.

If you only want to avoid a specific feature, create a more detailed vector-type. For example, the following will cause only standard lights to be considered obstructions:

```
(vector-obstruction-type
  ;; Light Standard (street light) detailed spec
  ;; illustrates how to create a detailed obstruction type
  (vector-type
    (integer-attributes
      (DtFidCode 537)
      (DtFeatureType 0)
      (DtSmc 1)
      (DtSedrisCodes 0)
      (DtOrientation 0)
      (DtHeight 3)
    )
    (string-attributes
      (DtFeatureName "537 Light Standard")
      (DtFeatureGroupName "Power")
      (DtFeatureGroup "MAK050")
      (DtDescription "US-Light Standard/Light Support  UK-Light
        Standard/Light Support/Lamp Post")
      (ECC "AL110")
    )
    (real-attributes
      (GDB_Width 0.0)
      (GDB_Length 2.0)
    )
  ))
```

7.11. Configuring Embarkation

Entities that are configured to allow other 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 identifies a location on the entity where an embarked entity can be placed. The embarkation-slot-entries 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.

7.11.1. Editing Embarkation Parameters

You can edit embarkation parameters in the OPD Editor and the Entity Editor. Ingress and egress points are specified as X, Y, Z values. This section provides a very general introduction to editing embarkation. Please contact MÄK technical support for more information.

The Entity Editor lets you edit single ingress and egress points; it does not let you edit embarkation routes. When you edit ingress and egress points in the Entity Editor, it does not show you the existing points. In fact, it clears out all existing ingress and egress values before it allows you to add new values. Therefore, it is recommended that you use the Entity Editor to edit ingress and egress points only if you know precisely what you want to do.

The OPD Editor lets you edit all aspects of embarkation. To edit embarkation parameters:

1. Select the entity whose embarkation values you want to edit.
2. Select the Components tab.
3. Expand the Controllers list.
4. Expand the occupancy-director-controller component.



Only entities that are configured to allow embarkation, such as the LCAC and M939A2 5-Ton Truck, have this component.

5. Edit values as desired.
6. Save the changes.

Adding a New Ingress or Egress Point

To add a new ingress or egress point:

1. Expand the occupancy-director-controller component as explained in the previous procedure.
2. Select the ingress-points parameter or egress-points parameter, as desired (Figure 7-4).

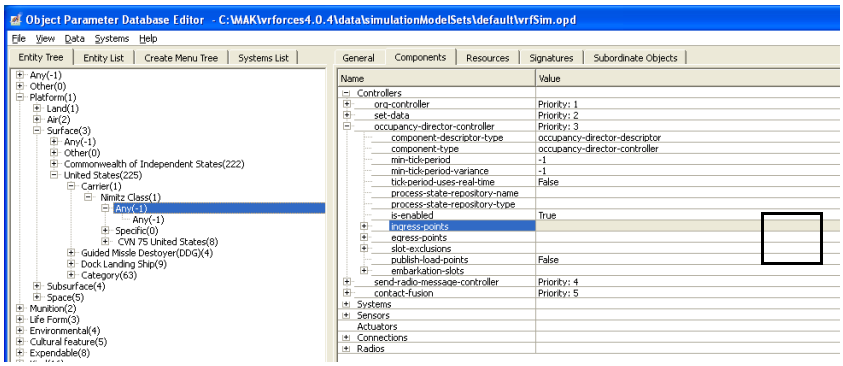


Figure 7-4. occupancy-director-controller component

3. Click the New button. The Set Load Points dialog box opens (Figure 7-5).

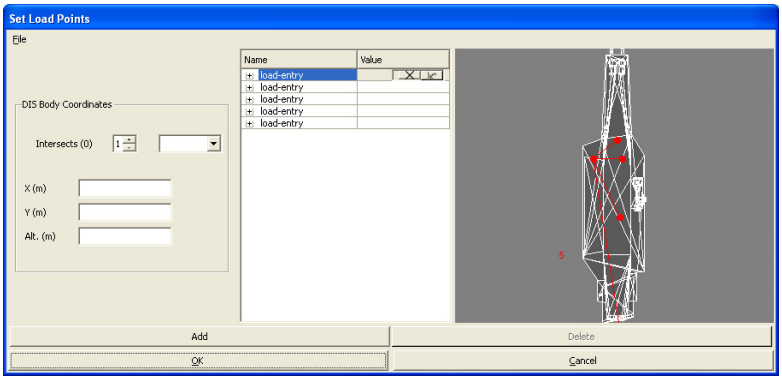


Figure 7-5. Set Load Points dialog box

4. You can edit existing load points and add new ones. Please contact MÄK technical support for assistance.

7.12. 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 Section 8.9, "Writing Plans for Remote Entities", in *VR-Forces Scenario Management Guide*.

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. Aggregates and environmental process objects are not supported.
- VR-Forces components cannot control the remote entities. That is, they cannot alter its 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/default/extra/componentAttachmentTable.mtl*. The entries are essentially extensions to the configuration data in OPE files.

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")
  )
)
```

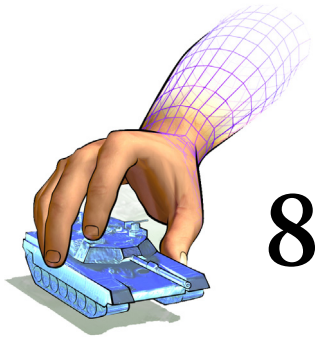
The entries in this table associate the key-names mig29-config and f18-config with appropriate entity type enumerations. You can also associate a key-name with marking text, as follows:

```
(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/default/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.



8

Using VR-Forces with an External Comms Effects Server

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

Introduction 8-2
Configuring VR-Forces 8-2
Enabling Radio Transmitter Publishing 8-3
Enabling the External Communication Model 8-3
Configuring the External Communications Model 8-4
Using VR-Forces with Qualnet 8-5
Running VR-Forces with Qualnet 8-5
Example Scenario 8-6
Running the Scenario 8-6

8.1. Introduction

You can configure VR-Forces to use an external communications effects server to control the transmission of radio messages between VR-Forces entities. The communications effects server determines which entities 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.

8.2. 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.
- Set the external communications model as the communications model to be used by VR-Forces.
- Configure the external communications model.

8.2.1. Enabling Radio Transmitter Publishing

The external communications model requires that any entities sending or receiving messages though the communications effects server publish a DIS or HLA radio transmitter object. By default, VR-Forces entities do not publish this object. You can enable publishing of transmitters for individual entities or for all entities. Enabling publishing for all entities is done through the GUI. For details, please see "Enabling the External Communication Model," on page 8-3.

To enable publishing of the radio transmitter object for an entity:

1. In the OPE file for each entity type for which you want to enable radio transmitter publishing, locate the **radios** block.
2. Change the **publish-transmitter** parameter to True for each radio, for example:

```
(radios
  (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)
    )
  )
)
```

8.2.2. 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 entities 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 Communications Model, edit the file *./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 entities 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.

8.2.3. Configuring the External Communications Model

If you enable the external communications 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 communications 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.

8.3. 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 8-2. 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.

8.3.1. 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 entities 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.

8.4. Example Scenario

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

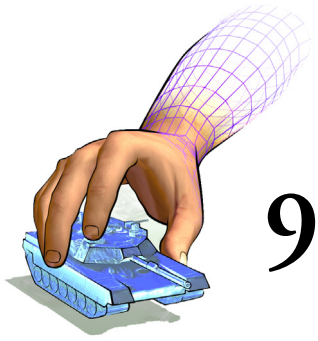
In the scenario, the M1A2 1 tank commands the UAV to fly over the area where the opposing forces are and send back spot reports. When the first tank receives spot reports, it commands the second one to move to the waypoint, and also moves there itself. All commands and spot reports are sent though the 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 entities start moving after 10 seconds of running, there is probably an error communicating with Qualnet.

8.4.1. 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/vrforces-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.



9

Display Engine Configurations

This chapter explains how to add and configure windows and channels and how to save display engine configurations.

The Display Engine 9-2
- Window Types 9-2
- Adding a Window 9-3
- Adding a Channel to a Window 9-5
- Removing a Window 9-6
- Removing a Channel 9-6
- Saving a Display Engine Configuration 9-6
- Loading a Display Engine Configuration 9-6

Changing a Window's Attributes 9-7

Changing a Channel's Attributes 9-8
- Setting the Clipping Planes 9-10
- Specifying the Projection Resize Policy Attribute 9-11
- Changing a Channel's Frustum (Field of View) 9-14
- Changing the Viewport 9-15

Configuring Multichannel Displays 9-16
- Changing the Camera's Position and Orientation Offset 9-19

Stereoscopic Displays 9-20
- Configuring Anaglyphic Stereo 9-21
- Configuring Polarized Stereo 9-22

9.1. 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.

9.1.1. 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 Section 8.3, "Inset Views", in *VR-Forces Users Guide*.)
- 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 VR-Forces 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.

9.1.2. Adding a Window

You can add as many windows as you want. New windows are named Window 2, Window 3, 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:

1. Choose **File** → **Stealth Window** (or **Plan View Window**). A new window opens (Figure 9-1).

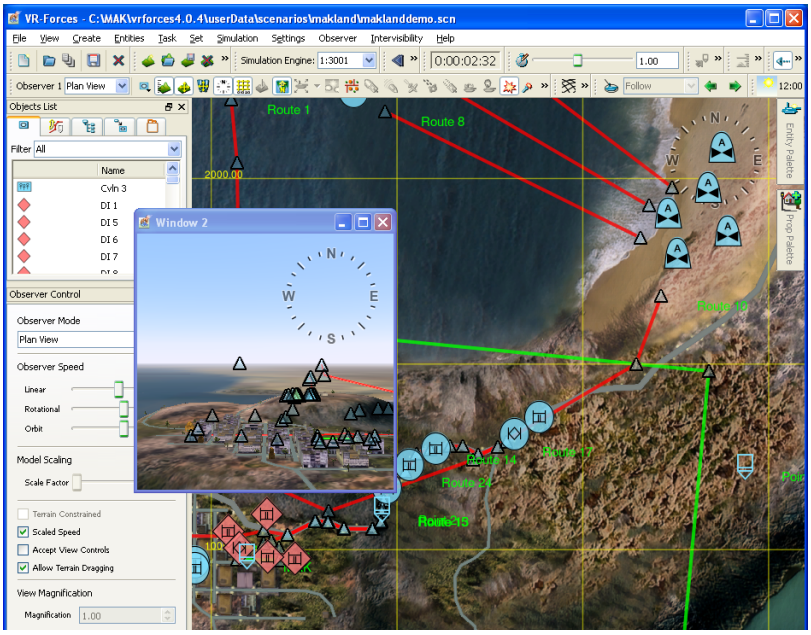


Figure 9-1. New window

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 9-2).

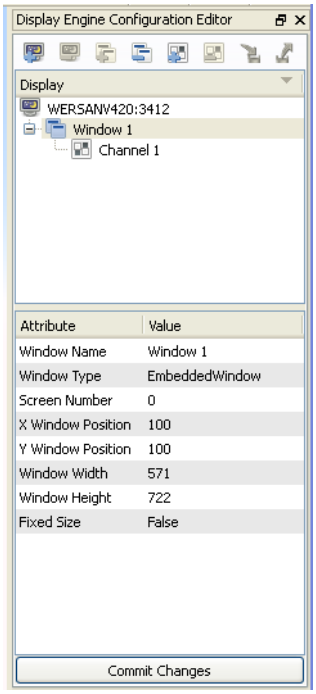


Figure 9-2. Display Engine Configuration Editor Panel

2. Select the display engine at the top of the Display tree.
3. Click the Add a Window icon (), or right-click the display engine name and choose **Add a Window** on the menu. A new window opens and is listed in the Display Engine Configuration Editor Panel (Figure 9-3).

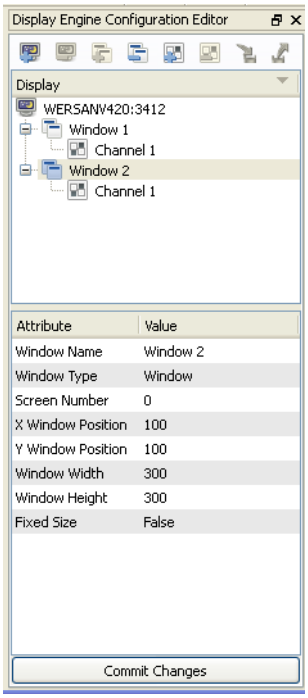


Figure 9-3. New window

9.1.3. 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 icon (), 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 9-10). For details about editing channel attributes, please see "Changing a Channel's Attributes," on page 9-8.

9.1.4. 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 icon (), or right-click the window name and choose **Remove Window** from the menu.

9.1.5. 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 icon (), or right-click the window name and choose **Remove Channel** from the menu.

9.1.6. 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 icon (), 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.

9.1.7. 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 icon (), 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.

9.2. 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 Section 9.1.1, "Window Types").
- 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 9-2).
2. Click the value of the attribute that you want to change.
3. To change a numeric value, type a new value or use the spin control to change the value. To change the Window Type or Fixed Size, select a value from the drop-down 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.

9.3. Changing a Channel's Attributes

You can change the following attributes of a channel:

- Channel Name.
- Observer Name.
- 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).
- Dynamic Near Far Planes.
- Parent Channel.
- Camera Position Offset (X, Y, Z).
- Camera Orientation Offset (Psi (heading), Theta (pitch), Phi (roll)).
- Viewport (Left, Right, Bottom, Top).
- Keywords – user-defined 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 9-4).

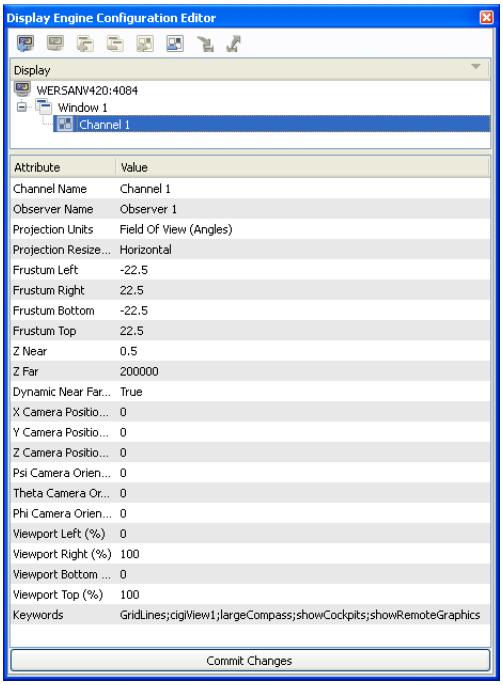


Figure 9-4. Channel attributes

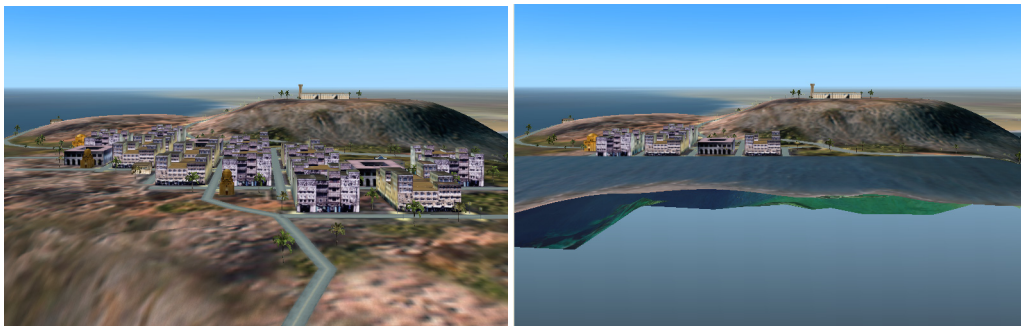
2. Click the value of the attribute that you want to change.
3. To change a numeric value, type a new value or use the spin control to change the value. To change the Observer Name, Project Units, Projections Resize Policy, or Parent Channel, select a value from the drop-down list. To change the Channel 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.

9.3.1. 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.

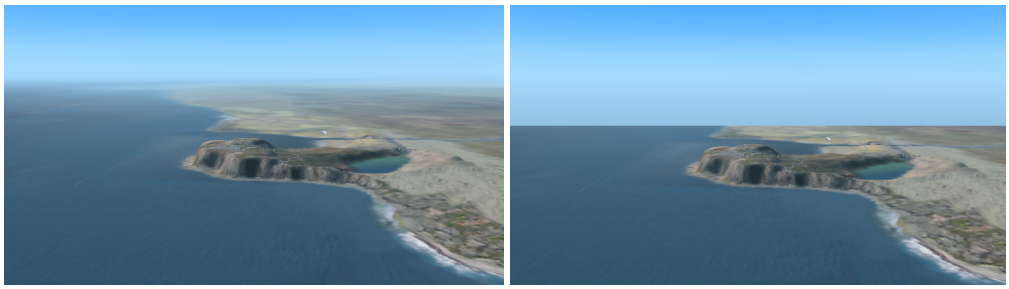
By default, VR-Forces uses dynamic clipping planes. The near and far clipping values are determined based on the altitude of the observer. You can disable dynamic clipping planes and set the clipping planes manually with the Z Near and Z Far attributes of a channel.

Figure 9-5 illustrates the effect of changing the Z Near value. Figure 9-6 illustrates the effect of changing the Z Far value.



Z Near = 1 Z Near = 500

Figure 9-5. Near clipping



Z Far = 10000000 Z Far = 10000

Figure 9-6. Far clipping

- To enable or disable use of dynamic clipping planes, set the Dynamic Near Far Planes attribute to True or False.
- To change the near clipping plane, edit a channel's Z Near attribute. To change the far clipping plane, edit the Z Far attribute.

9.3.2. 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 9-8 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 9-7).

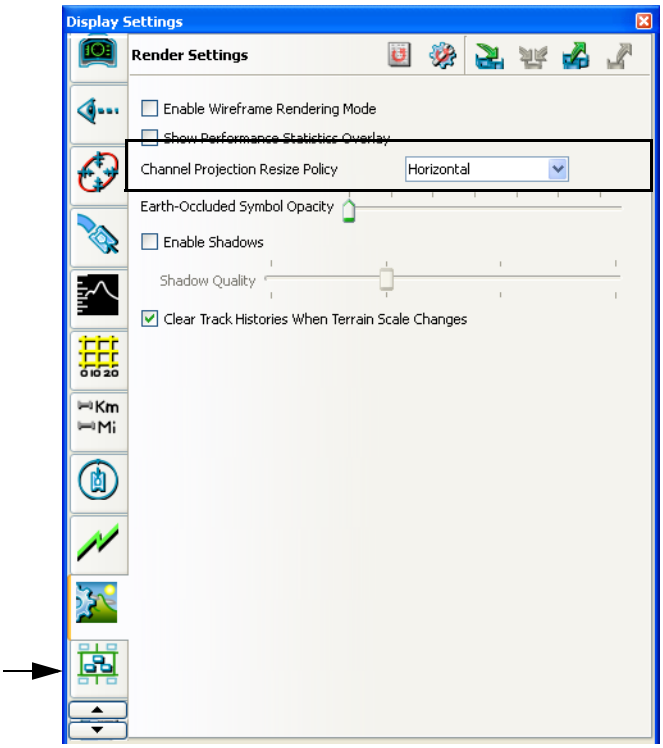
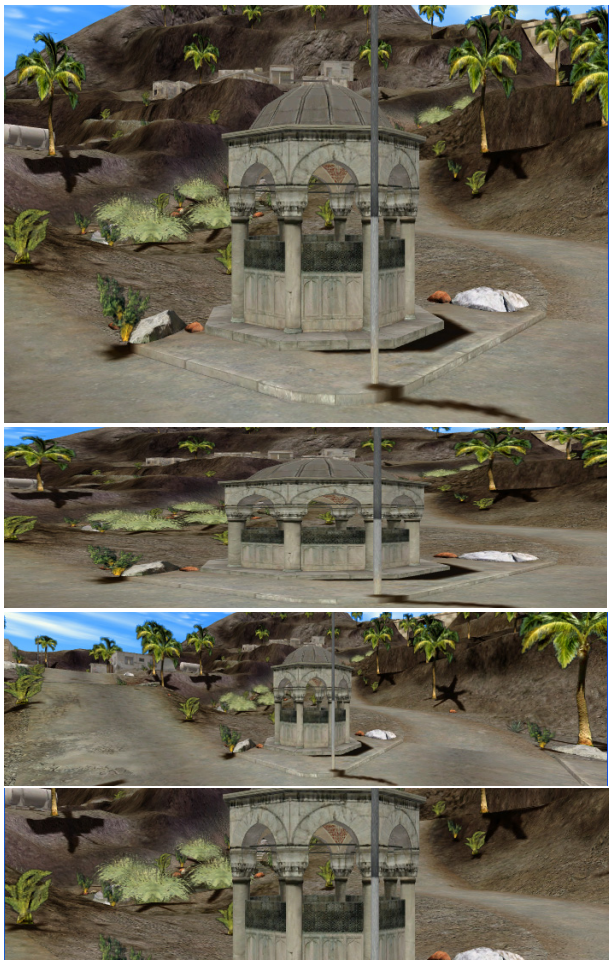


Figure 9-7. Display Settings, Render Settings page

3. Select an option on the Channel Projection Resize Policy drop-down list.



Initial view

Resized with Fixed resize policy

Resized with Horizontal resize policy

Resized with Vertical resize policy

Figure 9-8. Effect of different projection resize policies

9.3.3. 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 9-9 illustrates different field of view settings. The view on the left has frustum values -30, 30, -30, 30. The view on the right has the frustum values -10, 10, -10, 10.

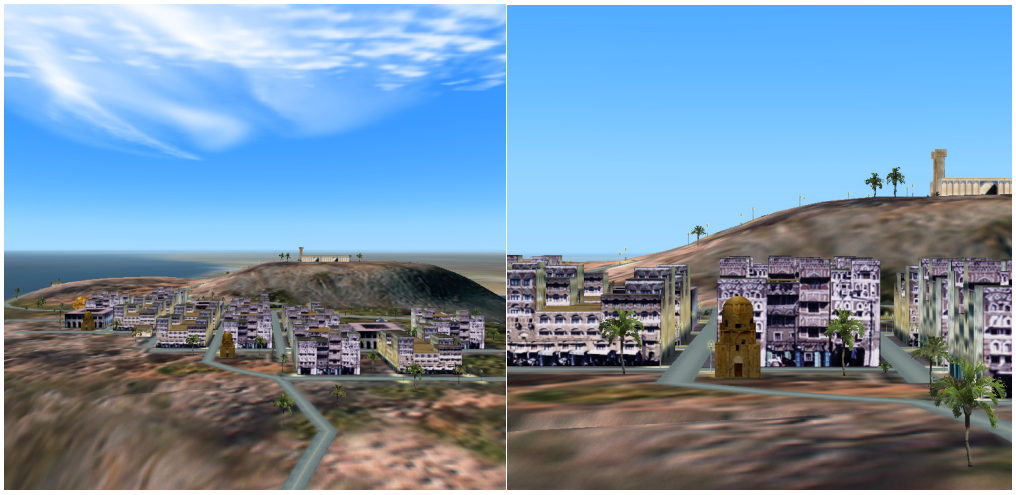


Figure 9-9. Field of view

➤ To change a channel's field of view, edit the frustum values in its attributes list.

9.3.4. 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 9-10 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.

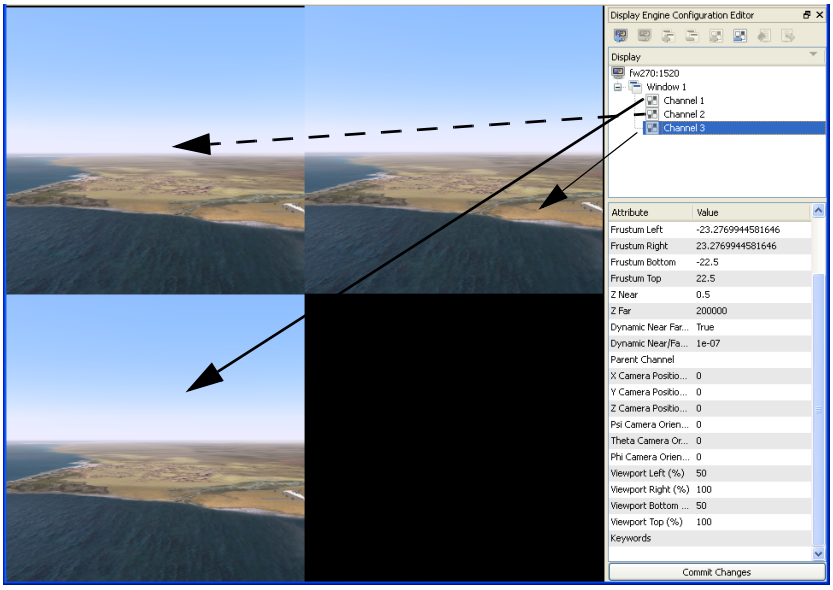


Figure 9-10. Window with three channels and three viewports

Viewports are specified as a percentage of the window from left to right and top to bottom. Figure 9-11 shows the percentage allocation for the viewports in (Figure 9-10).

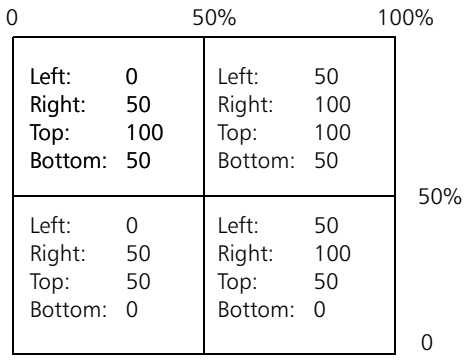


Figure 9-11. Viewport configuration



When you change viewports, the window might not completely refresh. To see the changes, resize the window.

9.4. 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 9-12).

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.

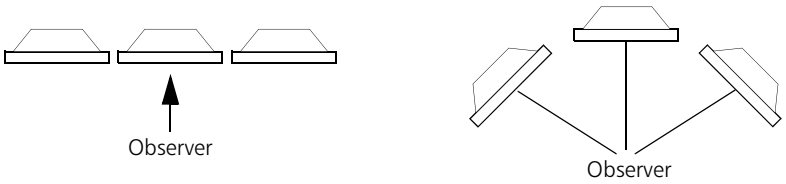


Figure 9-12. Multi-channel displays (top-down view)

By default, all windows and channels show the same view of the terrain (Figure 9-14, 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 9-13 illustrates the default frustum for Channel 1. It has a 60° field of view, from -30° to +30°.

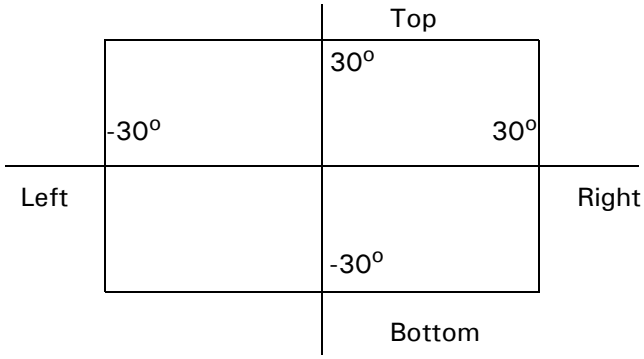


Figure 9-13. Default frustum for Channel 1

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 9-14. The field of view is the same: -30° to +30°. However, now each channel shows a 20° segment of the total field of view.

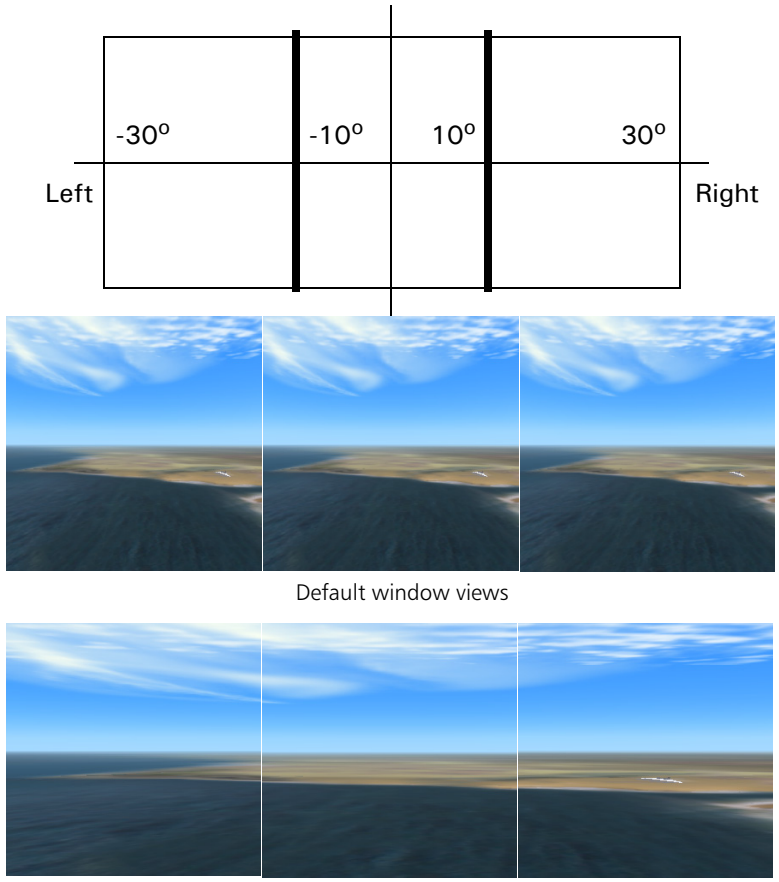


Figure 9-14. Frustum values for three channels

9.4.1. 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 9-15. (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).

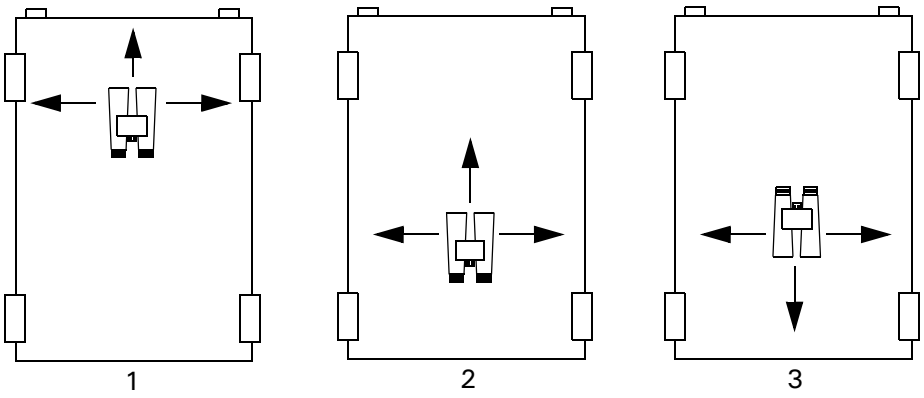


Figure 9-15. Position and orientation offsets

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).

9.5. 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.

9.5.1. 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 9-1.

Table 9-1: Environment variables for anaglyphic stereo

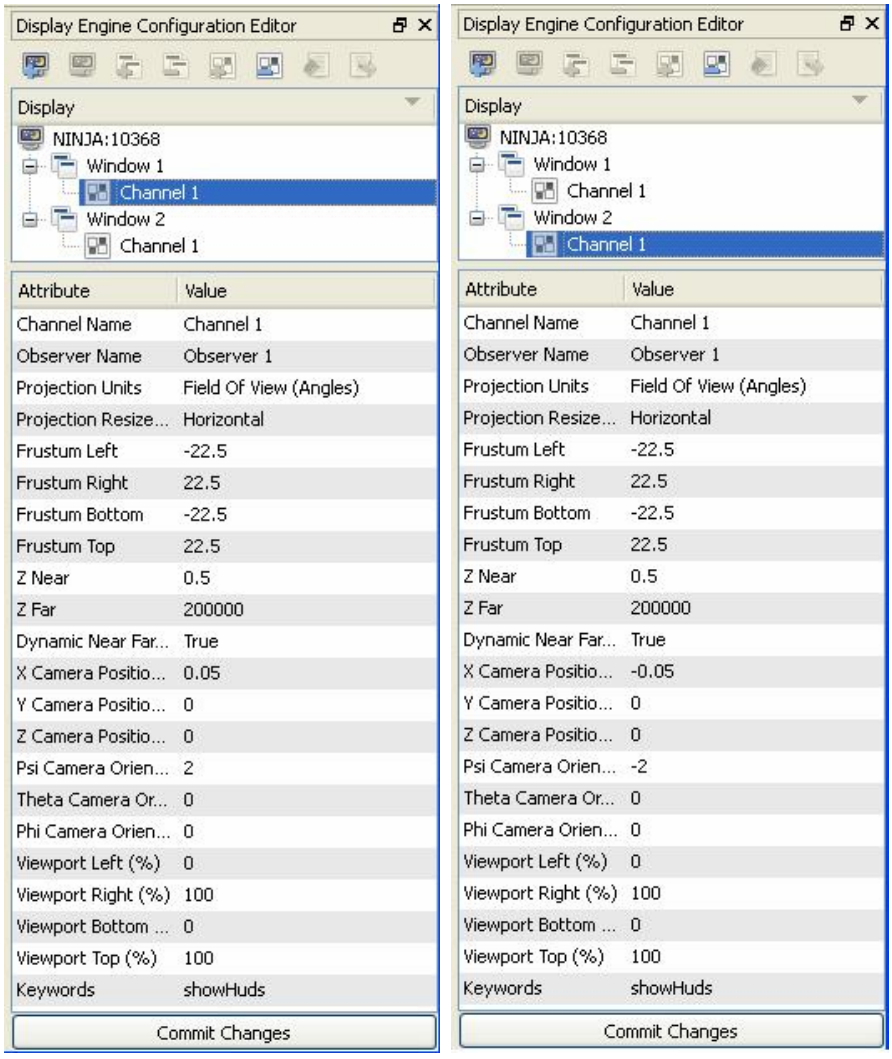
Environment Variable	Description	Default
OSG_STEREO	Turn stereo on.	ON
OSG_STEREO_MODE	Use anaglyphic stereo when in stereo.	ANAGLYPHIC
OSG_SCREEN_DISTANCE	Specifies the distance, in meters, that the viewer is from the screen.	0.50
OSG_SCREEN_HEIGHT	Specifies the height of the image, in meters, on the 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.

9.5.2. 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 9-16 shows an example configuration.

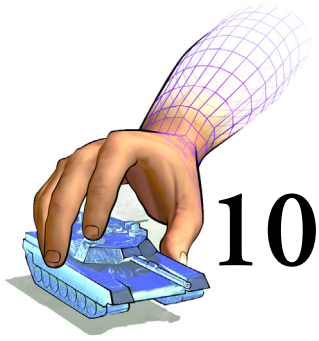


Left channel Right channel

Figure 9-16. 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.



10

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 "Terrain Databases," on page 3-29, in *VR-Forces Users Guide*. "Composing a Terrain," on page C-2 is a tutorial that steps you through the entire process of creating a terrain.

Creating a Composed Terrain 10-3
Considerations and Limitations for Building Terrains 10-3
Saving a Terrain 10-4
Adding Elevation Data (Terrain Patches) to a Terrain 10-5
Adding Images to a Terrain 10-6
Changing the Display Order of Raster Maps 10-9
Adding a Feature Layer 10-10
Connecting to Terrain Servers 10-11
Adding Terrain Server Connections 10-13
Editing a Terrain Server Configuration 10-15
Loading MetaFlight Terrains 10-15
Preprocessing Paged Terrains 10-15
Manually Paging-In Terrain 10-16
Adding Props to a Terrain 10-17
Extracting Props from a Terrain Patch 10-17
Adding Props from a Feature Layer 10-19
Viewing a List of Props 10-21
Selecting Props 10-21
Setting the Opacity of Props 10-22
Changing a Prop's Type 10-22
Changing a Prop's Position or Orientation 10-23
Changing a Prop's Model Definition 10-23

Configuring File Caching 10-24
Caching Terrain Server Data 10-24
Enabling Texture Compression 10-25
Clearing the File Cache 10-25
Displaying DDS Textures Correctly 10-25
Flipping DDS Textures Globally 10-26
Displaying Information About Paged and Streaming Terrains 10-27
Displaying Additional Terrain Paging Information 10-28
Configuring Terrain Paging for the Back-End 10-29
Building Efficient MetaFlight Terrains for VR-Forces 10-30

10.1. Creating a Composed Terrain

The general procedure for composing a terrain is as follows:

1. Choose **File** → **New Terrain**, or click the New Terrain icon () 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 10-5.
3. Optionally, add imagery, as described in "Adding Images to a Terrain," on page 10-6.
4. Optionally, add feature data, as described in "Adding a Feature Layer," on page 10-10.
5. Optionally, add props or extract props, as described in "Adding Props from a Feature Layer," on page 10-19 and "Extracting Props from a Terrain Patch," on page 10-17.
6. Save the terrain, as described in "Saving a Terrain," on page 10-4.

"Composing a Terrain," on page C-2 demonstrates how to create a composed terrain.

10.1.1. 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, use the MÄK Terrain Database Tool (TDB Tool) to create a GDB terrain composed of the DTED cells. Then load the GDB terrain into VR-Forces.
- The larger a terrain database is (as measured by the number of polygons), the more it will affect performance.

Loading DTED

By default, when you load DTED files as terrain patches, the files are loaded with their altitude referenced to WGS84. If you want the altitude to reference a different vertical datum (most likely EGM96), you must provide a geoid grid file for that vertical datum. If you use a geoid grid file, the altitudes in the DTED file are adjusted based on the information in the grid file.

To use a geoid grid file, set the environment variable MAK_GEOID_GRID to the path of the file.

For information about EGM96 and geoid file, please see the following URL:

http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm96/egm96.html



This section does not apply to terrains loaded using an Earth file.

10.1.2. 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. 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 10-24.) However, 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. 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. If you need to process the terrain in the TDB Tool, you can save it as GDB.

To save a terrain:

1. Choose **File** → **Save Terrain**, or click the Save Terrain icon () on the Terrain Toolbar. The Save Terrain dialog box opens.
2. Type a name for the terrain.
3. Click Save.

10.2. 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 10-3.

! 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 icon ( ) on the Terrain Toolbar.The Terrain Settings dialog box opens.
2. Select the 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 10-1).

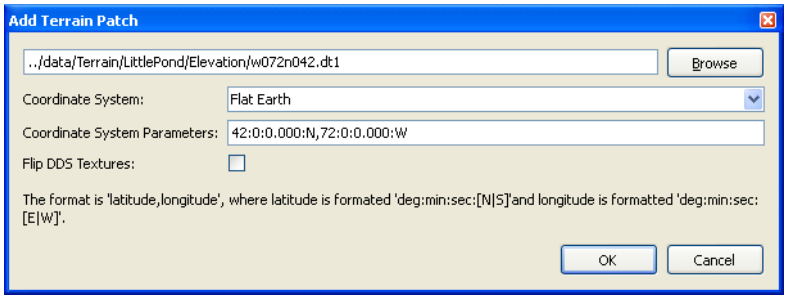


Figure 10-1. Add Terrain Patch dialog box

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.
7. Click OK. The terrain patch is added.
8. Optionally, add additional terrain patches.
9. Save the terrain.

10.3. 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.

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 icon (+) that is above the list of raster maps. A line is added to the Raster Maps list (Figure 10-2).

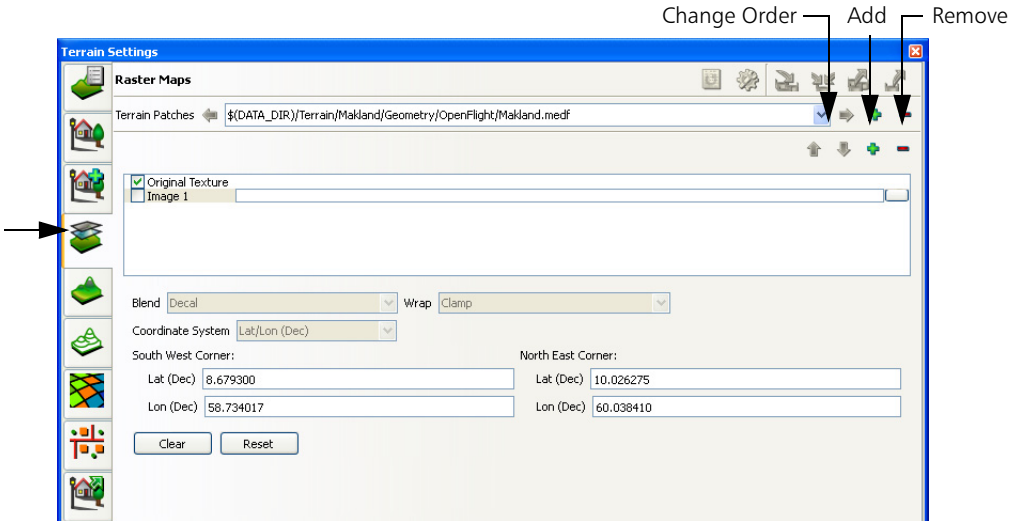


Figure 10-2. Raster Maps page

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 window, select the check box to the left of the entry (Figure 10-3).

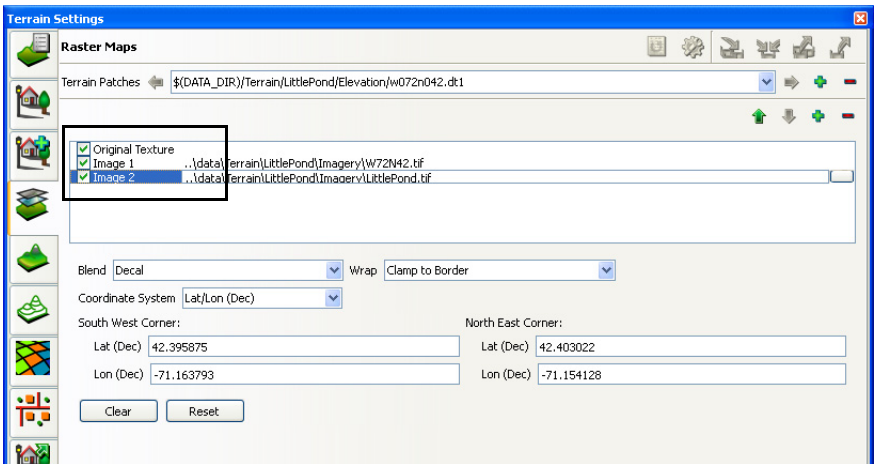
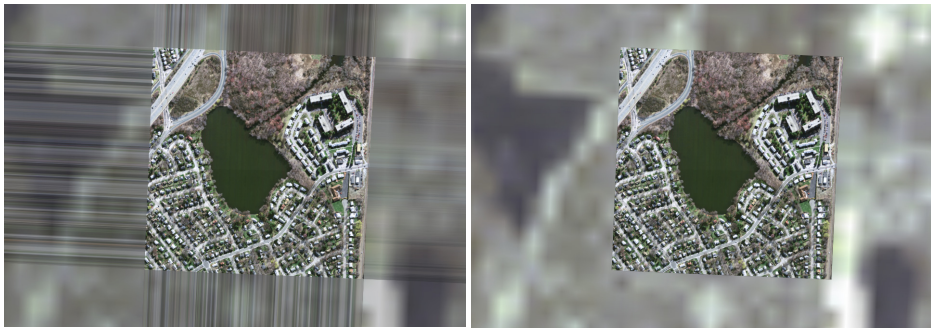


Figure 10-3. Making an image visible

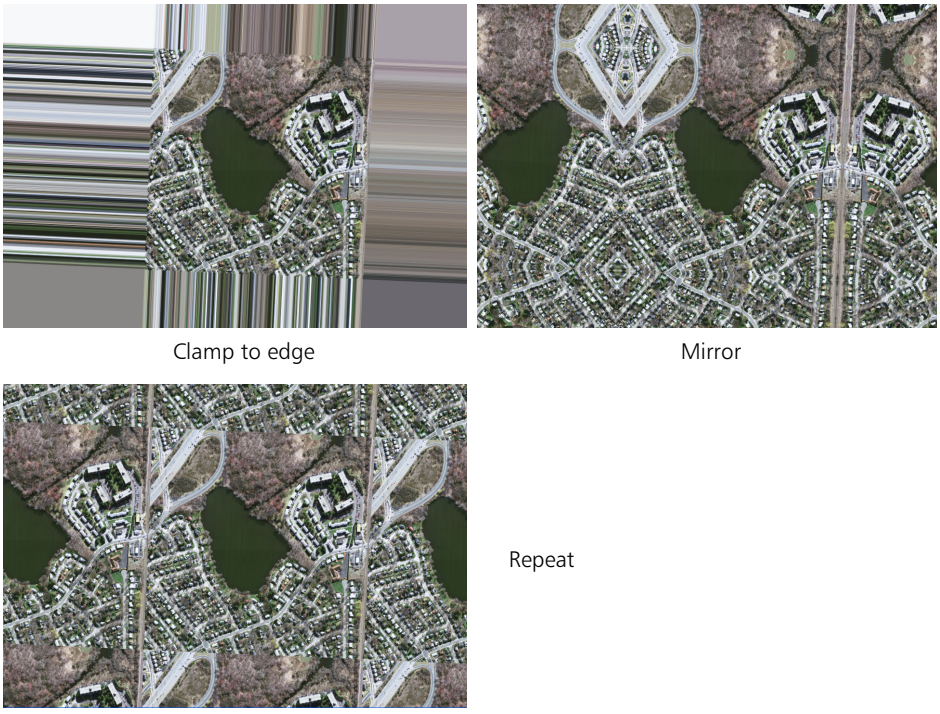
7. Optionally, specify the attributes of the image, as follows:
   - Blend specifies how the image interacts with underlying images, as follows:
     - Decal: covers the polygons with the image. Use it to apply an opaque texture or a texture with transparency.
     - Modulate: color values of the image 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 10-4 and Figure 10-5):
     - 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 10-4. Raster image wrap effects (Clamp and Clamp to Border)



Clamp to edge | Mirror

Repeat

Figure 10-5. Raster image wrap effects (Clamp to Edge, Mirror, Repeat)

10.3.1. 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 10-6 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 10-6. 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 10-2).
3. Select the raster image whose order you want to change.
4. Click the Up Arrow or Down Arrow to the right of the Raster Maps label. The image moves in the list and the window changes as appropriate.

10.4. Adding a Feature Layer

A feature layer is a file that contains source data from which you can add props to a terrain. A feature layer can be a MÄK Terrain Format (GDB) file, a shapefile (SHP), or DFAD data (DFD).

- Adding a feature layer does not, by itself, make any visible difference to a terrain. To take advantage of 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 10-17.
- 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.

To add a feature layer:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Add Props page (Figure 10-7).

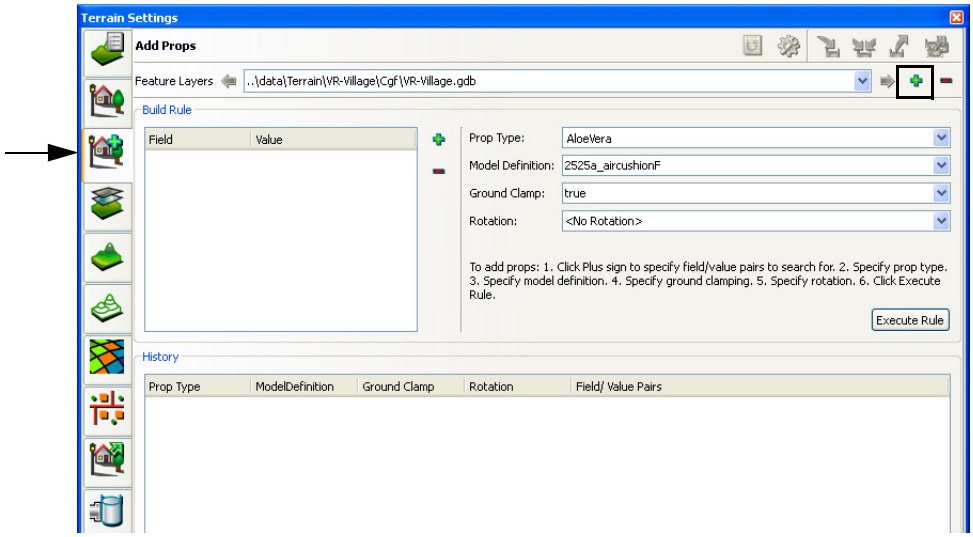


Figure 10-7. Add Props page

3. Click the Add icon (+) to the right of the Feature Layers list. 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 feature layers.

10.5. Connecting to Terrain Servers

VR-Forces supports the osgEarth libraries for creating terrains from remote or local data sources (Figure 10-8). Terrain servers support streaming of terrain data and are, therefore, ideal for displaying terrains that have large amounts of data.

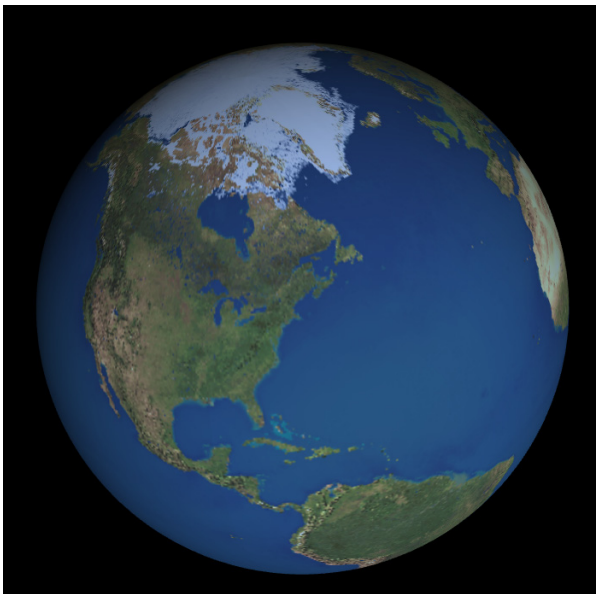


Figure 10-8. Geocentric terrain

You can manipulate the terrain from the keyboard and mouse and can save it as an MTF terrain for quick loading.

VR-Forces includes several terrain server configurations for servers that are accessible over the internet, including a server hosted by VT MÄK (*VR-TheWorld.earth*, which connects to VR-TheWorld Online). 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.

"Adding a Terrain Server," on page C-13, 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.

! 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.

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 10-9).

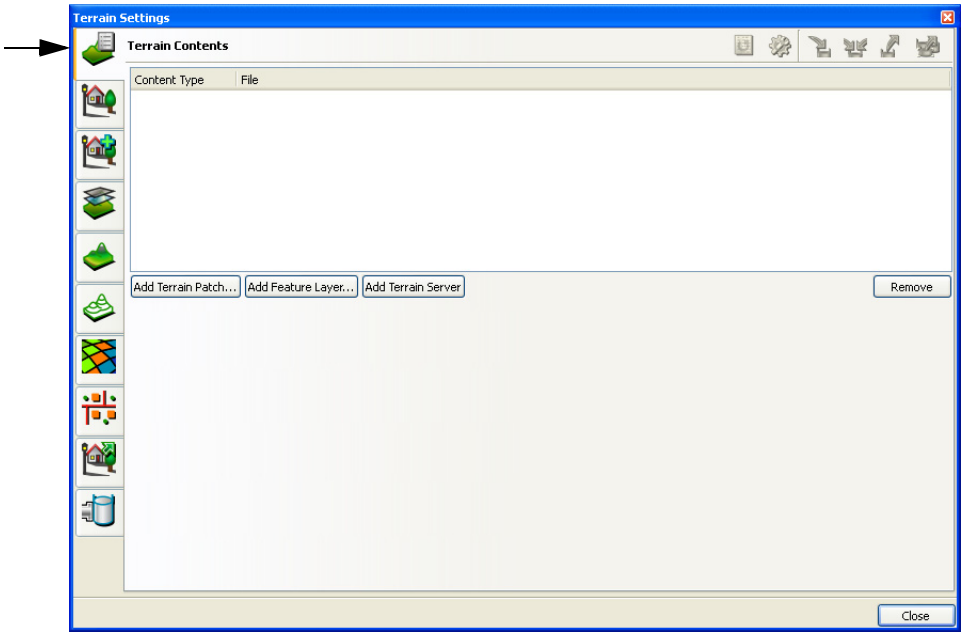


Figure 10-9. Terrain Contents

4. Click Add Terrain Server. The Add Terrain Server dialog box opens (Figure 10-10). The dialog box contains information about the selected terrain server. It is not editable. (For information about editing terrain server configurations, please see "Editing a Terrain Server Configuration," on page 10-15.)

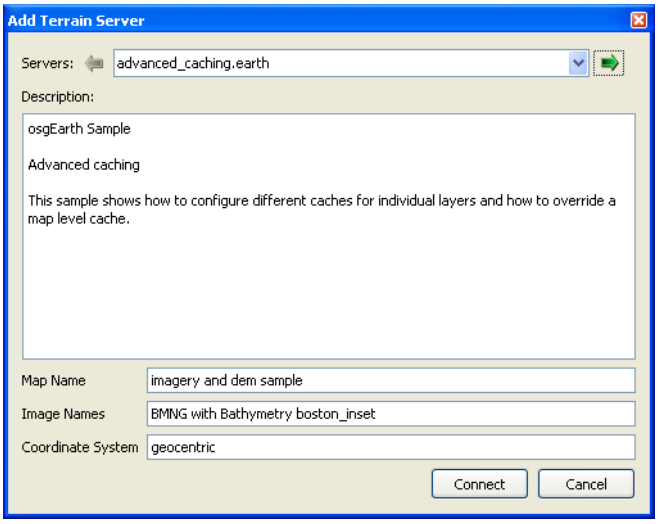


Figure 10-10. Add Terrain Server dialog box

5. Select a server from the Servers drop-down list.
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 or scene. For details, please see "Saving a Terrain," on page 10-4.

10.5.1. 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. (Appendix H, *Editing Earth Files*, provides a brief introduction to Earth file syntax.)

"Adding a Terrain Server," on page C-13, is a tutorial that shows how to create a terrain server that serves local data.

To add a terrain server connection:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Server Settings page (Figure 10-11). The dialog box displays the contents of the configuration file for the selected server.

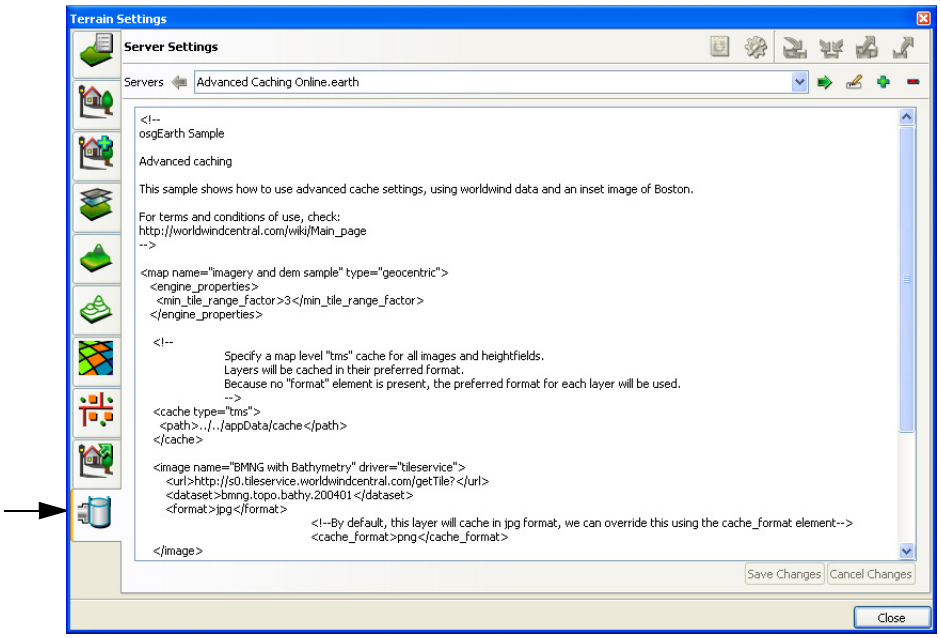


Figure 10-11. Server settings page

3. Select a terrain server configuration that is similar to the one that you want to add.
4. Click the Add icon (). A new configuration file is added to the list. It copies the contents of the previously selected configuration and names the server in the format *oldServerName*_1.
5. Make whatever changes are necessary to configure your new server. This would usually mean changing the URLs that point to the data. A URL can be an internet URL, a path to a directory, or a path to a specific file.
6. Optionally, click the Rename icon () and rename the server.
7. Click Save Changes. New server files are saved to *./userData/servers*.

10.5.2. Editing a Terrain Server Configuration

VR-Forces provides a text editor interface that lets you edit the osgEarth terrain server configuration files. You can also edit them in any text editor. You are responsible for ensuring proper syntax.

To edit a terrain server:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Server Settings page (Figure 10-11). The dialog box displays the contents of the configuration file for the selected server.
3. In the Servers list, select the terrain server configuration that you want to edit.
4. In the text window, edit the file.
5. Click Save Changes.

10.6. 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 first process it using the mftTool and medfTool. For details, please see Appendix G, *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 10-24.

10.7. Preprocessing Paged Terrains

VR-Forces supports preprocessing external references used by paged terrains. VR-Forces does not require terrain formats such as TerraPage or pagable 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 will create a file called */externals/tree.medf*. When VR-Forces loads the terrain, it will load the MEDF version instead of the OpenFlight version.

We recommend that you preprocess all data to minimize load times and maximize run-time performance.

10.8. Manually Paging-In Terrain

VR-Forces pages in terrain based on the location of simulated 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 entities over a wide area or many fast-moving entities. 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 global or entity 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 an entity to ensure that terrain in front of the entity gets paged in.

Terrain page-in areas are supported for paged IVE files and supported versions of the MetaFlight format.

To create a terrain page-in area:

1. Choose **Create** → **Control Object** → *force* → **Terrain Page-in Area**.
2. Create the area as described in Section 2.1, "Creating Objects", in *VR-Forces Scenario Management Guide*.

10.9. 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 Create menu and 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.

10.9.1. 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 *Bridge_Arch_Stone.flt* file 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 C-17.

To extract props from a terrain:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Extract Props page (Figure 10-12).

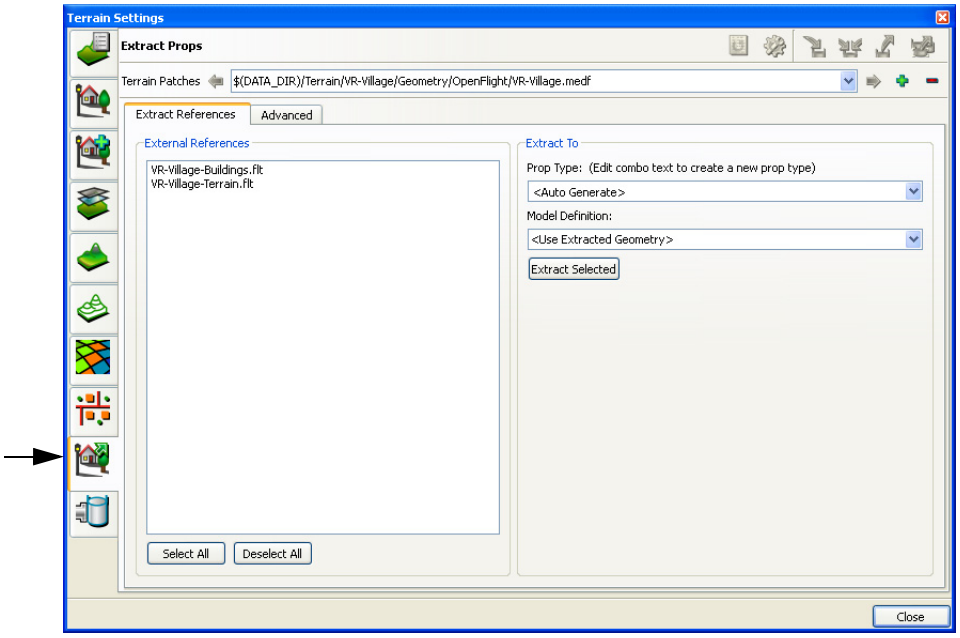


Figure 10-12. Extract Props page

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 Terrain Contents dialog box.

10.9.2. 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 "Composing a Terrain," on page C-2 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 10-13).

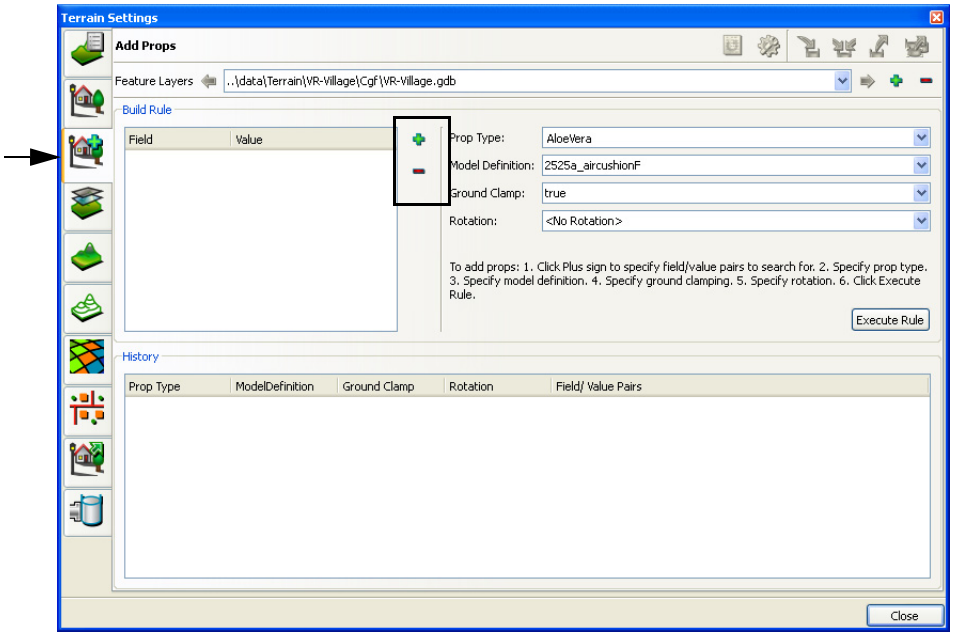


Figure 10-13. Terrain Settings, Add Props page

3. In the Feature Layers list, select the feature layer from which you want to import the props.
4. Click the Add icon (+) to the right of the Field/Value window (Figure 10-13). The Build Field/Value Pair dialog box opens (Figure 10-14).

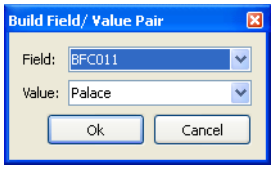


Figure 10-14. Build Field/Value Pair dialog box

5. In the Field drop-down list, select a field. Fields are the column headings in the feature definition.
6. In the Value drop-down 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 10-15).

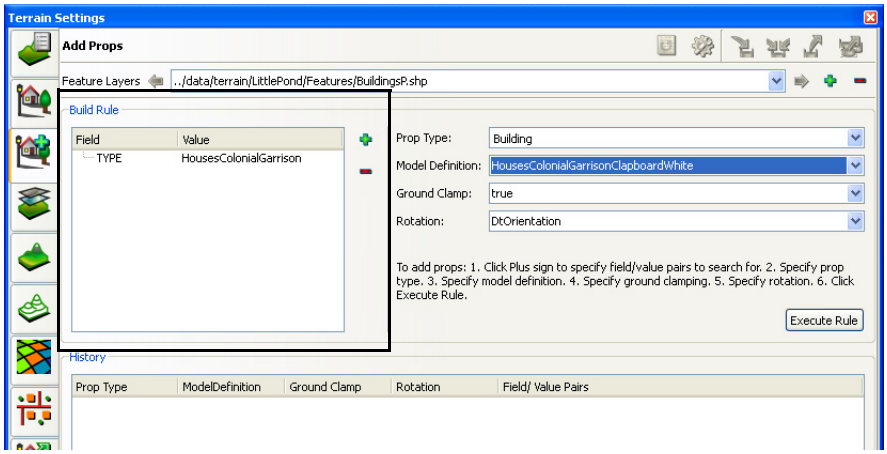


Figure 10-15. New Field/Value pair

8. Add additional Field/Value pairs as needed to define the prop that you want to add.
9. In the Prop Type drop-down list, select the prop type to assign to the prop you want to add.
10. In the Model Definition drop-down 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.

10.9.3. 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 10-16).

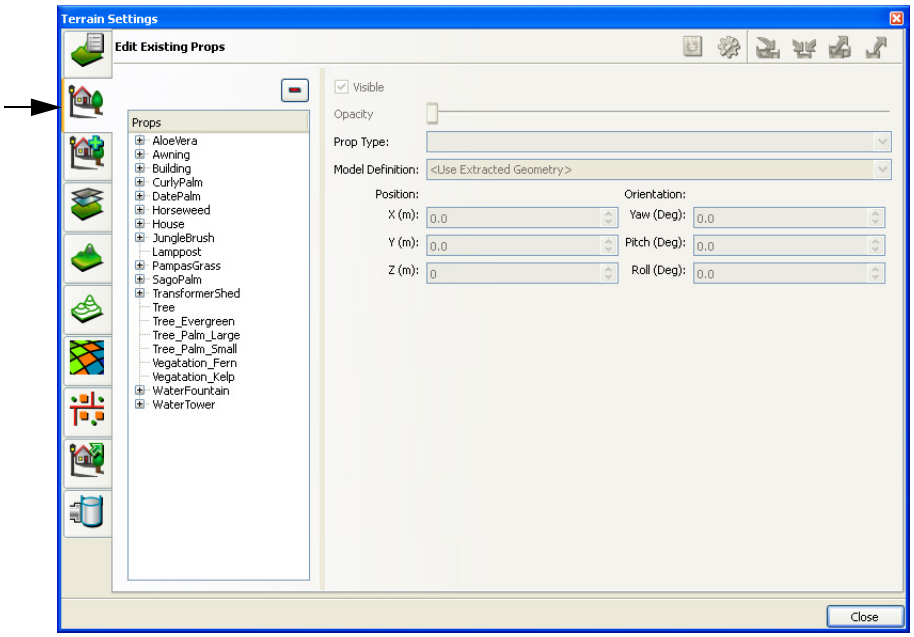


Figure 10-16. Edit Existing Props page

3. Expand the list to view the entries under a particular type of prop.

10.9.4. 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.

10.9.5. Setting the Opacity of Props

To set a prop's opacity:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 10-16).
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**.

10.9.6. 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 10-16).
3. In the list of props, select the props whose type you want to change.
4. Select a prop type from the Prop Type drop-down list, or type a new type name. The selected props are moved to the new prop type in the Props window.

10.9.7. 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 10-16).
3. In the list of props, select the prop whose position or orientation you want to change.
4. To change position, type new values in the X, Y, or Z boxes, or use the spinner controls to change the position values.
5. To change the orientation, type new values in the Yaw, Pitch, or Roll boxes, or use the spinner controls to change the orientation values.

10.9.8. Changing a Prop's Model Definition

You can change the model definition used to render a prop in the scene. For information about model definitions, please see Chapter 12, *Model and Element Definitions*. For an example of changing a prop's model definition, please see "Test the Model Mapping," on page D-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 10-16).
3. In the list of props, select the prop whose model definition you want to change.
4. In the Model Definition drop-down list, select a model definition. The selected props are rendered using the new model definition.

10.10. 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.

To enable or disable file caching:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 10-17).

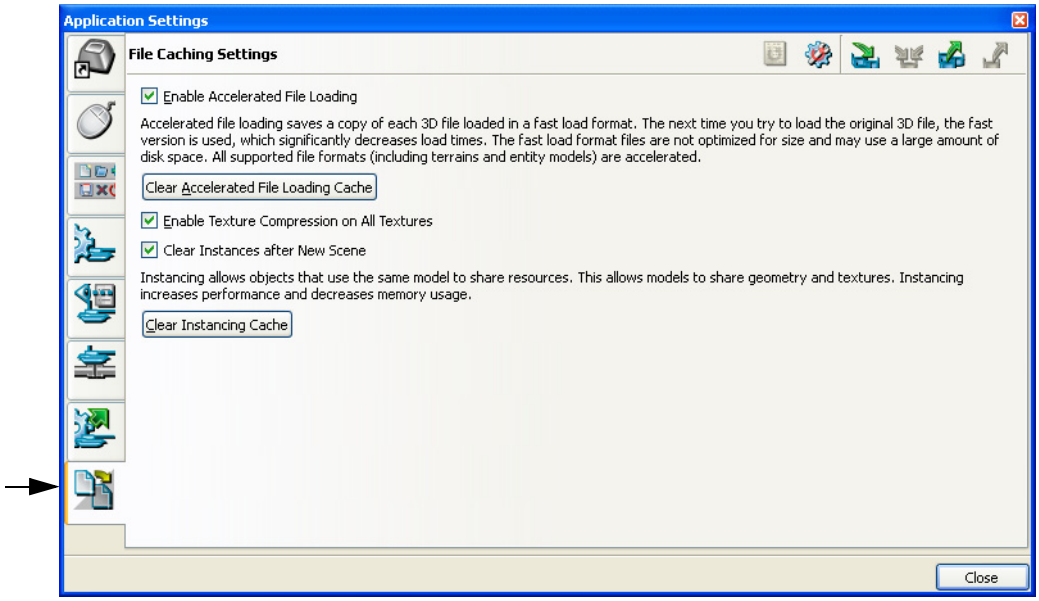


Figure 10-17. File Caching Settings page

3. Select or clear the Enable Accelerated File Loading check box.

10.10.1. 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 <cache> element in the *.earth* file or in the OSGEARTH_CACHE_PATH environment variable.

10.10.2. 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 10-17).
3. Select or clear the Enable Texture Compression on all Texture check box.

10.10.3. 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 10-17).
3. Click the Clear Accelerated File Loading Cache button.

10.11. 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 cornet. 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 12-31.)
- When you add a terrain patch. (For details, please see "Adding Elevation Data (Terrain Patches) to a Terrain," on page 10-5.)
- As a global setting. (For details, please see "Flipping DDS Textures Globally," on page 10-26.)

10.11.1. Flipping DDS Textures Globally

You can specify the DDS textures setting to 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 10-18).

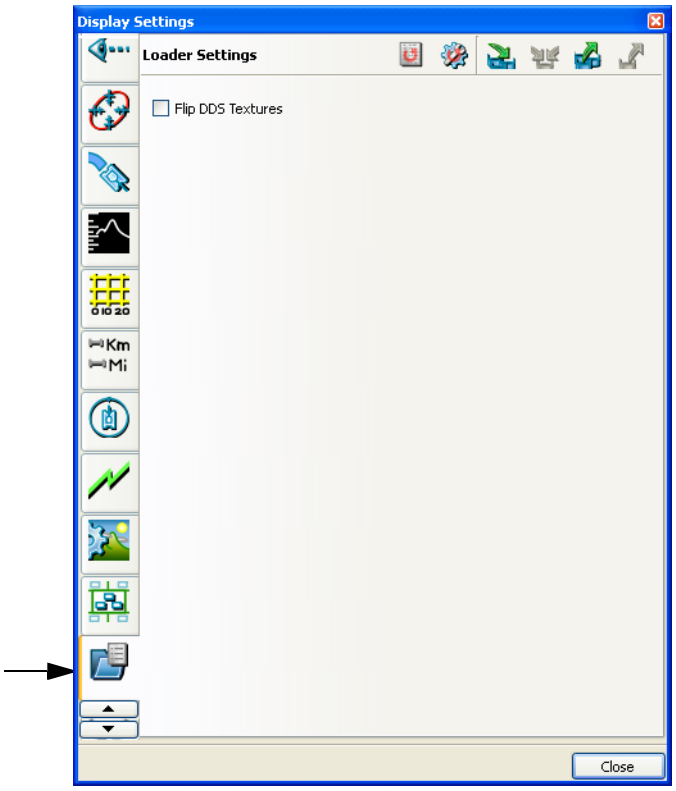


Figure 10-18. Loader Settings page

3. Select the Flip DDS Textures check box.
4. Click Close.

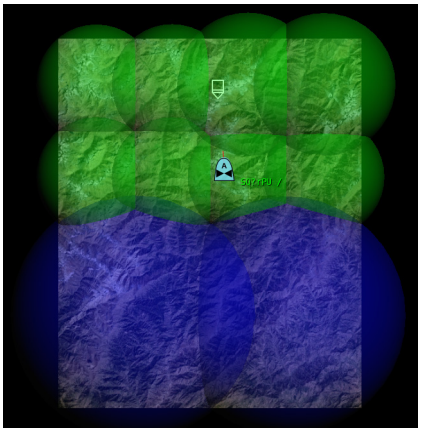
10.12. Displaying Information About Paged and Streaming Terrains

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 as tiles for streaming terrain, and as bubbles for paged terrains. Figure 10-19 shows the paged-in terrain for a paged terrain (*HarrisAghanVillageFlatEarthMetaFlight*) and a streamed terrain (VR-TheWorld). In the paged terrain, green spheres represent the area that is paged in. Blue spheres represent area that is not paged in. In the streamed terrain, yellow squares show the tiles that have been paged in as an entity moves over the terrain.



Drawing of paged terrain tiles is done using shared memory. Therefore, you can draw them only if the front-end and back-end are running on the same computer.

You can display different information by entering commands on the back-end console.



HarrisAghanVillageFlatEarthMetaFlight.mtf



VR-TheWorld.mtf

Figure 10-19. Paged-in terrain

➤ To enable or disable display of terrain paging information, choose **Settings** → **Simulation Terrain Paging Information**.

10.12.1. Displaying Additional Terrain Paging Information

You can display the following terrain paging information by entering commands on the back-end console. Table 10-1 lists the commands and their results.

Table 10-1: Terrain paging information commands

Command	Result
`terrain drawmode` *mode*	Specifies a bitmask that controls what information to display, as follows: ◆ 0 — Turn off information display. ◆ 1 — Display polygons (Figure 10-20). ◆ 2 — Display collision object bounding boxes (paged terrains) (Figure 10-21). ◆ 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*.
`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.

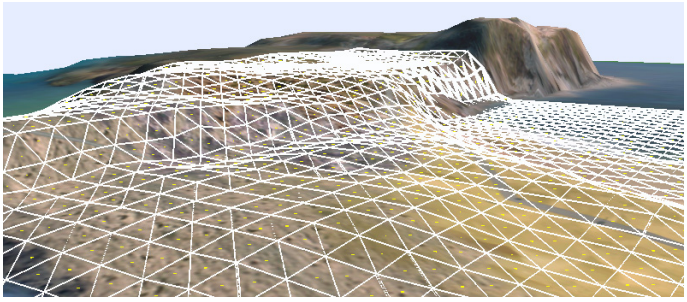


Figure 10-20. Polygon display

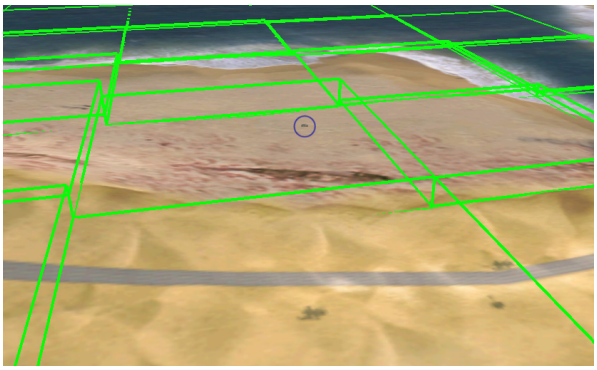


Figure 10-21. Collision objects display

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 will remember what you typed.
3. Press Enter.

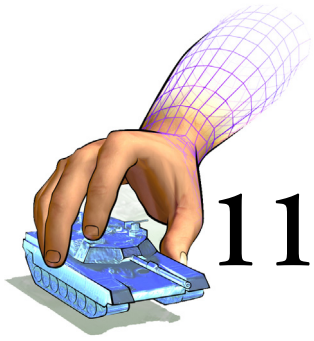
10.12.2. Configuring Terrain Paging for the Back-End

You can configure terrain paging for the back-end in the *./appData/settings/vrfSim/terrainOptions.mtl* configuration file. 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 Section 3.11.5, "Using Large Terrain Databases", in *VR-Forces Users Guide*. 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 10-15.)

10.13. Building Efficient MetaFlight Terrains for VR-Forces

VR-Forces can only load MetaFlight terrains built with certain tools. All MetaFlight files must be pre-processed 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.)



11

Mapping Models and Effects

This chapter explains how to map models to DIS and HLA entity types. Appendix D, *Model Tutorials* walks you through the process of adding a model.

Mapping Models to Entities and other Scene Elements 11-2
- Adding a Model Mapping 11-3
- Editing a Model Mapping 11-4
- Filtering the Element Definition List 11-5
- Deleting a Model Mapping 11-5
- How VR-Forces Maps DI-Guy Models 11-6

Clearing the Model Instancing Cache 11-6

Compressing Model Files 11-7

11.1. Mapping Models to Entities and other Scene Elements

Model mapping is the process of associating an entity or other scene element with an element definition.

DIS and HLA objects are specified by an entity type enumeration. An entity type enumeration is a list of seven components as specified by the DIS protocol and HLA RPR FOM. If none of the seven components contains a wildcard (-1) value, then the entity type refers to a specific 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. For complete details about entity type enumerations, please see *Enumeration and Bit Encoded Values for Use with Protocols for DIS Applications* (DIS enumerations document). It is available at the IEEE World Wide Web site: http://shop.ieee.org/store/ and at the SISO web site: http://www.sisostds.org/.

When a VR-Forces application discovers an entity, it reads the enumeration and then displays the entity using the element definition that is mapped to the enumeration. (Element definition is a generic term that includes entity definitions, tactical graphic definitions, and other scene elements.)

You map models in the Entity Type Mappings dialog box. It has pages for mapping:

- Entity models.
- Aggregate entity models.
- Firing effects.
- Detonation effects.
- Tactical graphics.
- Cockpit displays.

The mapping process is the same for all model types. You can add new model mappings, edit existing model mappings, and delete mappings.



- You can map more than one element definition to an entity 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.
- The procedures in this section are for entities (and hence refer to entity definitions). However, the procedures are the same for all of the pages on the Entity Type Mappings dialog box.

11.1.1. Adding a Model Mapping

This procedure describes how to map an element definition to an entity type enumeration. The element definition must already exist. The procedure is the same for any page on the Entity Type Mappings dialog box.

To add a model mapping:

1. Choose **Settings** → **Entity Type Mappings**. The Entity Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 11-1). The page shows a list of entity type enumerations in the Entity Type column. The entity definition mapped to the entity type is listed in the Entity Definition column. By default, the list of entity types is filtered to just show entities.

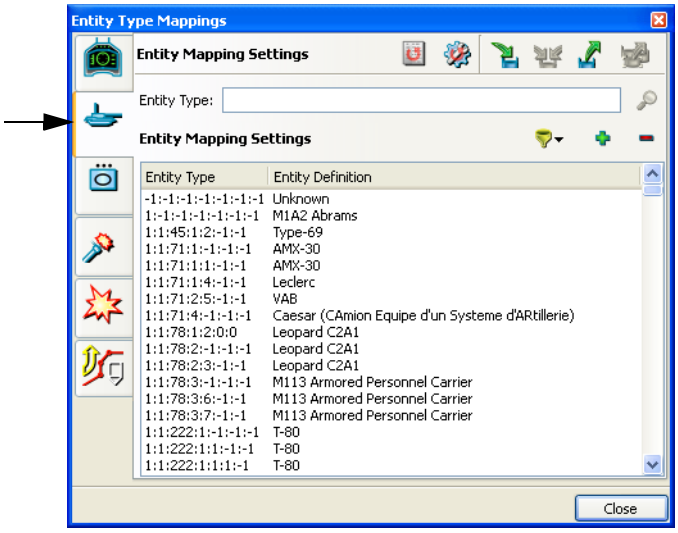


Figure 11-1. Entity Mapping Settings page

3. Click the Add icon (+). The Create New Entity Type Model Mapping dialog box opens (Figure 11-2). It lists a default entity type enumeration.

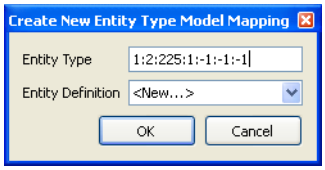


Figure 11-2. Create New Entity Type Model Mapping dialog box

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 drop-down list, select the element definition that you want to map to this entity type.
6. Click OK.
7. The new model mapping is added to the list.

11.1.2. Editing a Model Mapping

This procedure describes how to edit a model mapping for an entity. The procedure is the same for any page on the Entity Type Mappings dialog box.

To edit a model mapping:

1. Choose **Settings** → **Entity Type Mappings**. The Entity Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 11-1).
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 icon (🔎). 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 Element Definition List," on page 11-5.)
   b. In the Entity Definition column, click the element definition to make it an editable list (Figure 11-3)).
   c. Select the new element definition from the drop-down list.
   d. Click someplace else in the window.

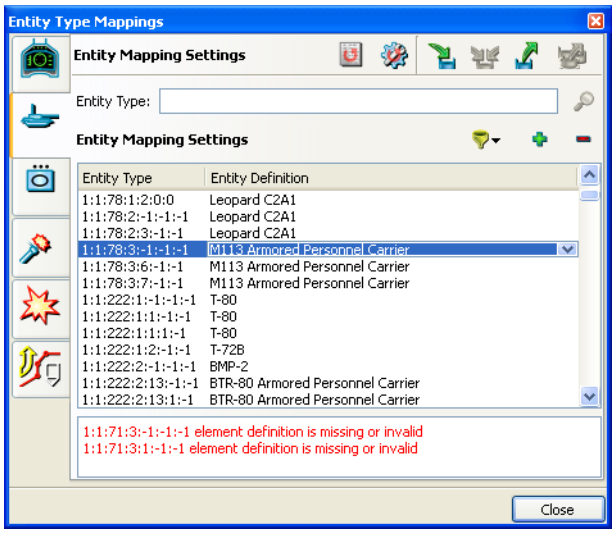


Figure 11-3. Editable entity definition

11.1.3. Filtering the Element Definition List

You can filter the list of element definitions that are available in the drop-down list for mapping to entity types.

To filter the list of element definitions:

1. On any page of the Entity Type Mappings dialog box, click the Filter icon ( ). 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.

11.1.4. Deleting a Model Mapping

The procedure for deleting a model mapping is the same for any page on the Entity Type Mappings dialog box.

To delete a model mapping:

1. Choose **Settings** → **Entity Type Mappings**. The Entity Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 11-1).
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 icon ( ). VR-Forces highlights the entry for that enumeration if it is present.

4. Click the Delete icon (-).

11.1.5. 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 will no longer try to choose animations and will expect to be told how to render that lifeform for the remainder of the simulation.

11.2. 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. You can clear it automatically whenever you create a new scene, or you can clear it manually.

To clear the model instancing cache after each scene change:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 11-4).

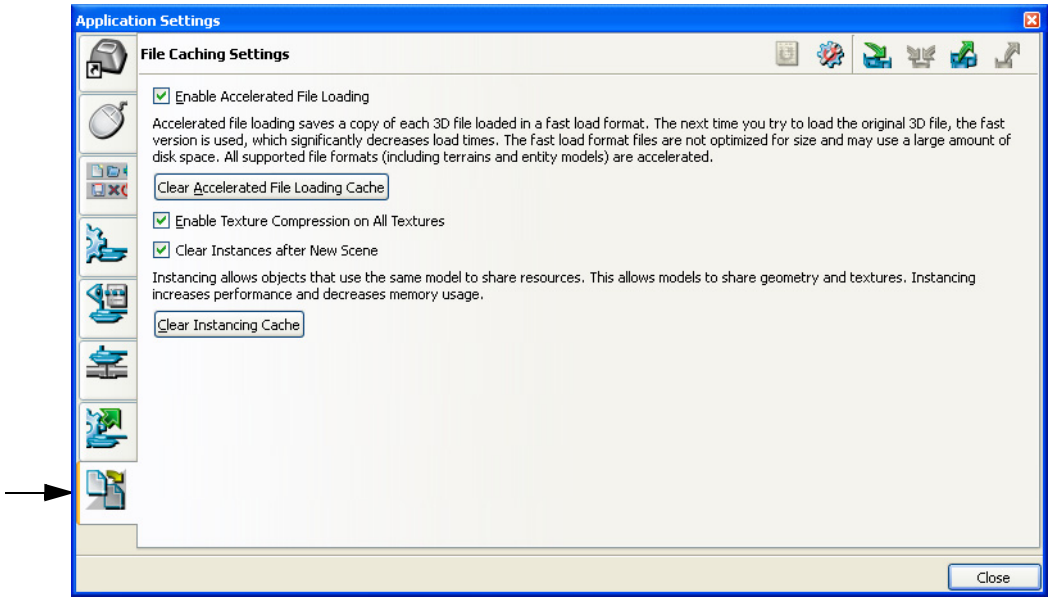


Figure 11-4. File Caching Settings page

3. Select the Clear Instances After New Scene check box.

To clear the model instancing cache manually:

1. Choose **Settings** → **Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 11-4).
3. Click the Clear Instancing Cache button.

11.3. 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 *./bin/medfTool.exe*. The compression tool supports the 3D modeling formats supported by VR-Forces, such as OpenFlight, 3DS, OBJ, and raster formats such as RGB and PNG. 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 11-1 describes the arguments.

Table 11-1: 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 11-1: medfTool arguments

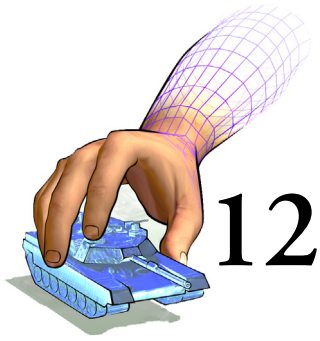
Argument	Description
(-p | --threads) *threads*	Specifies the number of threads to run. Specifying zero results in the number of threads equaling the number of logical processors the host machine has.
(-q | --forceOpaque) *simple_filename*	Specifies a filename and extension, but no path, to force polygons 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 will be 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 which 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*:

```
./bin/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.

```
./bin/medfTool.exe
  --directory %MAK_DATADIR%/Lifeforms --norecurse
  --extensions flt --image_extensions rgb
  --image_extensions rgba --image_extensions png
  --image_extensions dds
```



12

Model and Element Definitions

In Section 4.4, "Introduction to VR-Forces Object Modeling", in *VR-Forces Users Guide*, we describe the interplay of schemas, model definitions, entity 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 D-2 walks you through the process of adding a model.

Creating and Editing Schemas 12-3
Creating Schemas 12-3
Deleting Schemas 12-4
Copying Schemas 12-5
Adding a Parameter to a Schema 12-5
Deleting a Parameter from a Schema 12-7
Editing a Schema Parameter 12-7
Creating and Editing Model Definitions 12-8
Creating a Model Definition 12-9
Deleting a Model Definition 12-10
Copying a Model Definition 12-10
Filtering the List of Model Definitions 12-11
Editing a Model Definition 12-11
Saving Model Definitions 12-13
Creating and Editing Element Definitions 12-13
Creating an Element Definition 12-14
Editing an Element Definition 12-14
Deleting an Element Definition 12-18
Saving Element Definitions 12-18
Configuring 2D Icons 12-19
Editing Font-Based 2D Icons 12-19

Using Images for 2D Icons 12-21
Adding Images for Entities 12-26
Adding New Cockpit Display Models 12-27
Installing a Cockpit DLL 12-28
Creating a Model Definition for a Cockpit 12-29
Flipping DDS Textures for a Model 12-31
Configuring SpeedTree Trees 12-32
Specifying the SpeedTree Performance Profile 12-33
Best Practices for Creating Models for VR-Forces 12-34

12.1. Creating and Editing Schemas

The elements that make up a scene have parameters that help define them, for example a model filename. The set of parameters for a particular scene element is a schema. VR-Forces uses schemas to validate element definitions. You can create, copy, and delete schemas. For any schema, you can add and delete parameters and edit parameter values.



Changes to schemas are saved automatically. For more information, please see Section 5.6, "Managing VR-Forces Settings", in *VR-Forces Users Guide*.

VR-Forces includes model definition schemas. 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
- GridLines
- ParticleSystem
- Ribbon
- SpeedTree
- Terrain
- Tree
- WidgetTheme.
- Selection Visualizer.

12.1.1. 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 12-1).

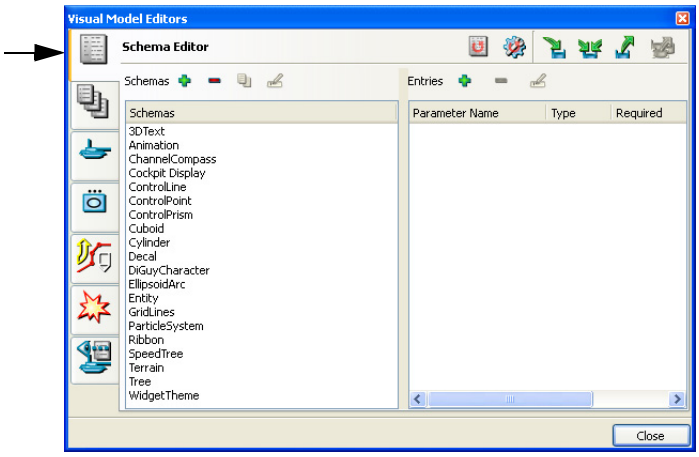


Figure 12-1. Schema Editor page

3. Click the Add icon (+) next to the Schemas label. The Create Instance dialog box opens (Figure 12-2).

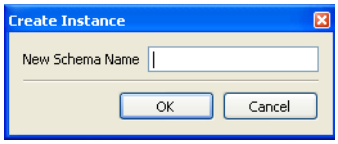


Figure 12-2. Create Instance dialog box

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 12-5.

12.1.2. Deleting Schemas

If a schema is used by a model definition, you cannot delete it. The Delete Schema icon 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 12-1).
3. In the Schemas list, select the schema you want to delete.
4. Click the Delete icon (−) next to the Schemas label. You are prompted to confirm the deletion.

5. Click Yes. The schema is removed from the list.

12.1.3. 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 12-1).
3. Select the schema that you want to copy.
4. Click the Duplicate icon ( ). 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 12-7.

12.1.4. 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 12-1).
3. In the Schemas list, select the schema to which you want to add a parameter.
4. Click the Add icon ( ) next to the Entries label. A parameter called "key" is added to the parameter list (Figure 12-3).

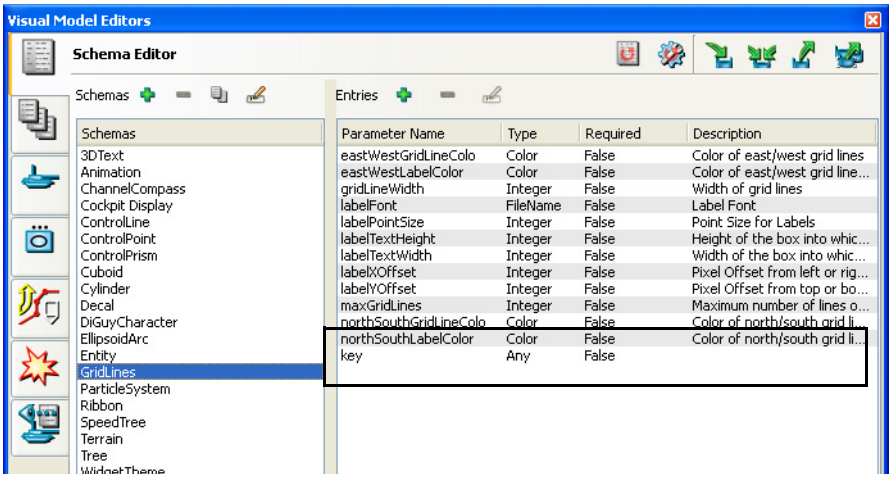


Figure 12-3. New schema parameter

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 definitions 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.

12.1.5. 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.
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 icon () next to the Entries label.

12.1.6. 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 12-1).
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.

12.2. Creating and Editing Model Definitions

Each model displayed in VR-Forces is defined by a set of parameters that are appropriate to the model type. (The parameters are specified in the model's schema. For information about model schemas, please see "Creating and Editing Schemas," on page 12-3.) 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.

All model configuration is done in the Model Definition Editor page of the Visual Model Editors dialog box (Figure 12-4). The icons all have tooltips. For more information, please see Section 5.6, "Managing VR-Forces Settings" and Section A.5, "Common VR-Forces Icons", in *VR-Forces Users Guide*.

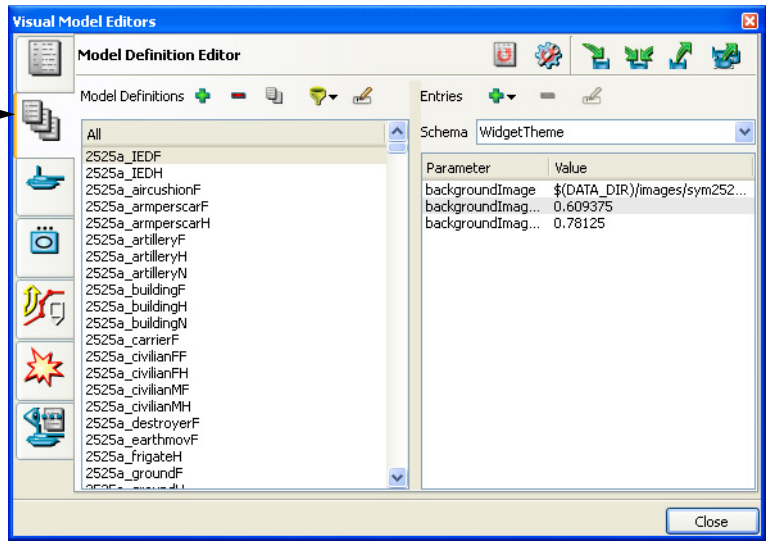


Figure 12-4. Model Definition Editor page

- Any model definition that uses the Entity, Tree, or SpeedTree schema is automatically added to the Prop 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 Entity Editor. For details, please see "Editing an Entity's Visual Model," on page 4-21.

12.2.1. 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 12-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 12-11.)
4. Click the Add icon (+) next to the Model Definitions label. The Add Model Definition dialog box opens (Figure 12-5).

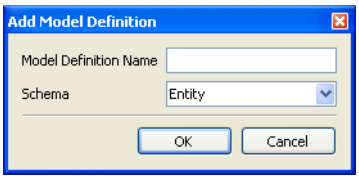


Figure 12-5. Add Model Definition dialog box

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 drop-down 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 12-11.

12.2.2. 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 12-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 12-11.)
4. Select the model that you want to delete.
5. Click the Delete icon () next to the Model Definitions label. You are prompted to confirm the deletion.
6. Click Yes. The model definition is removed from the list.

12.2.3. 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 12-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 12-11.)
4. Select the model that you want to copy.
5. Click the Duplicate icon (). The Duplicate Model Definition dialog box opens (Figure 12-6).

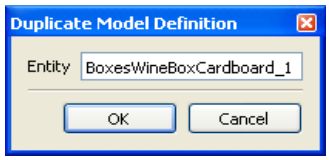


Figure 12-6. Copy model definition

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 12-11.

12.2.4. 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 icon (▾) and select a schema type from the list.

12.2.5. 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 an entity that is already in the scene, the updated definition is not applied to the entity. To apply an edited model definition to all entities, 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 12-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 12-11.)
4. Select the model for which you want to add a parameter.
5. Click the Add Parameter icon (▾). A list of parameters is displayed (Figure 12-7).

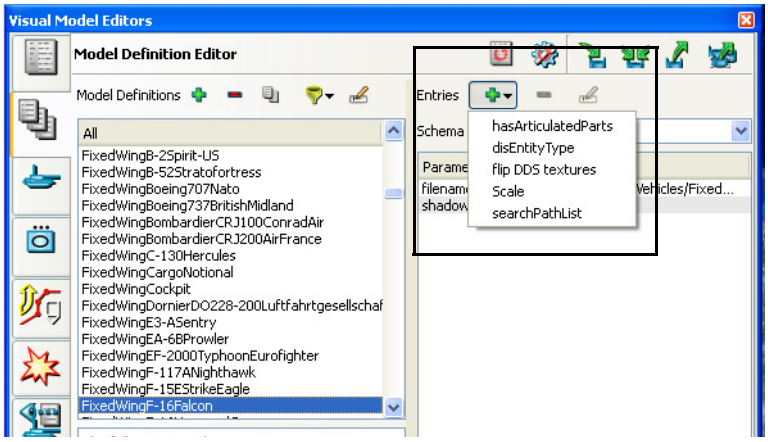


Figure 12-7. Add parameter list

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 12-8).

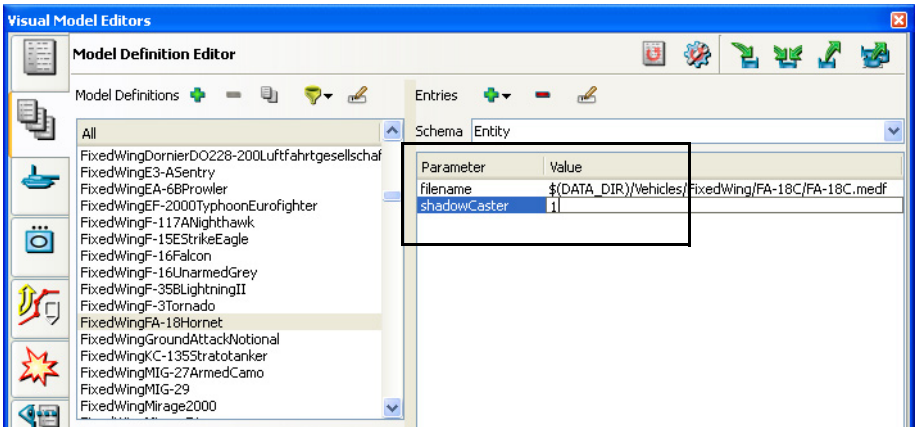


Figure 12-8. Editable parameter

8. Type 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 12-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 12-11.)

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 12-8).
6. Type a 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 12-4).
3. Optionally, filter the model definitions in the list. (For details, please see "Filtering the List of Model Definitions," on page 12-11.)
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 icon () next to the Entries label. The parameter is removed.

12.2.6. 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 Section 5.6, "Managing VR-Forces Settings", in *VR-Forces Users Guide*.

12.3. Creating and Editing Element Definitions

Element definitions let you assign visualizers to scene elements (entities, tactical graphics, interactions, aggregates, cultural features, and spot reports). Element definitions use a tree-style hierarchy to assign visualizers. At any level of the hierarchy an element definition inherits the visualizers of its parents. This makes it possible to assign a visualizer 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 visualizer.

You can create and edit element definitions. The procedures are the same for each element type.

12.3.1. 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 (Figure 12-9).

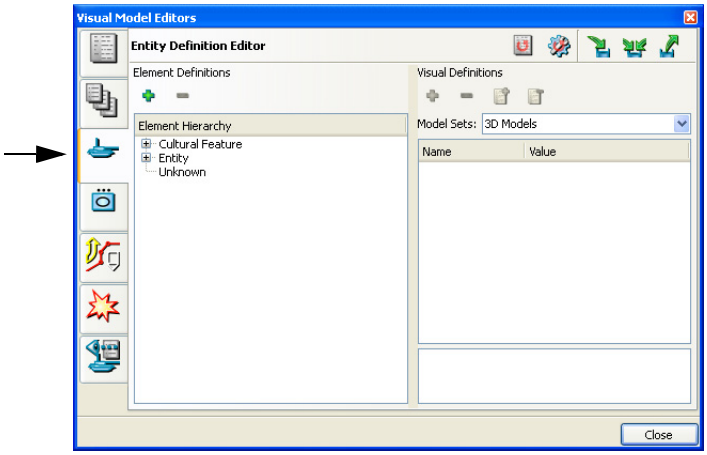


Figure 12-9. Entity Definition Editor page

3. Expand the element definition hierarchy to the level at which you want to add an element definition and select the parent element.
4. Click the Add icon (✚). 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 visualizers assigned to the element. For details, please see "Editing a Visualizer," on page 12-15.
8. Optionally, add a visualizer to the element. For details, please see "Adding a Visualizer," on page 12-15.
9. Optionally, save the element. For details, please see "Saving Element Definitions," on page 12-18.

12.3.2. Editing an Element Definition

An element definition consists of a list of visualizers for that element. You can add visualizers and you can edit a visualizer's parameters. In most cases, you cannot delete an inherited visualizer.

Adding a Visualizer

To add a visualizer 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 12-9).
3. Select the element definition that you want to edit.
4. Optionally, on the Model Sets drop-down list, select the model set you want to use.
5. Click the Add icon (✚) next to the Visual Definitions label. The Create Visualizer Definition dialog box displays a list of available visualizers.
6. Select the visualizer that you want to add.
7. Click OK. The visualizer is added to the element.
8. Optionally, edit the parameters for the visualizer. For details, please see "Editing a Visualizer," on page 12-15.

Editing a Visualizer

Visualizers and their attributes are displayed either as grey italicized text or as black text (Figure 12-10). Italicized text indicates that the visualizer or attribute is inherited from a parent element. Visualizers and attributes that are not italicized are particular to that element. (These entries could be inherited by child elements.)

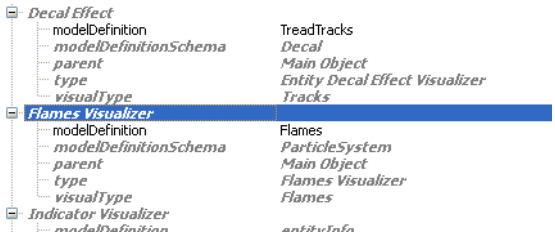


Figure 12-10. Visualizer entry

To edit a visualizer 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 12-9).
3. Select the element definition that you want to edit.
4. In the visualizer pane, click the value of the attribute that you want to edit. Depending on the attribute, a drop-down list of available values is displayed, a dialog box is displayed (Figure 12-11), or the field becomes editable.

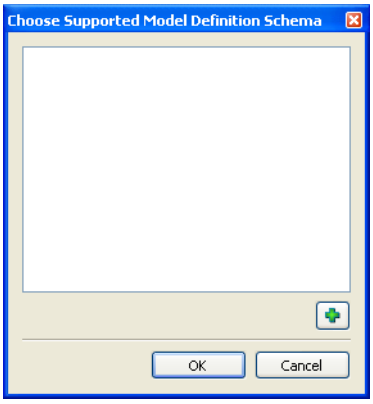


Figure 12-11. Choose Supported Model Definition Schema dialog box

5. For a drop-down 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 drop-down list, click the Add icon (). A drop-down 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 visualizer 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 visualizer 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 12-9).
3. Select the element definition that you want to edit.
4. In the visualizer 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 drop-down lists (Figure 12-12).

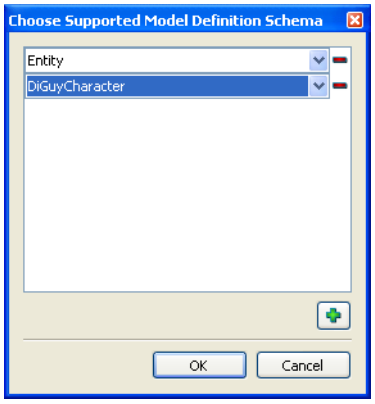


Figure 12-12. Parameter with multiple values

5. To add a value, click the Add icon (). A new list is added to the window.
6. Select a value from the new list.
7. To remove a value, click the Delete icon () next to the list for that value.
8. Click OK, The visualizer list does not show all of the values, but clicking it will open the dialog box again and show the selected values.

Adding an Attribute to a Visualizer Definition

Each visualizer 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 visualizer.

To add an attribute to a visualizer:

1. Select the visualizer that you want to edit.
2. Click the Add Attribute icon (). The Add Visualizer Definition Attribute window opens. It lists the attributes that are available for the visualizer (Figure 12-13).

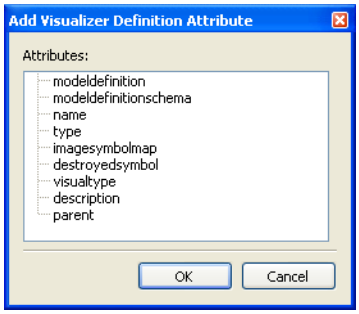


Figure 12-13. Add Visualizer Definition Attribute window

3. Select the attribute that you want to add.

4. Click OK. The attribute is added to the visualizer.
5. Specify a value for the attribute. This might require typing a value, selecting a value from a drop-down list, or selecting a value in another window.

Deleting an Attribute from a Visualizer Definition

You can delete individual attributes from a visualizer definition.

The Delete icon above the list of visualizer definitions deletes entire visualizer definitions. Do not click it thinking it will delete just the selected attribute.

To delete an attribute:

1. Select the visualizer that you want to edit.
2. Select the attribute that you want to delete.
3. Click the Delete Attribute icon ().

12.3.3. 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 12-9).
3. Select the element definition that you want to delete.
4. Click the Delete icon () next to the Entity Definitions label.
5. Confirm the deletion.

12.3.4. 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 Section 5.6, "Managing VR-Forces Settings", in *VR-Forces Users Guide*.

12.4. Configuring 2D Icons

The 2D icons in VR-Forces are configured in visualizers. By default, VR-Forces uses font-based icons that are configured in the Military Symbol Icon visualizer. If you want to use images for 2D icons, you can do so with the Entity Image Symbol visualizer.



You can quickly change the 2D icon for an entity in the Entity Editor. For details, please see "Editing an Entity's Visual Model," on page 4-21.

12.4.1. Editing Font-Based 2D Icons

By default, VR-Forces 2D icons are configured in the Military Symbol Icon visualizer, which is part of an entity's entity definition. This visualizer uses font-based icons. You can change an entity's icon by changing the fillGlyph and outlineGlyph attributes.

To change an entity's icon:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 12-9).
3. Select the entity whose icon you want to edit.
4. In the Model Sets drop-down list, select 2D Icons.
5. To change the fill glyph:
   a. Under the Military Symbology Icon visualizer, select the fillGlyph attribute (Figure 12-14).

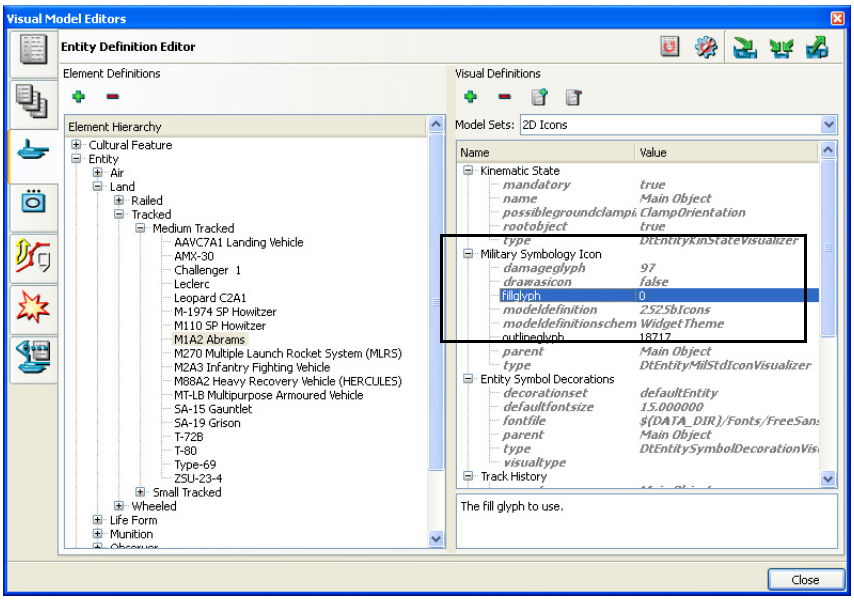


Figure 12-14. Military Symbol Icon Visualizer

b. Click the value in the Value column. The Choose Font Glyph Index dialog box opens (Figure 12-15).

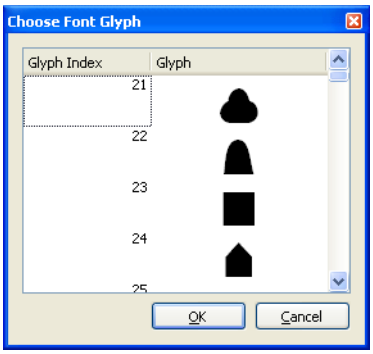


Figure 12-15. Choose Font Glyph dialog box

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.

12.4.2. 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 visualizer.



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 Definitions tab (Figure 12-9). (If you want to change the icon for an aggregate, select the Aggregates tab. To change the icon for a waypoint, select the Tactical Graphics tab.)
3. In the Element Definitions list, select Entity.
4. In the Model Sets drop-down list, select 2D Icons.
5. In the Element Definitions list, select Entity. (If you are editing aggregate 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.)
6. In the Visual Definitions list, select Military Symbology Icon (Figure 12-16).

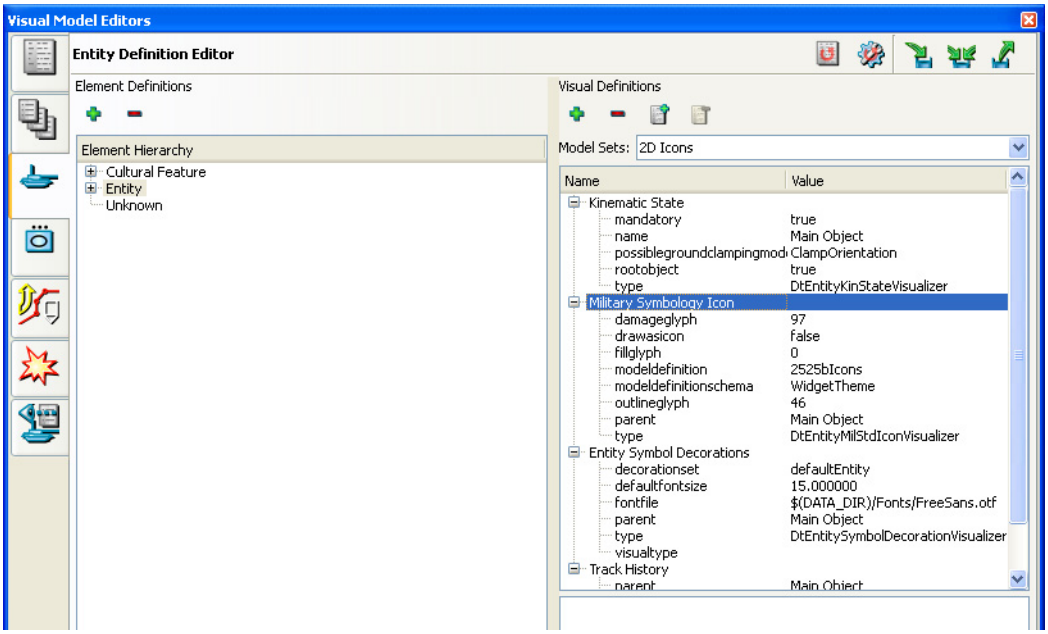


Figure 12-16. Entity element definition with Military Symbology Icon visualizer

7. Click the Delete icon (−).
8. Click the Add icon (+). The Create Visualizer Definition window opens. It displays a list of visualizers (Figure 12-17).

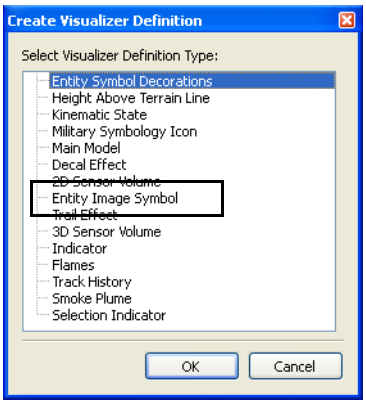


Figure 12-17. Create Visualizer Definition window

9. Select Entity Image Symbol. (If you are configuring images for aggregate entities, select Aggregate Image Symbol; for tactical graphics Tactical Graphic Image Symbol.)
10. Click OK.
11. In the Visual Definitions list, select Entity Image Symbol (Figure 12-18).

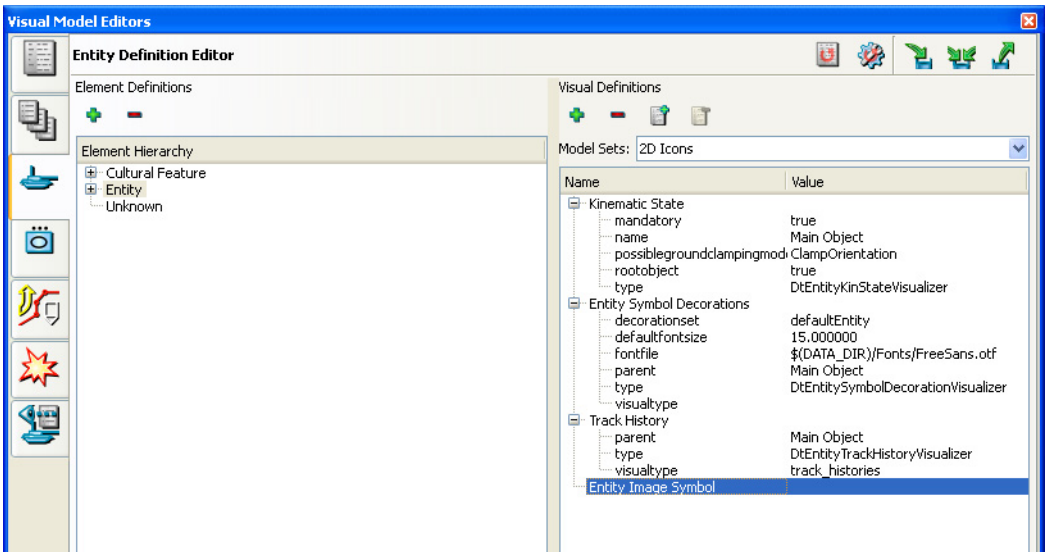


Figure 12-18. Entity element definition with Entity Image Symbol visualizer

12. Click the Add Attribute icon (). The Add Visualizer Definition Attribute window is displayed (Figure 12-19).

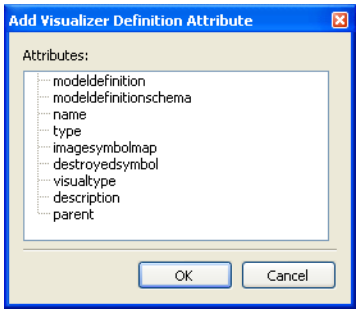


Figure 12-19. Add Visualizer Definition Attribute window

13. In the attribute list, select **parent**.
14. Click OK. The **parent** attribute is added to the visualizer.
15. Click the value field to make it editable.
16. Type Main Model and press **Enter**.
17. Click the Add Attribute icon.
18. In the attribute list, select **type**. The value is filled in automatically.
19. Click the Add Attribute icon.
20. In the attribute list, select **modeldefinitionschema**. The Choose Supported Model Definition Schema dialog box opens (Figure 12-20).

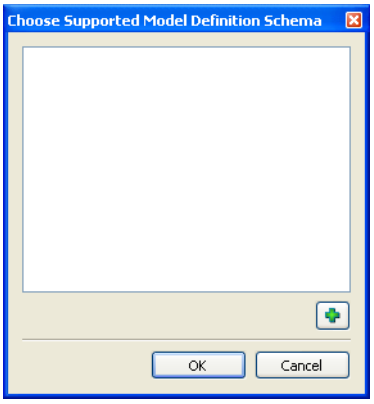


Figure 12-20. Choose Supported Model Definition Schema dialog box

21. Click the Add icon. A drop-down list is added to the dialog box.
22. In the list, select **WidgetTheme** (Figure 12-21).

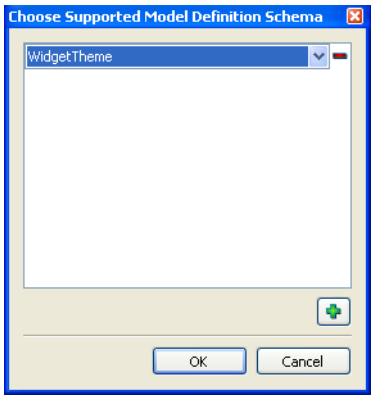


Figure 12-21. Completed Choose Supported Model Definition Schema dialog box

23. Click OK.
24. Click the Add Attribute icon.
25. In the attribute list, select **imagesymbolmap**. The 2D Symbol Image Mapper dialog box opens (Figure 12-22).

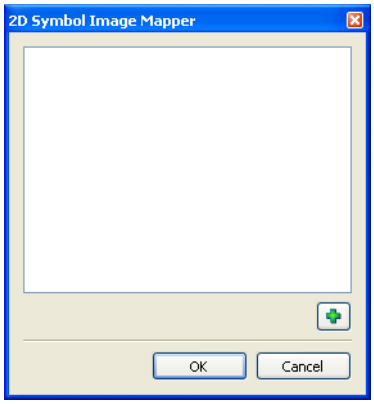


Figure 12-22. 2D Symbol Image Mapper dialog box

26. Click the Add icon. A drop-down 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
27. Select a force and an image from the lists (Figure 12-23).

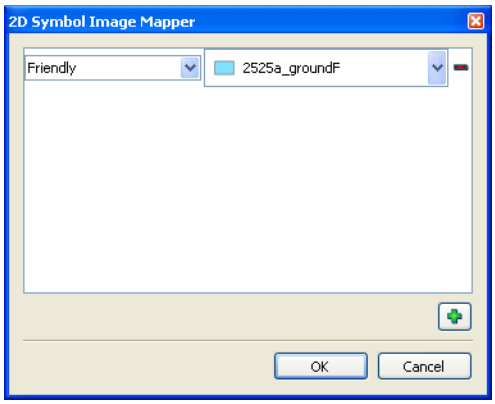


Figure 12-23. 2D Symbol Image Mapper dialog box with force chosen

28. Add an entry for each force that you want to represent with an image.
29. Click OK.
30. Click the Add Attribute icon.
31. In the attribute list, select **destroyedsymbol**. The **destroyedsymbol** attribute is added to the visualizer.
32. In the **destroyedsymbol** list, select Destroyed. The completed visualizer should look similar to Figure 12-24.

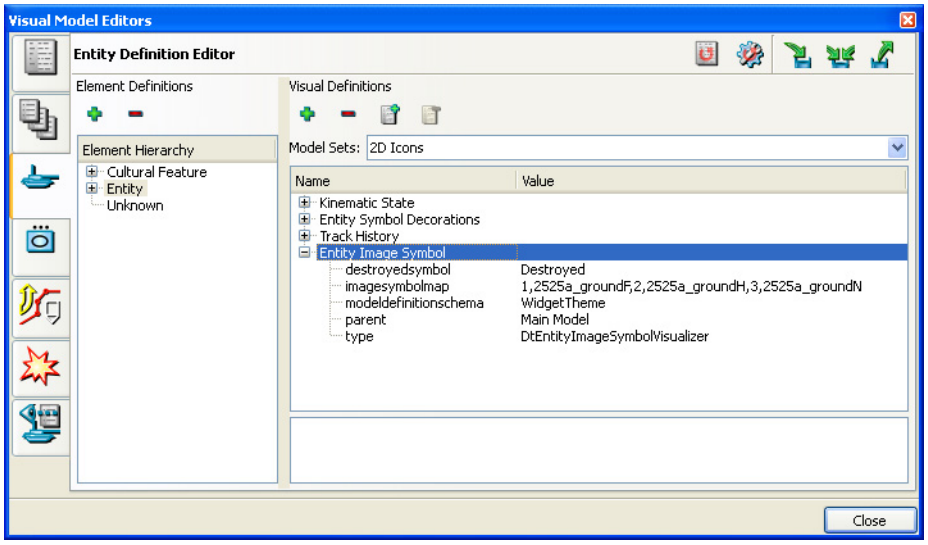


Figure 12-24. Entity Image Symbol visualizer

The settings for the Entity Image Symbol visualizer areinherited 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 entity type, such as Fixed Wing and Life Form, to use the proper image for that entity type. Optionally, you can specify an image for each entity definition at the leaf level.



When you edit the Entity Image Symbol visualizer attributes, be sure that you select the 2D Icons model set.

12.4.3. Adding Images for Entities

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 visualizer.

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 12-4).
4. In the Schema drop-down 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 12-8. 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.

12.5. 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 12-25 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.

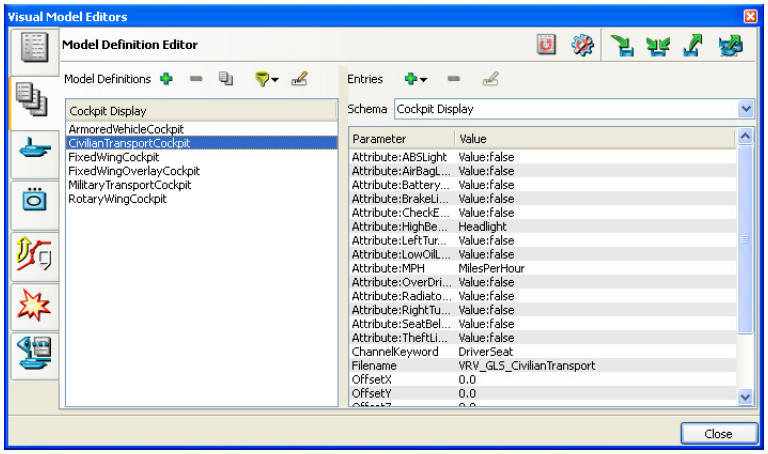


Figure 12-25. Cockpit model definition

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 entity types to the model definition, as described in "Adding a Model Mapping," on page 11-3.

12.5.1. 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.

12.5.2. 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 12-9. Be sure to select Cockpit Display as the schema. A new model definition is added to the display (Figure 12-26). The message window lists the parameters that you must add. These are the parameters defined by the schema.

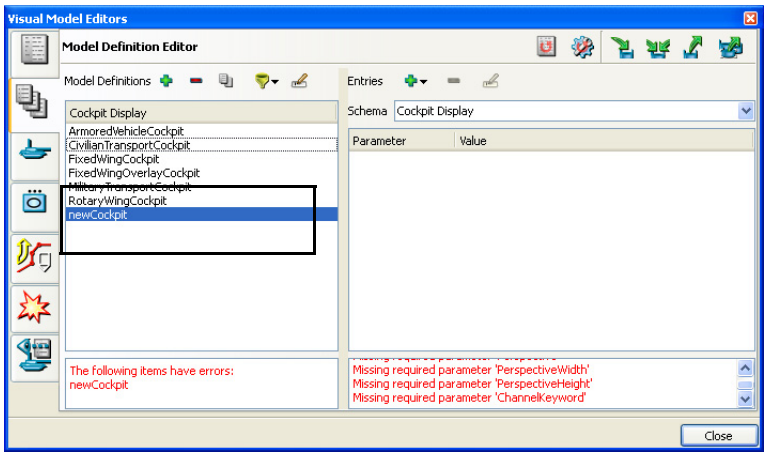


Figure 12-26. New cockpit model definition

2. Add the required parameters, as described in "Adding Parameters to a Model Definition," on page 12-11. Table 12-1 describes the required parameters.

Table 12-1: 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. Currently, only 2D cockpit displays are provided with VR-Forces.
PerspectiveWidth PerspectiveHeight	Adjusts the orthogonal projection for a 2D cockpit. Adjusts the perspective projection for a true 3D cockpit.

Table 12-1: Required parameters for cockpit models

Parameter	Description
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 12-11.
   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 12-2.
4. Map an entity type to your definition, as described in "Adding a Model Mapping," on page 11-3.

Table 12-2: 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).
Roll	Returns the roll of the entity clamped (-180 to 180 degrees).
Yaw	Returns the yaw of the entity clamped (0 to 360 degrees).

Table 12-2: Updaters provided

Updater	Description
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.

12.6. 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 10-25.)

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 12-4).
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 icon () to see if this is a valid parameter for this model. If it is, you can add the parameter and set it to 1.

12.7. 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 9-10.

To configure SpeedTree trees:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the SpeedTree Settings page (Figure 12-27).

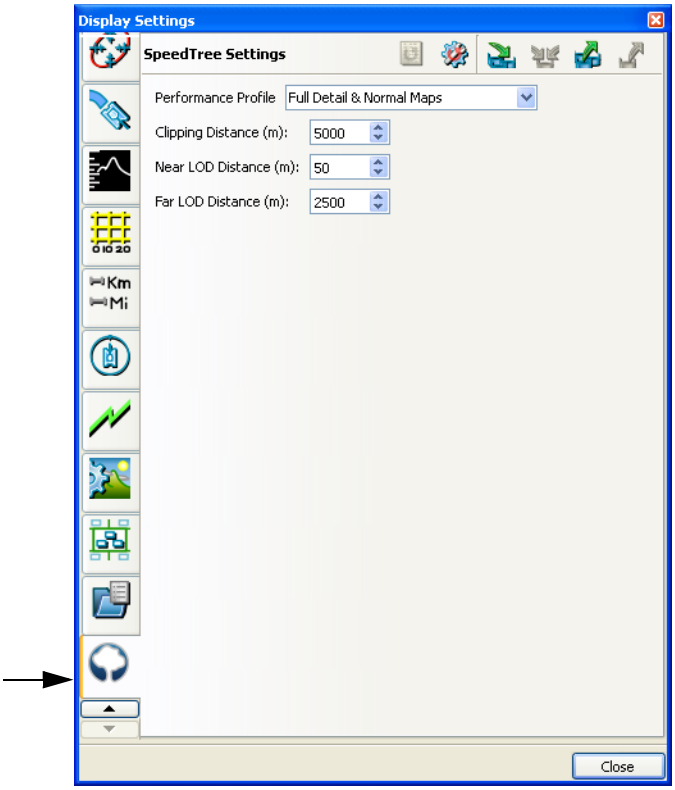


Figure 12-27. SpeedTree Settings page

3. In the Performance ~~group box~~Profile drop-down list, select a performance profile. (For details, please see "Specifying the SpeedTree Performance Profile," on page 12-33.)
4. Specify the clipping distance, and the near and far LOD distance.
5. Click Close.

12.7.1. Specifying the SpeedTree Performance Profile

The number of SpeedTree objects in a scene can affect performance. Therefore, VR-Forces provides several profiles for display of SpeedTrees so that you can balance performance against visual quality. You specify the performance profile on the SpeedTree Settings page of the Display Settings dialog box. Table 12-3 describes the profiles.

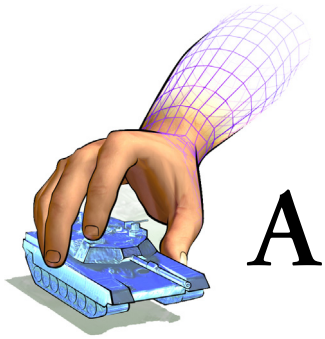
Table 12-3: SpeedTree Performance Profiles

Profile	Branch Texture	Frond Texture	Vertical Billboards	Horizontal Billboards
Disabled				
Single Color Map	Color	Color	Low LOD	
Partial Detail & Normal Maps	Color Detail Normal map	Color	Low LOD	
Full Detail & Normal Maps	Color Detail Normal map	Color Normal map	Low LOD	
Full Detail & Normal & Shadow Maps	Color Detail Normal map Shadow	Color Detail Normal map	Low LOD	Low LOD

12.8. Best Practices for Creating Models for VR-Forces

If you build your own models for use in VR-Forces, following the guidelines in this section 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.



A

Editing the Create Menu

This chapter describes how to edit the Create menu manually.

The Create Menu A-2

The vrf.ent Configuration File A-2

- Object Blocks in vrf.ent A-4
- Reorganizing the Create Menu A-4
- Editing Menu Items A-4

Adding New Entity Types to VR-Forces A-5

- Adding a New Entity or Control Object to vrf.ent A-5

A.1. The Create Menu

By default, the Create menu is generated automatically from the *entity.mst* file. However, you can use a customized Create menu if you want to. The recommended way to create a customized Create menu is to use the Entity Editor, as described in "Editing the Create Menu," on page 4-40. This appendix explains how to edit the Create menu in a text editor.

You can change the Create menu by:

- Reorganizing entries
- Editing existing entries
- Adding new entries.

You modify the Create menu and its sub-menus by editing *./data/simulationModel-Sets/<model_set>/gui/vrf.ent*. If you make changes to *vrf.ent*, you may also need to make changes to the object parameter database.

A.2. The vrf.ent Configuration File

The *vrf.ent* file contains the information VR-Forces uses to build the Entity and Aggregates menu options on the Create menu, and the list of control objects. The menu structure is completely configurable. It consists of a set of nested menu blocks. The highest block specifies the name of the menu. The second level specifies entries on the main menu. Nested menu blocks specify entries for submenus The `create-entity-menu-item` entries specify the objects you can create. Figure A-1 is a highly edited version of the *vrf.ent* file that shows nested menus and an illustration of the menu structure that would be created by the sample *vrf.ent* file.

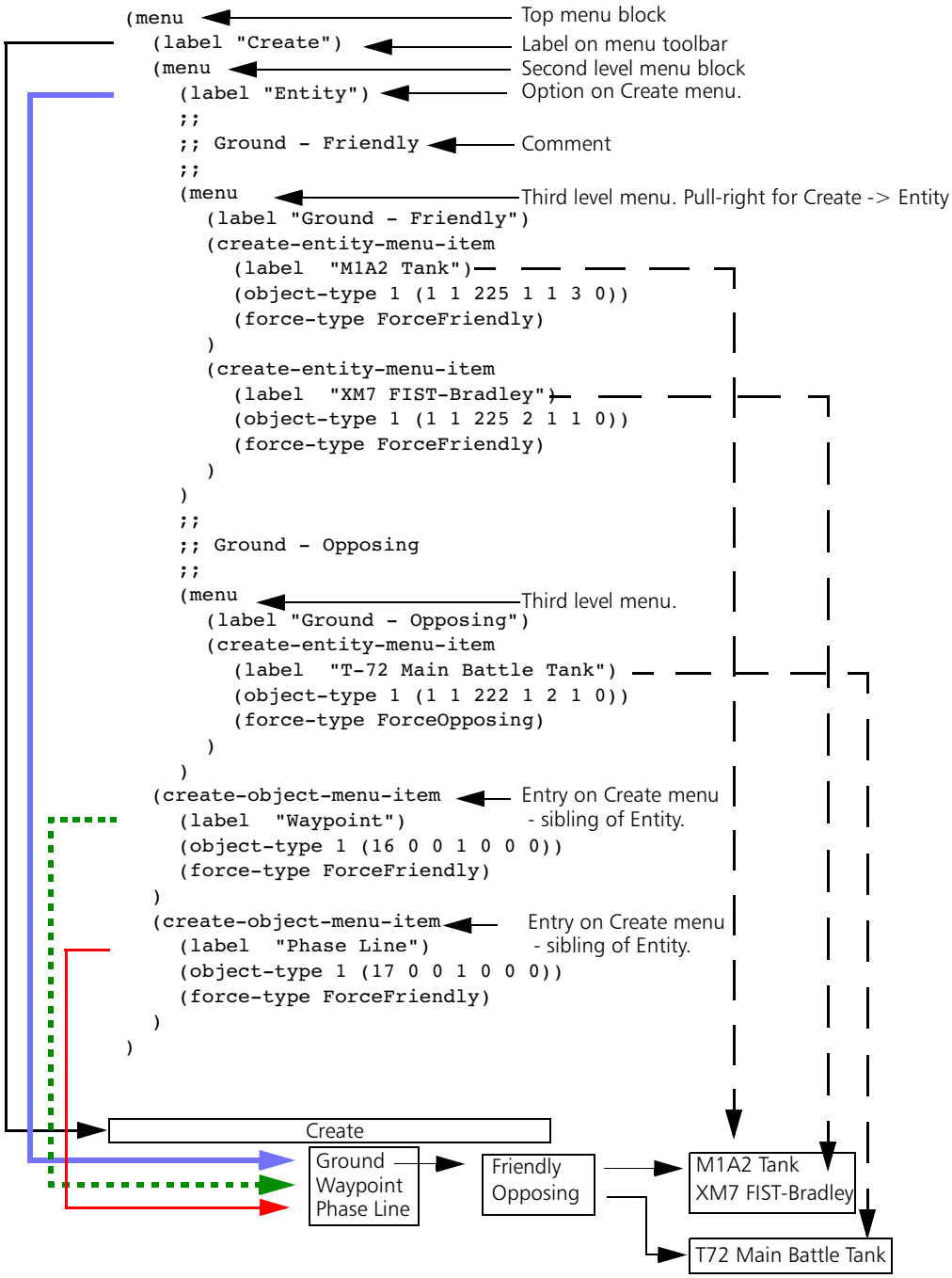


Figure A-1. Abbreviated *vrf.ent* file and menus created

A.2.1. Object Blocks in *vrf.ent*

The entry for each type of entity that you can create defines a label, object type, and force type, similar to the following:

```
(create-entity-menu-item
  (label  "M1A2 Tank")
  (object-type 1 (1 1 225 1 1 3 0))
  (force-type ForceFriendly)
)
```

where:

- **label** specifies the text that is displayed on the menu.
- **object-type** is the eight-digit object type for the object. It must match an object-type in the object parameter database using exact match or best match criteria.
- **force-type** determines the force assigned to a newly created object. The options are:
  - ForceOther
  - ForceNeutral
  - ForceFriendly
  - ForceOpposing
  - ForceUserDefined5 up to ForceUserDefined255.

A.2.2. Reorganizing the Create Menu

The default *vrf.ent* file organizes entities by force and domain. It organizes aggregates by hierarchical level. If you would prefer to organize entities differently, all you have to do is create menu blocks with your preferred organization and move the appropriate `create-entity-menu-item` blocks to the new menu blocks. For example, you could move the control object entries to a Control Objects sub-menu.

A.2.3. Editing Menu Items

You can change the labels and force types for entries on the Create Menu by editing the text in *vrf.ent*. You can also change the objects associated with the entries by editing the **object-type**. However, if you change an object type, you must be sure that it matches an entry in the object parameter database. Please see "Adding New Entity Types to VR-Forces," on page A-5 for details.

A.3. Adding New Entity Types to VR-Forces

You can add new entity types to VR-Forces without using the VR-Forces API toolkit.

To add a new entity type, you must:

1. Add the entity to the Create menu.
2. Ensure that the object parameter database has an entry whose object-type matches the object type in *vrf.ent* or meets best-match criteria.

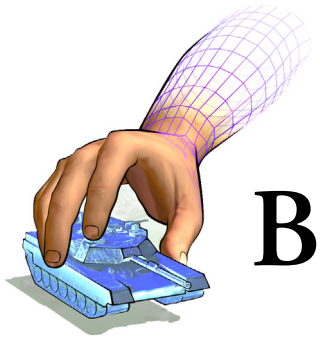
A.3.1. Adding a New Entity or Control Object to *vrf.ent*

Before making changes to a configuration file, be sure to make a backup copy of the original file.

To add a new object to *vrf.ent*:

1. Open *vrf.ent* in a text editor.
2. To add an entity (including preconfigured aggregates), add a new **create-entity-menu-item** block to the file. To add a control object, add a new **create-object-menu-item**. Be sure that you match the syntax and parentheses of an existing block. (You could copy a block, paste it at the end of the existing block, and change the entries for the various fields as described in the following steps.)
3. In the **label** field, enter a string for the new entity. This is the text that will be displayed on the menu.
4. In the **object-type** field, enter an 8-digit object-type for the entity. Do not use wild-card values. The object type must match (using the best match method) an object type that is specified in *vrfSim.opd* (or an alternative OPD file.)
5. In the **force-type** field, enter the force type for the entity.
6. Save the file.

The next time you start VR-Forces, the new entity will be listed on the appropriate submenu of the Create menu.



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 an Entity 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
Entity Parameters B-8
Object Type Enumerations B-9
Object Super-type B-9
AggregateKind B-9
ObjectKind B-10
Aggregate Category B-10
Aggregate Specific B-11
Point Object Category B-12
LinearObjectKind B-12
LinearObjectCategory B-12
Interaction B-12

B.1. The Structure of an Entity Parameter

The object parameter database uses MÄK Technologies LISP (MTL). (For details about MTL, please see "MÄK Technologies Lisp (MTL) Files," on page 2-10.) 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.

B.2. The Parameter Type Hierarchy

Figure 6-3, in "Parameter Types," on page 6-9, 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 6-6.
parameter-type	(Required) One of the following (string): ◆ "vrf-base-object-param" ◆ "vrf-object-param" ◆ "moving-object-param" ◆ "fixed-wing-param" ◆ "ground-vehicle-param" ◆ "rotary-wing-entity-param" ◆ "missile-param" ◆ "surface-entity-param" ◆ "human-param" ◆ "cultural-feature-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 entity 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 entity-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. Pseudo-aggregate entities have this kind of state repository. The more entity-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. ◆ "human-state-repository": For lifeform entities. ◆ "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 an entity's state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of entity you are simulating. May be one of the following types: ◆ "aggregate-local-entity-net-interface": For aggregate entities. ◆ "pseudo-aggregate-local-entity-net-interface": For aggregate entities that have subordinate entities that are separately simulated. ◆ "individual-local-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. ◆ "local-environmental-net-interface": For entities that are best represented as an environmental process. Non-entity objects such as control objects (waypoints, routes, areas) 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 an entity'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 entities). Expected value: "component-manager"
plan-manager	(Optional) The entity plan manager. If you want an entity 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 true aggregates: "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 entity-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 entity-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 individual and aggregate entities. ◆ "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. ◆ "cultural-feature-state-repository": For cultural features that are created as entities. ◆ "surface-entity-state-repository": For surface 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 an entities state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of entity you are simulating. May be one of the following types: ◆ "vrf- aggregate -remote-entity-net-interface": For remote aggregate entities. ◆ "vrf-individual-remote-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. ◆ "vrf-remote-environmental-net-interface": For entities that are best represented as an environmental process. Non-entity objects such as control objects (waypoints, routes, areas) 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 an entity'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 aggregates and entities) 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 individual entities and aggregates. Parameters include support points, maximum speed, maximum reverse speed, turning radius, maximum slope, ordered speed, maximum acceleration, maximum deceleration, and fuel efficiency. Aggregates 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 entities

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
Cultural feature	cultural-feature-param	moving-object-param
Missile	missile-param	moving-object-param
Pseudo-aggregate	moving-object-param	vrf-object-param

B.3. Entity Parameters

For descriptions of entity 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, aggregate, and non-entity 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 entity types are not guaranteed to be unique across all of the possible individual and aggregate 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 aggregates that it simulates.

Other	0
Individual	1
Aggregate	2
PseudoAggregate	3
TrueAggregate	4
Interaction	5

B.4.2. AggregateKind

The enumeration for *DtAggregateKind* should start with 0, but VR-Forces starts with 10, to ensure that entity 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-entity objects, such as control objects (control points, routes, phase lines, obstacles, etc.)

DtPointObjectKind	16
DtLinearObjectKind	17
DtArealObjectKind	18
DtCircleObjectKind	19
DtTextObjectKind	20

B.4.4. Aggregate 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

Aggregate Subcategory

The Subcategory field enumerations for Kind field = Military Hierarchy specify the Echelon Type of the aggregate. 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. Aggregate Specific

The Specific field enumerations for Kind field = Military Hierarchy specify whether the aggregate 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*.

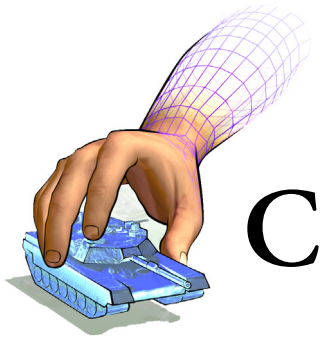
Other	0
ControlArea	1
Obstacle	2

ArealObjectCategory

B.4.9. Interaction

Kinds for Supertype interaction.

Other	0
Fire	1
Detonate	2



C

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 appendix contains tutorials that demonstrate how to combine various procedures to create terrains and how to create a local terrain server.

Composing a Terrain C-2
- Add Elevation Data (Add a Terrain Patch) C-3
- Add Imagery C-5
- Add a Feature Layer C-8
- Add Props C-9

Adding a Terrain Server C-13
- Connect to the SanLuisObispo Terrain Server C-15

Extracting Props from a Terrain C-17
- Create a New Terrain and Add a Terrain Patch C-17
- Extract the Props C-18
- Change the Model Definition for a Prop C-20

Adding Feature Data to a GDB Terrain C-21
- Create a New GDB Terrain in the TDB Tool C-22
- Add a Point Feature to the Terrain C-25
- Add the GDB Terrain to VR-Forces as a Feature Layer C-28

C.1. Composing a Terrain

This tutorial shows how to compose a terrain by combining elevation data, imagery, and feature data. 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 C-3).
2. Add imagery ("Add Imagery," on page C-5).
3. Add a feature layer ("Add a Feature Layer," on page C-8).
4. Add props ("Add Props," on page C-9).

C.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 C-1).

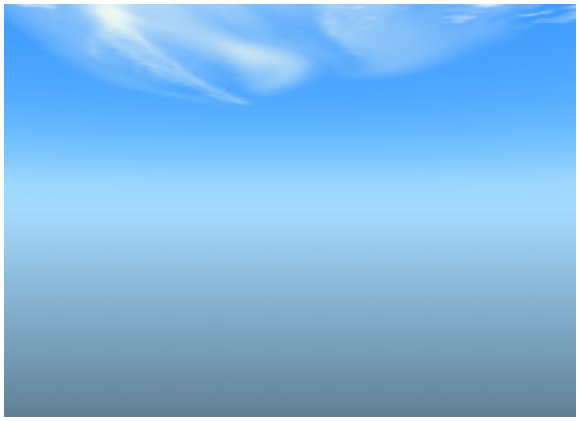


Figure C-1. No terrain loaded

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 C-2).

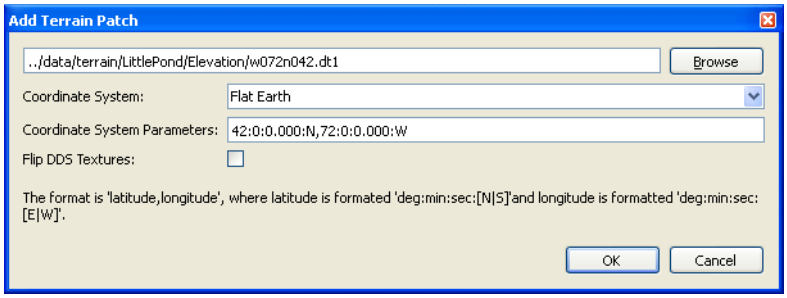


Figure C-2. Add Terrain Patch

10. Click OK. The terrain is added to the scene (Figure C-3).

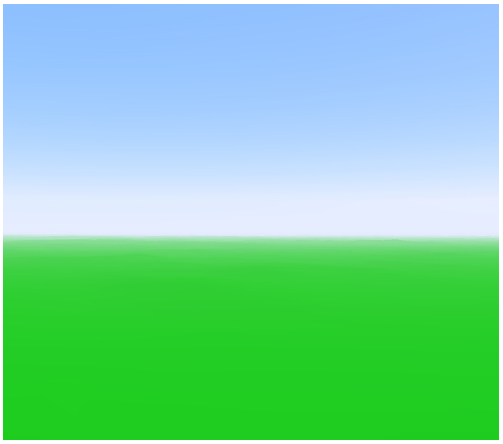


Figure C-3. Terrain patch added

C.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 C-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.

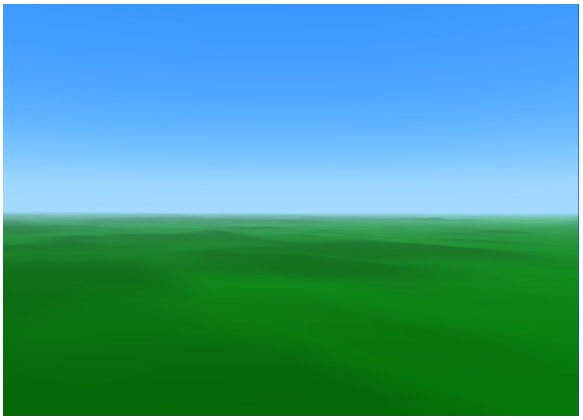


Figure C-4. Elevation coloring

To add image data:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Raster Maps page (Figure C-5).

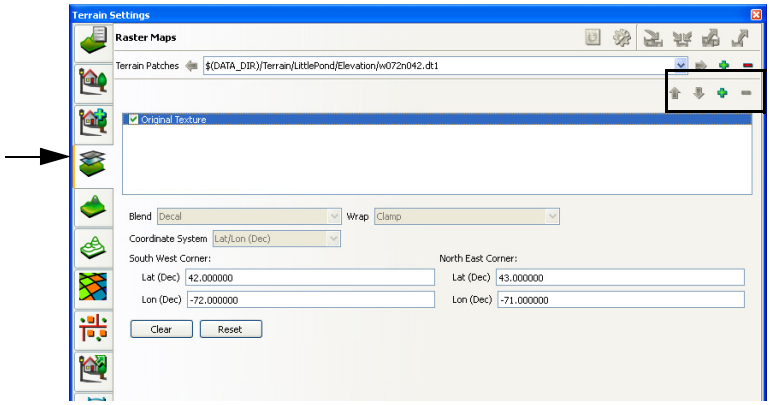


Figure C-5. Raster Maps page

3. Click the Add icon (✚) that is at the upper right of the list window (Figure C-5). A line labeled Image 1 is added to the list in the window (Figure C-6).

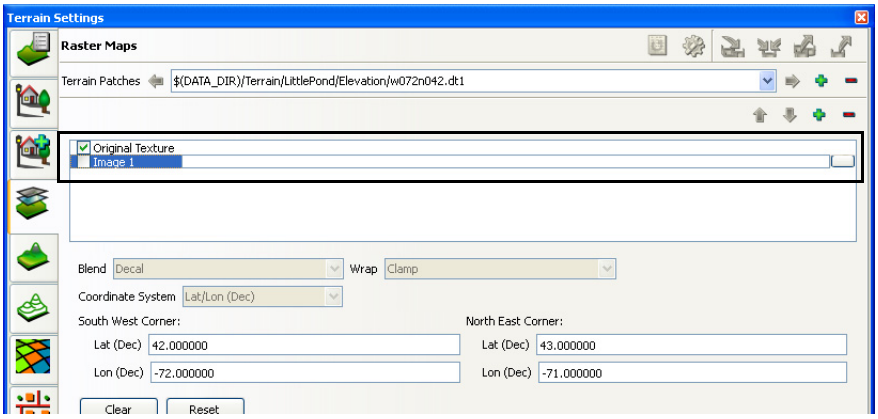


Figure C-6. New image

4. Click the Browse button at the right end of the image entry. The Image File dialog box opens.
5. In the Image File dialog box, navigate to *./data/Terrain/LittlePond/imagery*.
6. Select *W72N42.tif*.
7. 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 C-7).

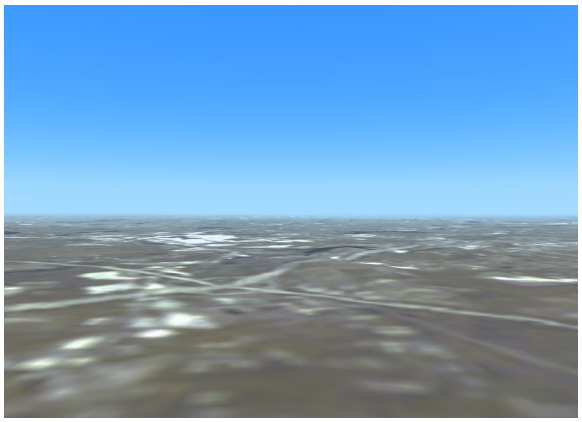


Figure C-7. Low resolution image

8. Add another image. This time, select *LittlePond.tif*.
9. Select the check box for the *LittlePond.tif* image (Figure C-8). A high resolution image is added to the Little Pond Area of the terrain (Figure C-9). (The high resolution image might be hard to find. The next section explains how to find it quickly.)

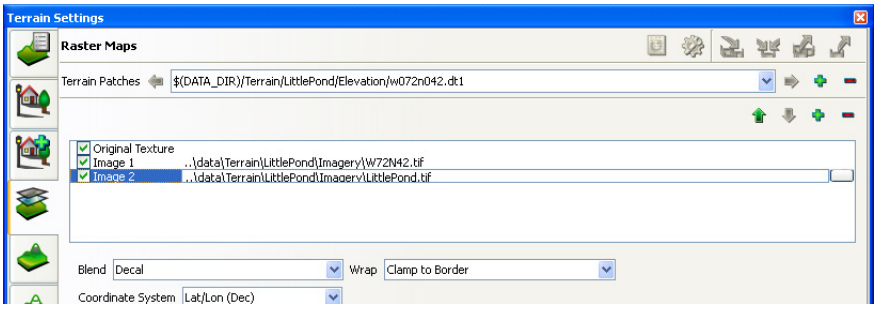


Figure C-8. Raster Maps page with both images added

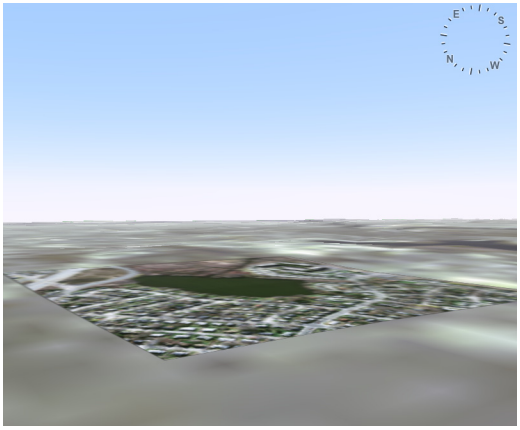


Figure C-9. Little Pond image

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 icon ( ). 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 C-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.

C.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 C-10).

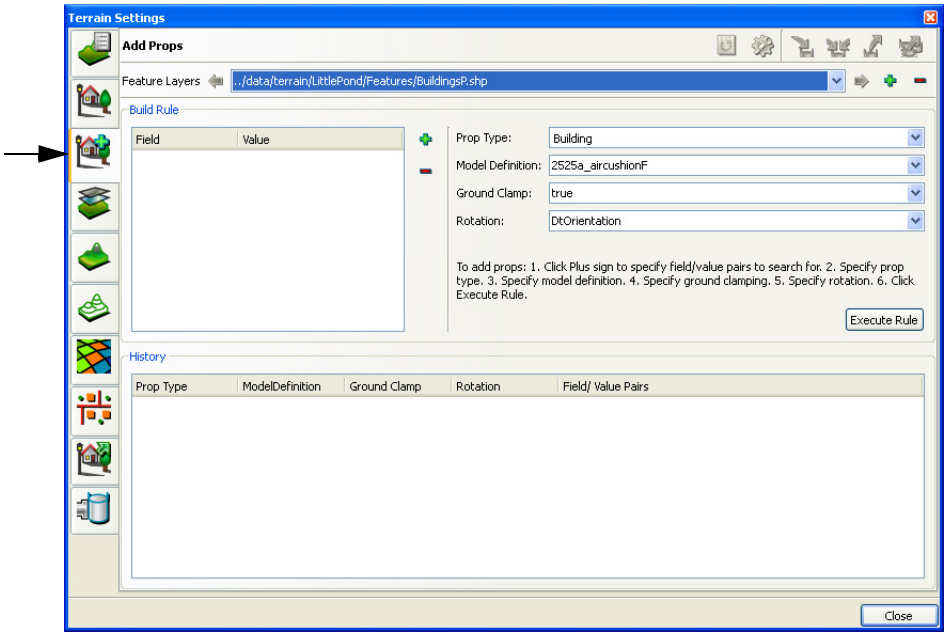


Figure C-10. Add Props page

3. Click the Add icon (+) 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.

C.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 icon (+) (Figure C-11). The Build Field/Value Pair dialog box opens.

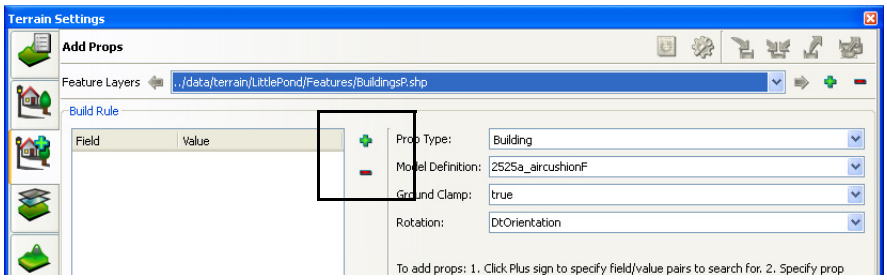


Figure C-11. Add build rule

2. In the Field drop-down list, select TYPE.
3. In the Value drop-down list, select HousesColonialGarrison (Figure C-12).

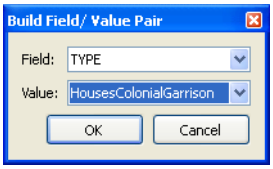


Figure C-12. Build Field/Value Pair dialog box

4. Click OK.
5. On the Add Props page, in the Prop Type drop-down list, select the prop type for the rule. In this case, select Building.
6. In the Model Definition drop-down list, select HousesColonialGarrisonClapboardWhite.
7. In the Ground Clamp box, specify whether or not you want the prop to be ground-clamped.
8. In the Rotation drop-down 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 C-13.

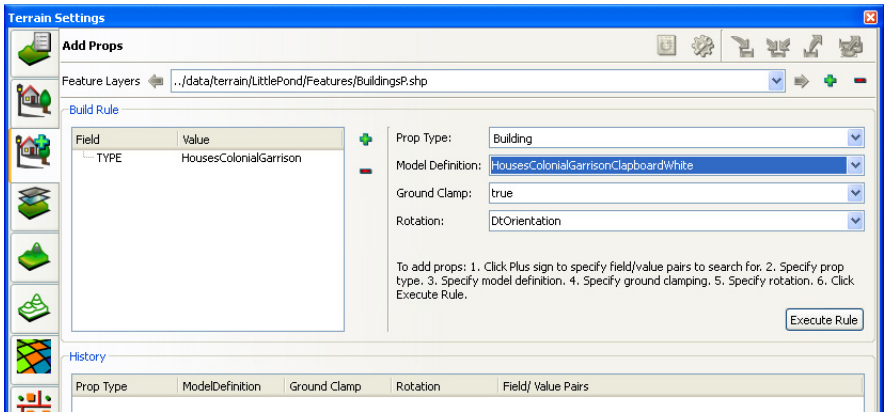


Figure C-13. Completed rule

9. Click Execute Rule. The rule is executed. The Add Props page is updated (Figure C-14) and houses are added to the terrain. Zoom in for a better look at them (Figure C-15). (You might need to close the Terrain Settings dialog box or move it out of the way to see the terrain.)

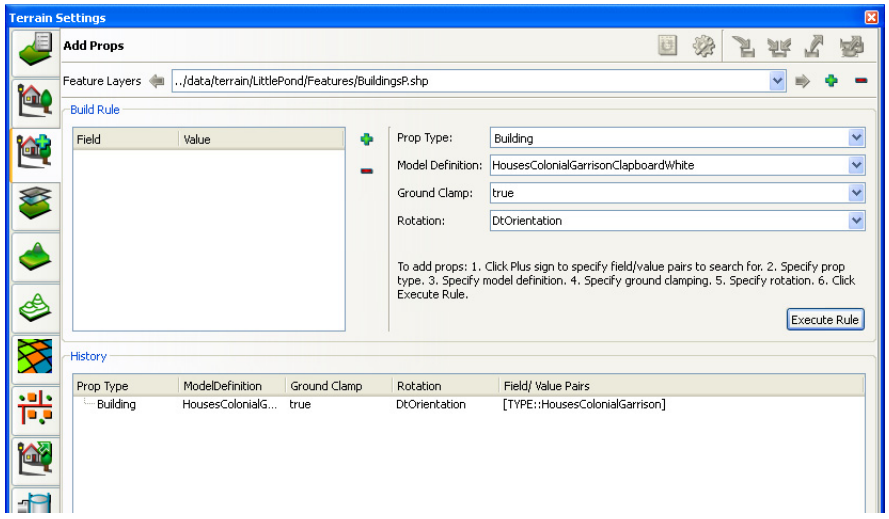


Figure C-14. Build rule implemented

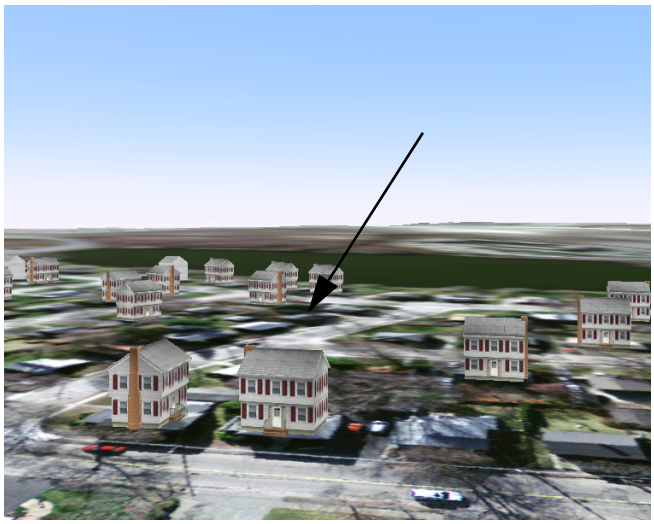


Figure C-15. Props added to terrain

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 icon (✚) 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 icon (✚). The Build Field/Value Pair dialog box opens.
7. In the field drop-down list, select VEG.
8. In the Value drop-down list, select TreesBigLeafMaple.
9. Click OK.
10. In the Prop Type drop-down list, select Tree.
11. In the Model Definition drop-down list, select TreesBigLeafMapleST. The build rule should look like (Figure C-16).

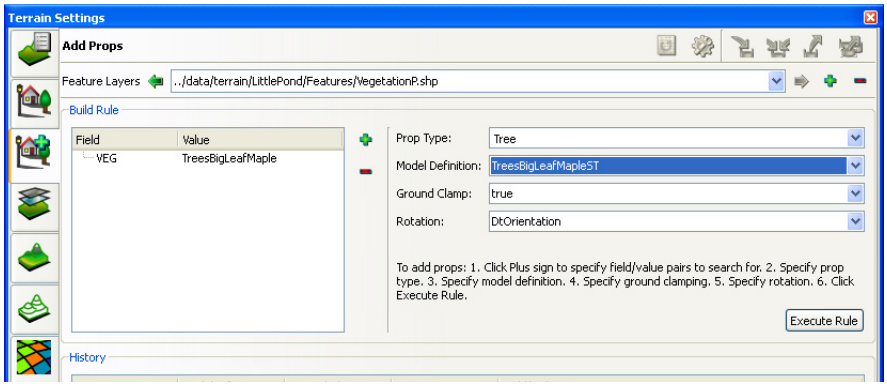


Figure C-16. Rule for big leaf maples

12. Click Execute Rule. The page is updated and trees are added to the terrain (Figure C-17).



Figure C-17. Little Pond with maples

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.

C.2. Adding a Terrain Server

This tutorial explains how to add a terrain server that streams data from directories on your local network. It uses the San Luis Obispo terrain data, which is provided with VR-Forces. In the tutorial, we copy an existing *.earth* file. Then we change the elevation and imagery data paths to point to the correct files and directories in *./data/Terrain/SanLuisObispo*.

To add a terrain server:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Server Settings page (Figure C-18).

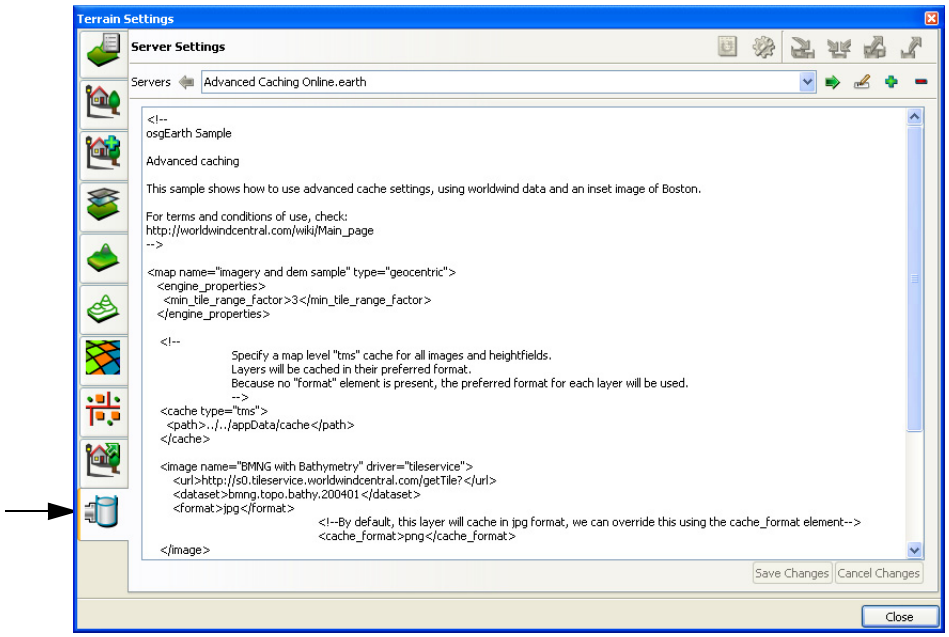


Figure C-18. Server settings page

3. In the Servers list, select *Local World.earth*.
4. Click the Add icon (+). A new configuration file is added to the list. It copies the contents of *Local World.earth* and names it *Local World_1.earth*.
5. The image element reads as follows:

```
<image name="World Imagery" driver="gdal">
      <url>../../data/Terrain/World/Imagery/World4km.tif</url>
  </image>
```

The SanLuisObispo/Imagery directory has multiple files, so we want to point the imagery URL to the directory, not to a specific file. VR-Forces will use all of the files as needed.

Change the image element name and URL to:

```
<image name="SanLuisObispo Imagery" driver="gdal">
  <url>../../data/Terrain/SanLuisObispo/Imagery</url>
</image>
```

6. Change the heightfield element's name value from:

   ```
   <heightfield name="World Elevation" driver = "gdal">
   ```

   to:

   ```
   <heightfield name="SanLuisObispo Elevation" driver = "gdal">
   ```

7. The elevation section of the file reads as follows:

   ```
   <url>../../data/Terrain/World/Elevation/earthnew/w001001.adf</url>
   ```

   If there is only one elevation file, the URL must specify the file. If there are multiple elevation files, the URL must specify the directory.

   Change the elevation URL line to:

   ```
   <url>../../data/Terrain/SanLuisObispo/Elevation</url>
   ```

8. Click the Rename icon ( ). (If you are prompted to save changes, click Save.) The Rename Server dialog box opens.

9. Change the name to *mySanLuisObispo.earth*. (If you are prompted to save changes, click Save.) The completed server should look like Figure C-19.

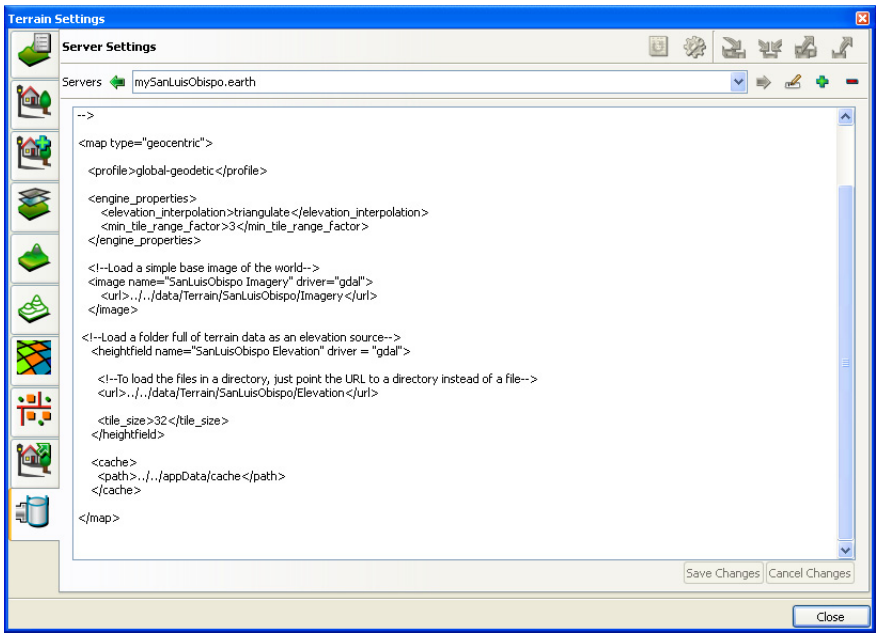


Figure C-19. *SanLuisObispo.earth* server

10. Click Save Changes. The new server file is saved to *./userData/servers*.

11. Close the Terrain Settings dialog box.

C.2.1. Connect to the SanLuisObispo Terrain Server

Now we will connect to the new server. We can also compare it to the San Luis Obispo terrain that ships with VR-Forces, which was created using the composable terrain method.

To connect to the server:

1. Choose **New** → **Scene**.
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 *mySanLuisObispo.earth*.
6. Click Connect. The server is loaded. It will be visible as a small speck in the window (Figure C-20), so you will have to move the observer closer to it (Figure C-21). Notice how as you move the observer closer, the terrain imagery gets paged in at greater levels of detail.

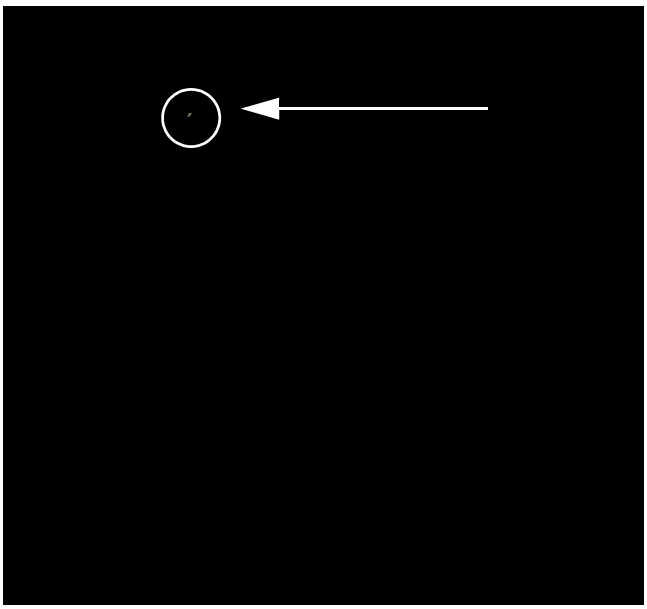


Figure C-20. *SanLuisObispo.earth* initial view

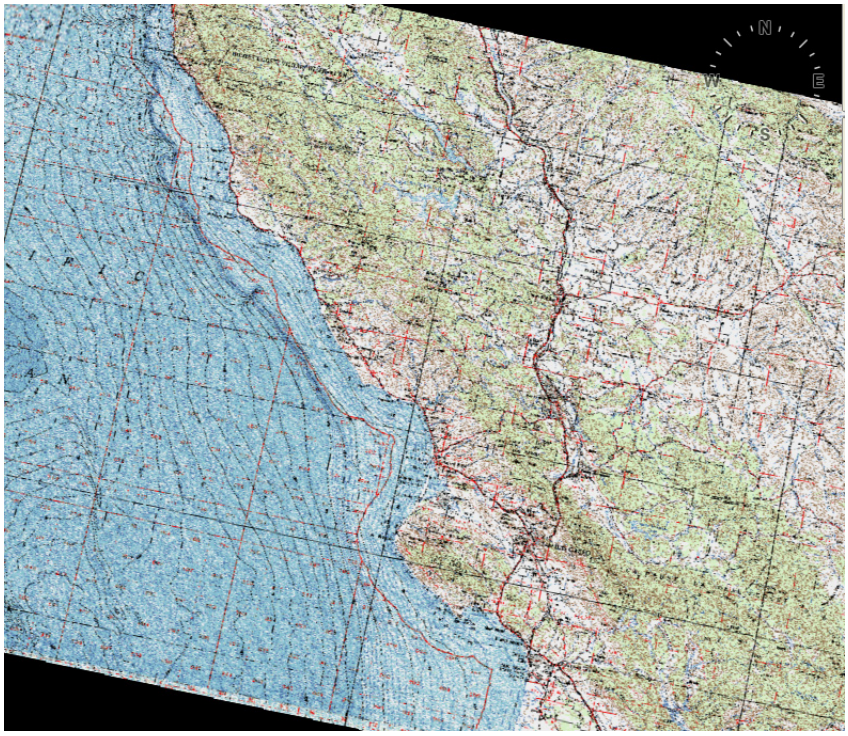


Figure C-21. *SanLuisObispo.earth* close view

7. Choose **File** → **Save Terrain**. The Save Terrain dialog box opens.
8. Type a name, such as mySanLuisObispoEarth. (Do not name it SanLuisObispo or you will overwrite the terrain shipped with VR-Forces.)
9. Click Save.
10. Choose **File** → **Close Terrain**.
11. Open the *SanLuisObispo.mtf* terrain. Then open the *SanLuisObispoTopoGdal.mtf* terrain. Compare these terrains to each other and the one you just built.

C.3. Extracting Props from a Terrain

In the tutorial "Composing a Terrain," on page C-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 C-17).
2. Select the external references that you want to extract as props and extract them ("Extract the Props," on page C-18).
3. Optionally, change the model definitions ("Change the Model Definition for a Prop," on page C-20).

C.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.

C.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 C-22.



Figure C-22. 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 C-23). 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.

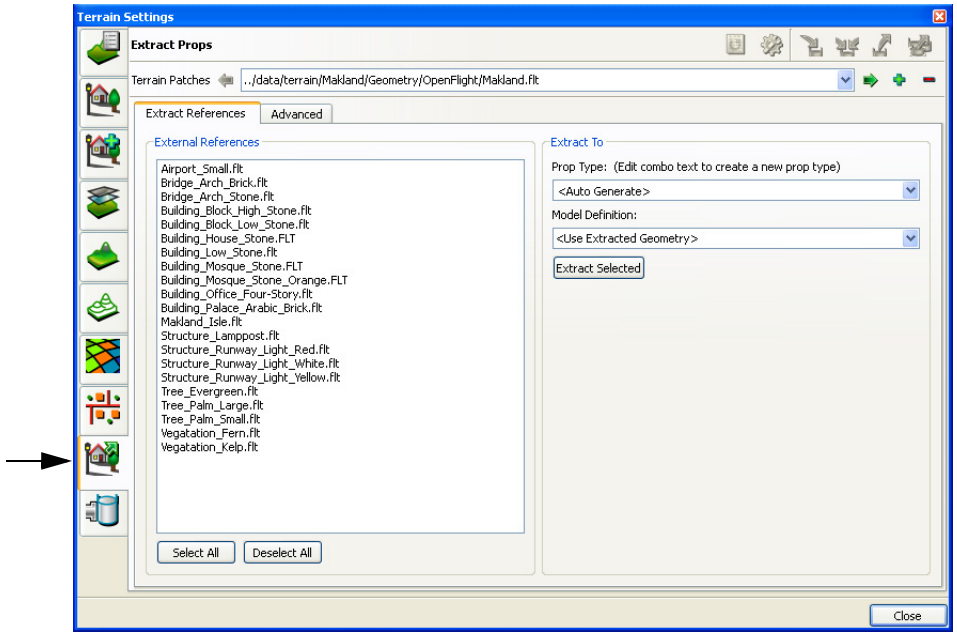


Figure C-23. Extract props page

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 C-24). 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 C-24. Selected props

C.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 C-25).

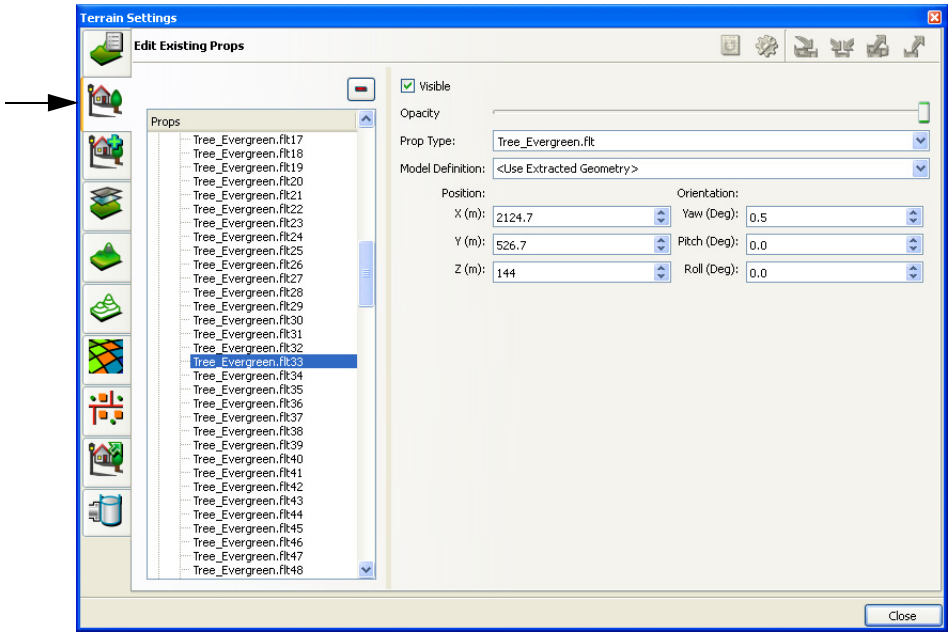


Figure C-25. Edit Existing Props page

4. In the Model Definition box, select a model definition from the drop-down list, such as TreesGiantRedwoodST. The tree changes to the new model definition (Figure C-26).



Figure C-26. 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.

C.4. Adding Feature Data to a GDB Terrain

The documentation for creating props focuses on adding props from terrain data files that already contain feature data. However, what if you do not have shapefiles or other feature data for your terrain? How would you add a SpeedTree or a building? The MÄK Terrain Database Tool, which is included with VR-Forces, lets you add feature data to a GDB terrain. You can then extract these features and map them to models as described in this manual.

This tutorial shows you how to add a tree point feature to a terrain and then add it as a feature layer. A companion tutorial explains how to extract the feature and map it to a Speed Tree model. (For that portion of the tutorial, please see "Extracting a Point Feature as a Prop (Adding a SpeedTree)," on page D-13.) The process for adding a building or some other object would be similar.

The general process is as follows:

1. Create a new GDB terrain in the TDB Tool or open an existing terrain ("Create a New GDB Terrain in the TDB Tool," on page C-22).
2. Add point features in the locations where you want them ("Add a Point Feature to the Terrain," on page C-25).
3. Save the GDB file.

4. In VR-Forces, add the GDB file as a feature layer ("Add the GDB Terrain to VR-Forces as a Feature Layer," on page C-28).

C.4.1. Create a New GDB Terrain in the TDB Tool

This step of the tutorial minimally describes the process for creating a new terrain in the TDB Tool. Please refer to *MAK Terrain Database Tool Users Guide* for details about creating projects, GDB files, and MTD files. This tutorial uses the Little Pond terrain.

To create a Little Pond GDB file and add feature data:

1. Start the TDB Tool. (Choose **Programs** → **MÄK Technologies** → VR-Forces **4.0.4** → **Tools** → **TDB Tool**. The TDB Tool opens (Figure C-27).)

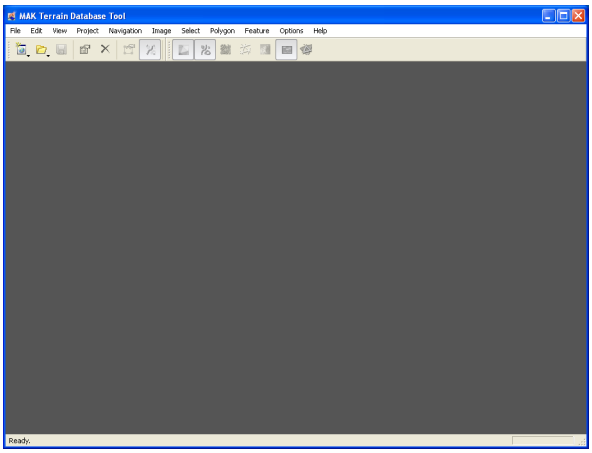


Figure C-27. TDB Tool window

2. Choose **File** → **New** → **Terrain Project**. The Utility Bar is added to the window (Figure C-28).

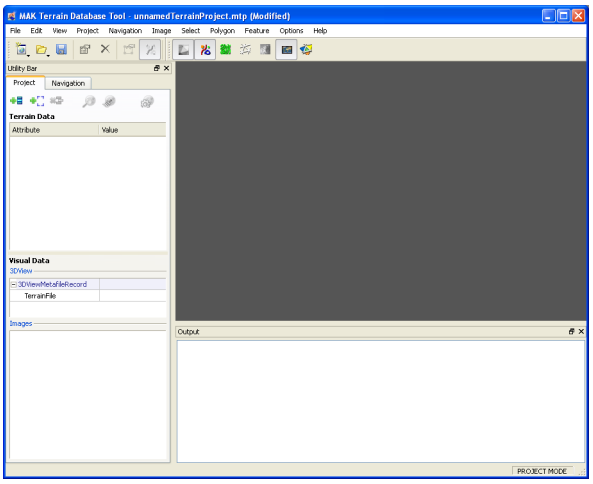


Figure C-28. New project

3. Choose **Project** → **Add Data From File**. The Add Data dialog box opens.
4. Navigate to *./data/terrain/LittlePond/Elevation* and choose *w072n042.dt1*.
5. Click Open. The data is added to the project. By default, the TDB Tool sets the projection for DTED files to UTM. However, the default for DTED in VR-Forces is Flat Earth, so you must change the projection.
6. In the Terrain Data panel on the Project tab, scroll through the Attribute list until you see the Projection attribute.
7. Click on the value (UTM). The value becomes editable (as a list).
8. In the list, select FlatEarth (Figure C-29).

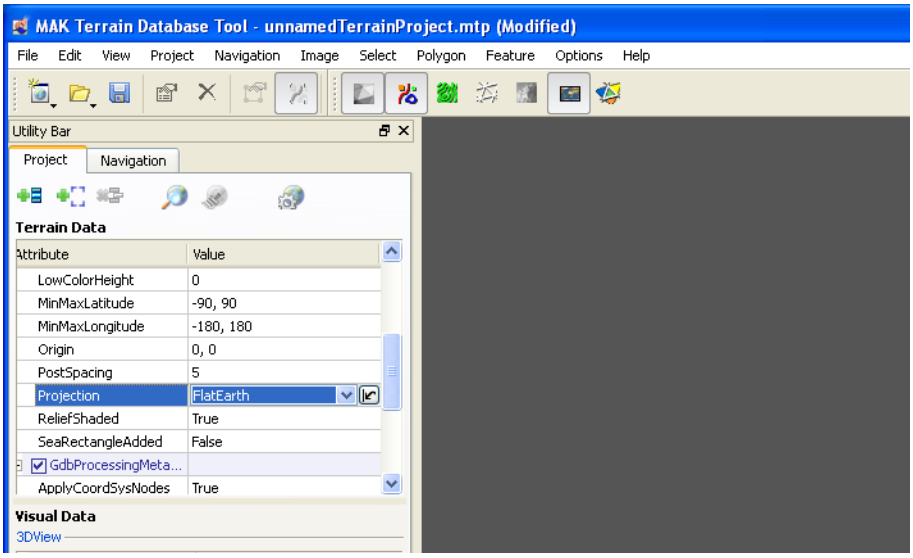


Figure C-29. Change the projection to FlatEarth

9. Choose **Project** → **Build Terrain**. The Build Terrain dialog box opens.
10. Type a name for your project, such as *myLittlePondFeatures*.
11. Click Save. The terrain is built and the TDB Tool asks you if you want to open the *myLittlePondFeatures.mtd* and *myLittlePondFeatures.gdb* files.
12. Click Yes.
13. Optionally, save the project when prompted. The terrain is displayed (Figure C-30).

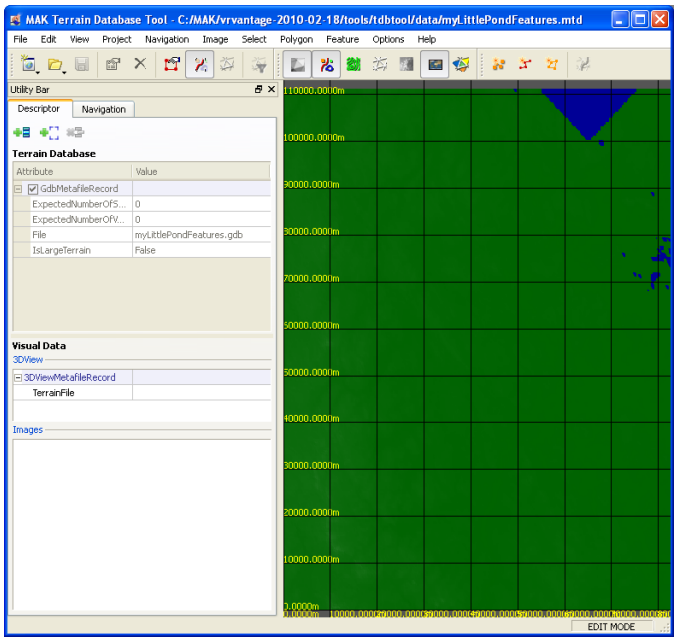


Figure C-30. New terrain

C.4.2. Add a Point Feature to the Terrain

This step of the tutorial minimally describes the process for adding point features in the TDB Tool. It uses the terrain created in the previous section. Please refer to *MAK Terrain Database Tool Users Guide* for details about creating projects, GDB files, and MTD files.

To add a point feature to the terrain:

1. On the Descriptor tab of the Utility bar, click Add Data. The Add Image Data dialog box opens.
2. Navigate to *./data/terrain/LittlePond/Imagery* and choose *LittlePond.tif*.
3. Click Open. The image is added to the terrain (Figure C-31). Its purpose is to give us an easily identified location on which to place our point features.

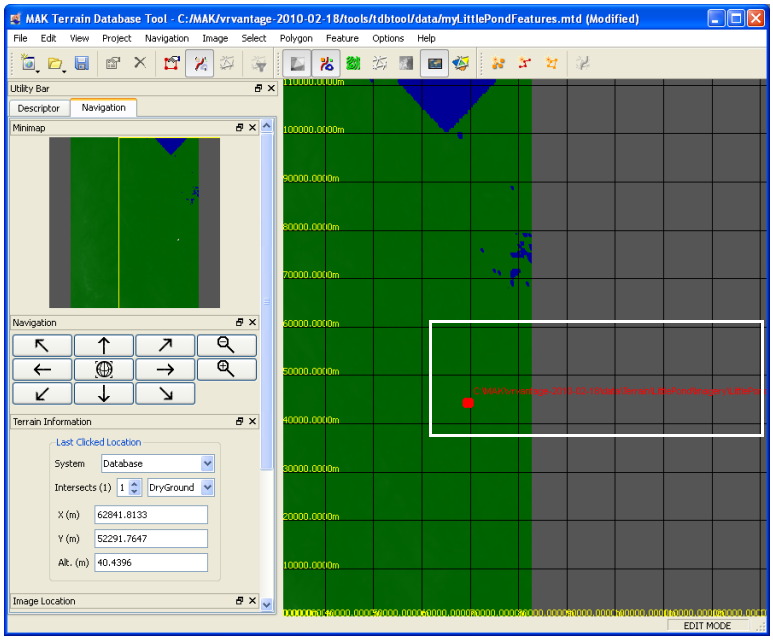


Figure C-31. Terrain with image added

4. Use the controls on the Navigation tab to zoom in on the image. Get close enough to see the roads easily, as in (Figure C-32).



Figure C-32. Zoomed-in view of image

5. Choose **Feature** → **Add Point Feature**. The cursor changes to a point and the Add Point Feature dialog box opens (Figure C-33).

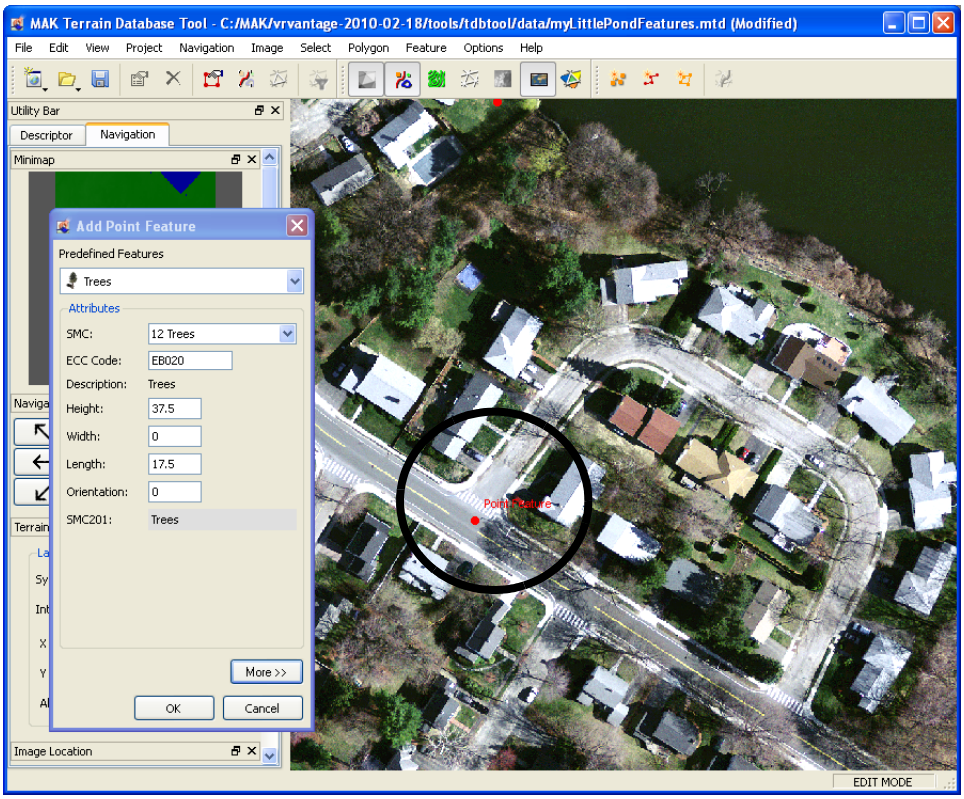


Figure C-33. Add Point Feature dialog box and new point

6. In the Predefined Features List, select Trees.
7. Click on the image in a place where your feature will be obvious, such as in the middle of the road (Figure C-33).
8. Click OK. The feature is added to the terrain (Figure C-34).

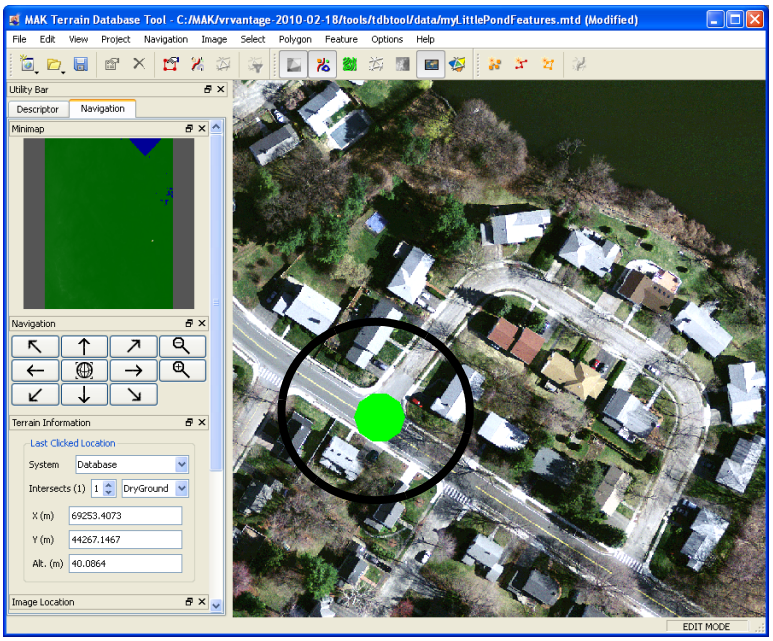


Figure C-34. Point feature added

9. Choose **File** → **Save**.
10. Close the TDB Tool.

C.4.3. Add the GDB Terrain to VR-Forces as a Feature Layer

In this step, we take the GDB terrain that we created in the previous steps and add it as a feature layer in VR-Forces.

To add a feature layer:

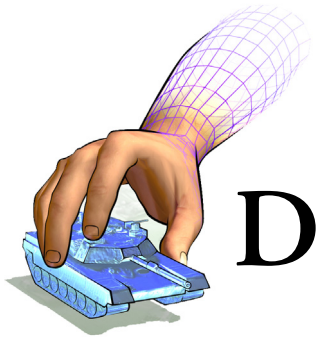
1. Start VR-Forces.
2. Change the observer mode to Stealth mode.
3. Open the *LittlePond.mtf* terrain.
4. Orient your view of the terrain so that you are viewing the area in which you created the point feature in the GDB file (Figure C-35). (If you open the Observer Views Panel and load the *LittlePond.osrx* saved views file, you can select the saved view and jump right to the inset.)



Figure C-35. 3D view of Little Pond

5. Open the Terrain Settings dialog box.
6. Select the Terrain Contents page.
7. Click the Add Feature Layer button. The Select Source File for Feature Layer dialog box opens on top of the Feature Layer Settings dialog box.
8. Select the file *./userData/terrains/myLittlePondFeatures.gdb*.
9. Click Open. The file is added to the list of feature layers.

To complete the tutorial by extracting the new point feature as a prop, please see "Extracting a Point Feature as a Prop (Adding a SpeedTree)," on page D-13.



D

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 appendix contains several tutorials that demonstrate how to combine various procedures to add models.

Adding a Model to VR-Forces D-2
- Create a Model Definition D-2
- Add an Element Definition D-5
- Add a Model Mapping D-8
- Test the Model Mapping D-10

Extracting a Point Feature as a Prop (Adding a SpeedTree) D-13

D.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.

The basic steps are:

1. Create a model definition ("Create a Model Definition," on page D-2).
2. Add an element definition ("Add an Element Definition," on page D-5).
3. Map the new model to an entity or object type enumeration ("Add a Model Mapping," on page D-8).
4. Test the new model ("Test the Model Mapping," on page D-10).

D.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 D-1).

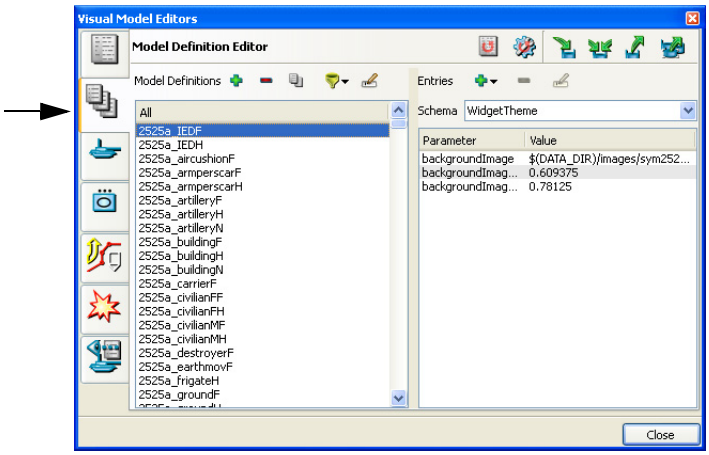


Figure D-1. Model Definition Editor page

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 12-11.)
4. Click the Add icon (+) next to the Model Definitions label. The Add Model Definition dialog box opens (Figure D-2).

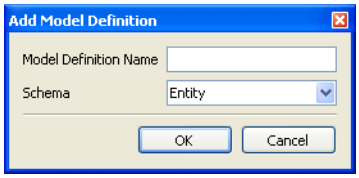


Figure D-2. Add Model Definition dialog box

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 drop-down list, select Entity. The dialog box now looks like Figure D-3.

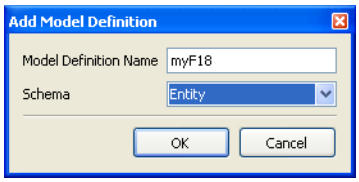


Figure D-3. New model definition

7. Click OK. The new definition is added to the list (Figure D-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.

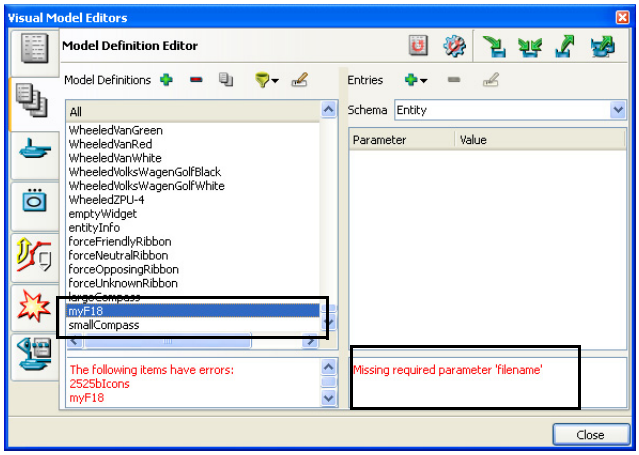


Figure D-4. New model definition in list

8. Select the new model definition.
9. Click the Add Parameter icon (). A list of parameters is displayed (Figure D-5).

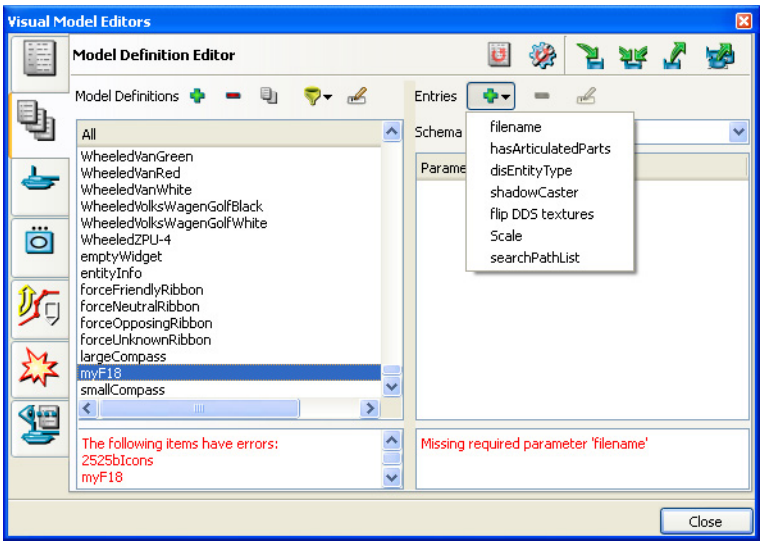


Figure D-5. List of parameters

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 D-6).

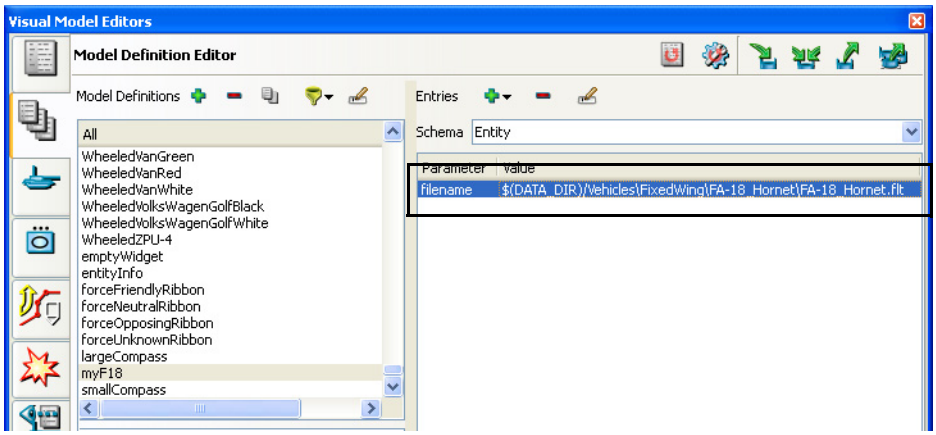


Figure D-6. Model definition parameter

D.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 D-1).

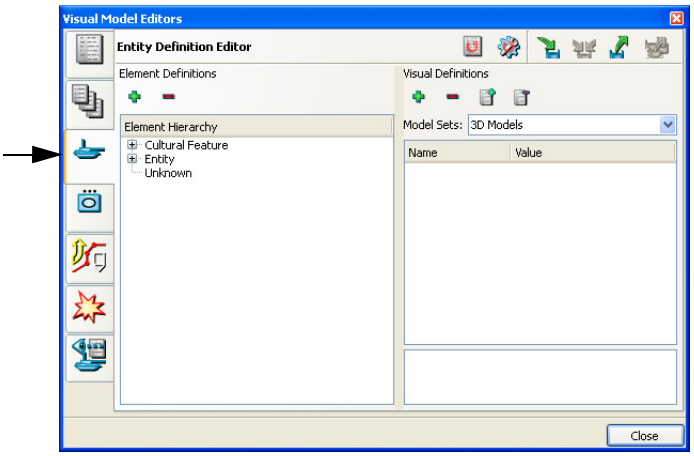


Figure D-7. Entity Definition Editor page

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 (+) 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 D-8). It has the same list of visualizers as the fixed wing parent level.

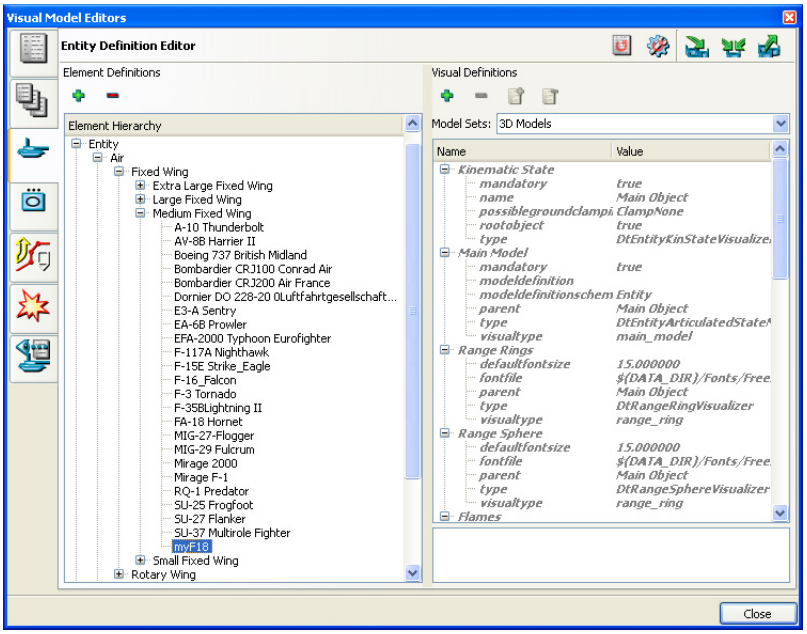


Figure D-8. New entity definition

9. Scroll through the list of visualizers until you find the Main Model Visualizer. This visualizer specifies the model definition that this element should use.

10. Click the Value column for the **modelDefinition** attribute to make it editable. The Choose Model Definitions dialog box opens (Figure D-9).

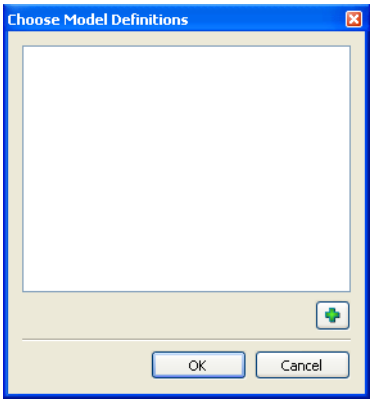


Figure D-9. Choose Model Definitions dialog box

11. Click the Add (+) icon. A drop-down list of model definitions is added to the dialog box Figure D-10().

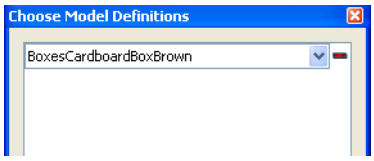


Figure D-10. Choose Model Definitions dialog box with list of definitions

12. Select the new model definition (myF18) and click OK. The model definition is now specified (Figure D-11).

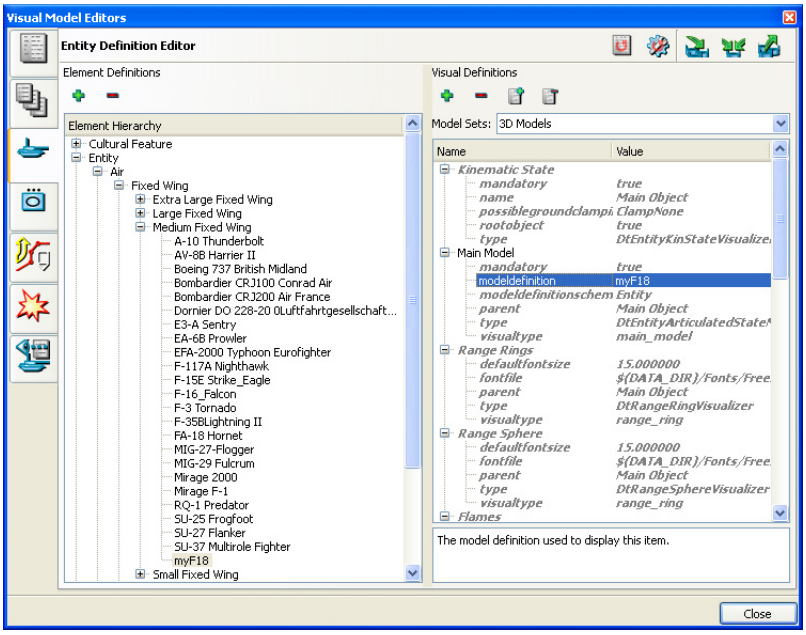


Figure D-11. Model Definition for Model Visualizer

13. Close the Visual Model Editors dialog box.

D.1.3. Add a Model Mapping

Now, you have to map the new model definition to an entity type enumeration.

To add a model mapping:

1. Choose **Settings** → **Entity Type Mappings**. The Entity Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure D-12).

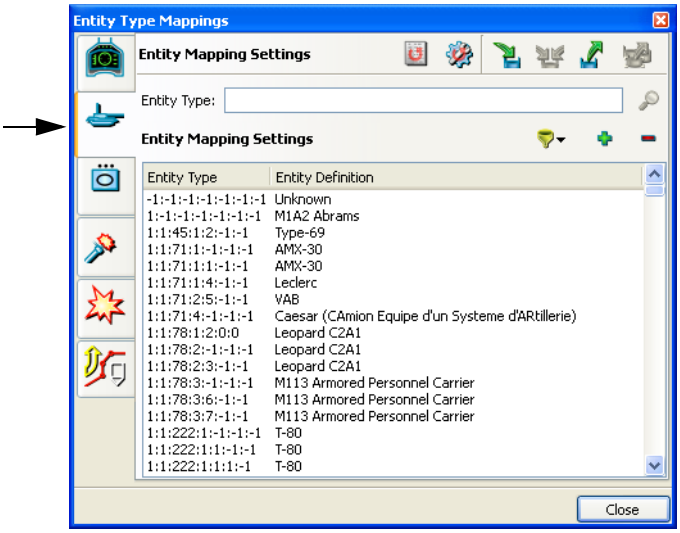


Figure D-12. Entity Mapping Settings page

3. Click the Filter icon ( ).
4. In the list, select **Entity** → **Air** → **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 icon ( ). The Create New Entity Type Model Mapping dialog box opens (Figure D-13). It lists a default entity type enumeration.

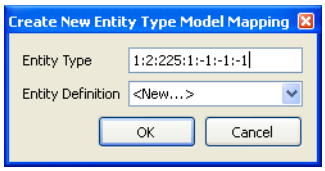


Figure D-13. Create New Entity Type Model Mapping dialog box

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 entity 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 entity type.

7. In the Entity Definition drop-down list, select the model definition that you added in the previous procedure (myF18) (Figure D-14).

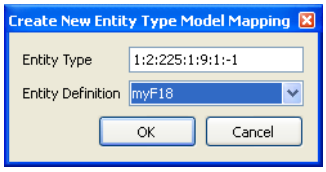


Figure D-14. Create new model mapping

8. Click OK.
9. The new model mapping is added to the list.

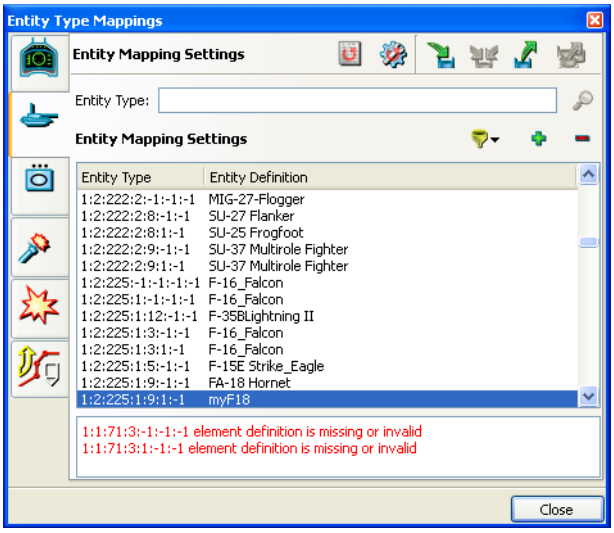


Figure D-15. New model mapping

D.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 D-16).

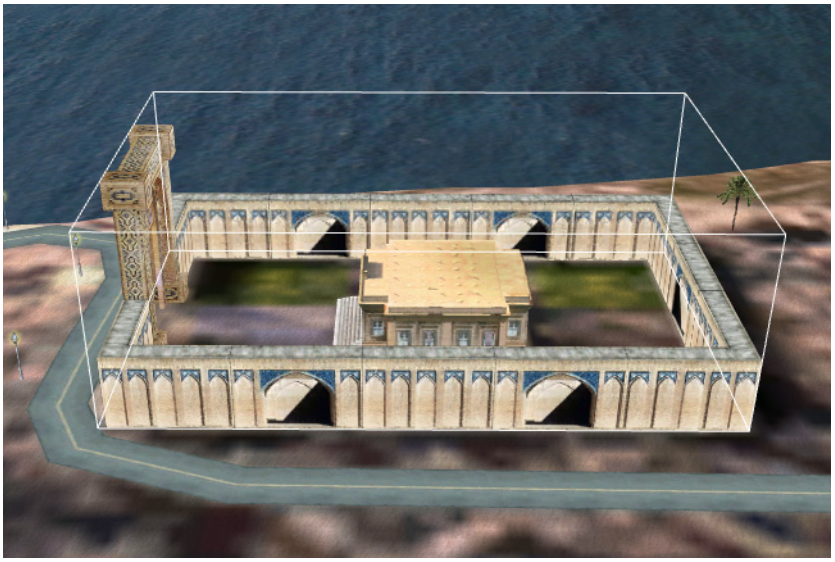


Figure D-16. Palace

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 D-17).

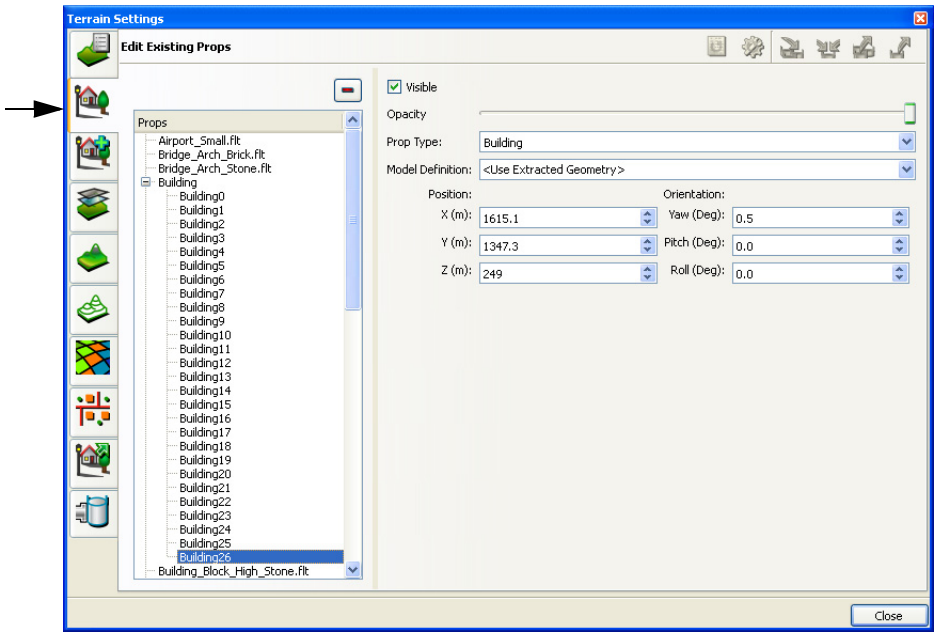


Figure D-17. Edit Existing Props page

6. In the Model Definition drop-down list, select the model definition that you added earlier in this tutorial (Figure D-18). The model of the palace is replaced with the model for the F/A-18 (Figure D-19). 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.

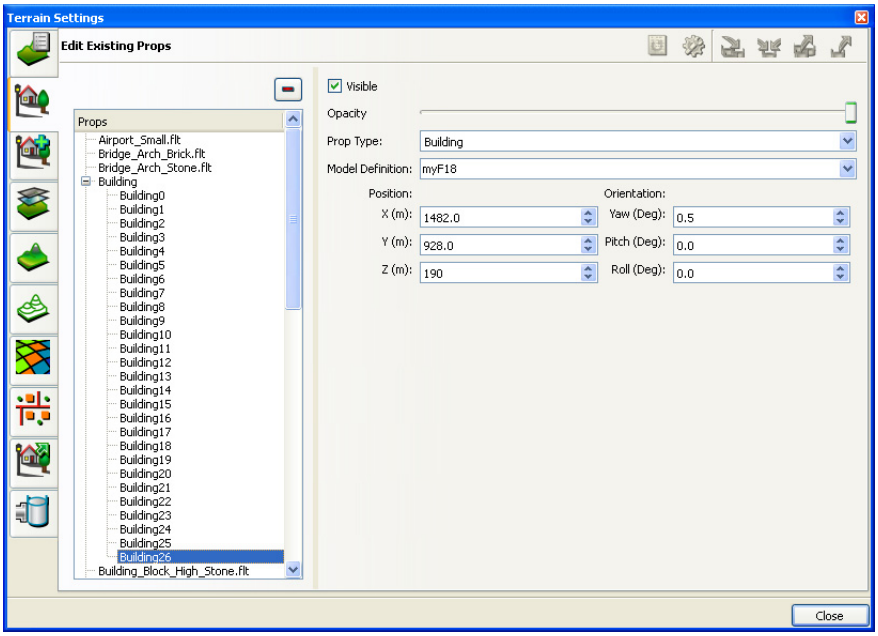


Figure D-18. Changed model definition

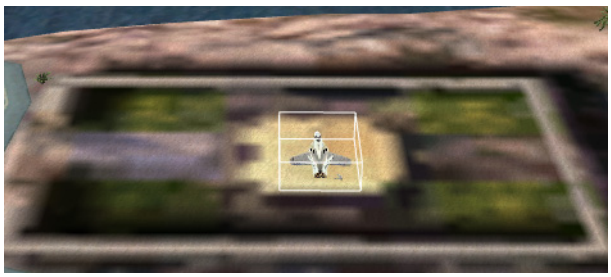


Figure D-19. F/A-18 replaces the palace

7. If you are going to use this terrain for simulations, change the model definition back to Use Extended Geometry.

D.2. Extracting a Point Feature as a Prop (Adding a SpeedTree)

In the tutorial "Adding Feature Data to a GDB Terrain," on page C-21, we explain how to add a point feature to a GDB terrain and add the terrain as a feature layer. In this tutorial, we extract the point feature and map it to a SpeedTree model. This tutorial assumes that you have completed the companion tutorial. Once the GDB file is added as a feature layer, you can extract the feature as a prop and map it to a model.

To extract a feature as a prop:

1. Load the LittlePond terrain.
2. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
3. Select the Add Props page.
4. Click the Add icon (+) next to the list of Feature Layers.
5. Select *myLittlePondFeatures.gdb*.
6. In the Build Rule group box, click the Add icon (+). The Build Field/Value Pair dialog box opens.
7. In the Field list, select DtFeatureName. The Value should be automatically set to Trees (Figure D-20).

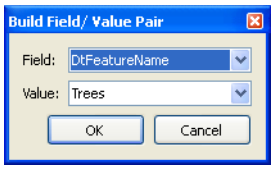


Figure D-20. Build Field/Value Pair dialog box

8. Click OK.
9. In the Prop Type list, select Tree.
10. In the Model Definition list, select the type of tree that you want to add, such as TreesGiantRedwoodSt (Figure D-21).

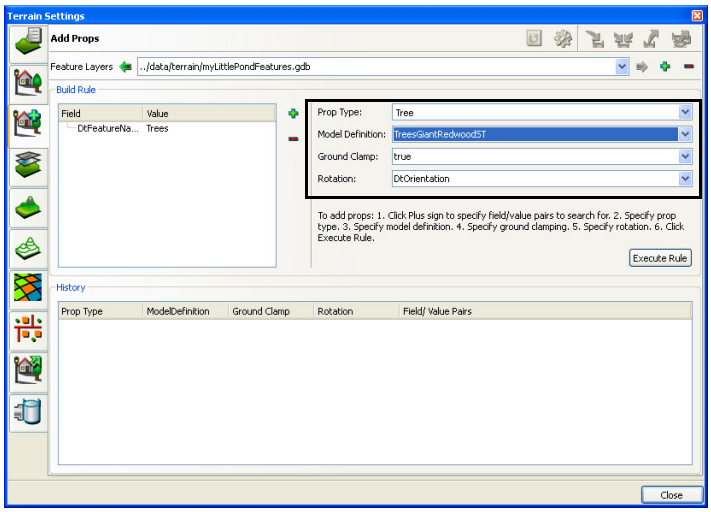
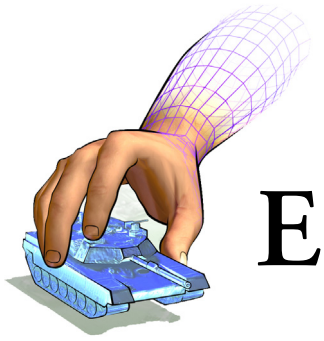


Figure D-21. Full build rule specified

11. Click Execute Rule. The tree is added to the scene where the point feature was located in the GDB file (Figure D-22).



Figure D-22. New tree in Little Pond



E

Creating and Editing Key Mappings

This appendix explains how to edit key mappings and create new key mappings.

Introduction E-2
The Key Mapping Editor E-2
Binary Key Mappings E-3
Editing a Key Map E-3
Adding a Key Mapping E-4
Changing a Key Mapping E-5
Deleting a Key Mapping E-6
Using Combined Key Mappings E-6
Filtering the Function List E-7
Creating a Key Map E-7
Deleting a Key Map E-7

E.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.

E.2. The Key Mapping Editor

The Key Mapping Editor (Figure E-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.

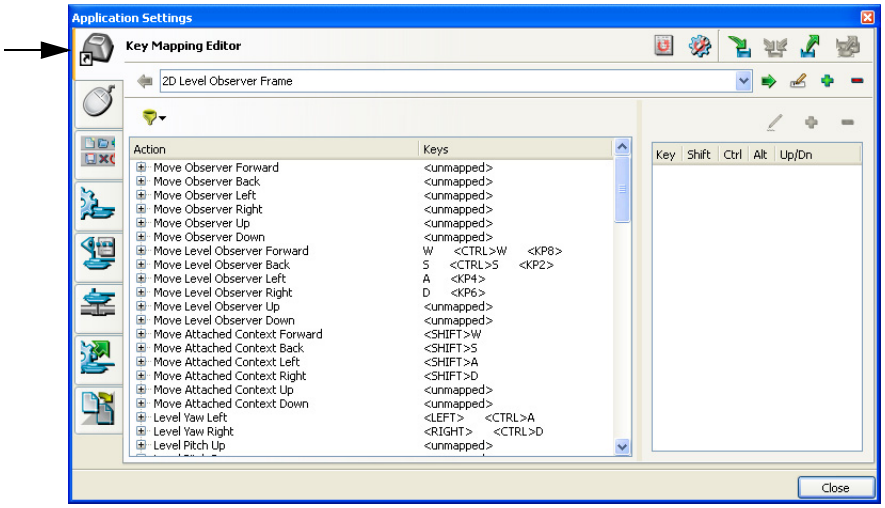


Figure E-1. Key Mapping Editor

E.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 entity.) 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 E-1.

E.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 E-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 E-4.
   - "Changing a Key Mapping," on page E-5.
   - "Deleting a Key Mapping," on page E-6.
5. Optionally, export the key map to a file.
6. Click OK.

E.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 E-7.
2. Select the function to which you want to add a key mapping.
3. Click the Add icon (+) above the Key Map window (Figure E-2). The Key Entry dialog box opens (Figure E-3).

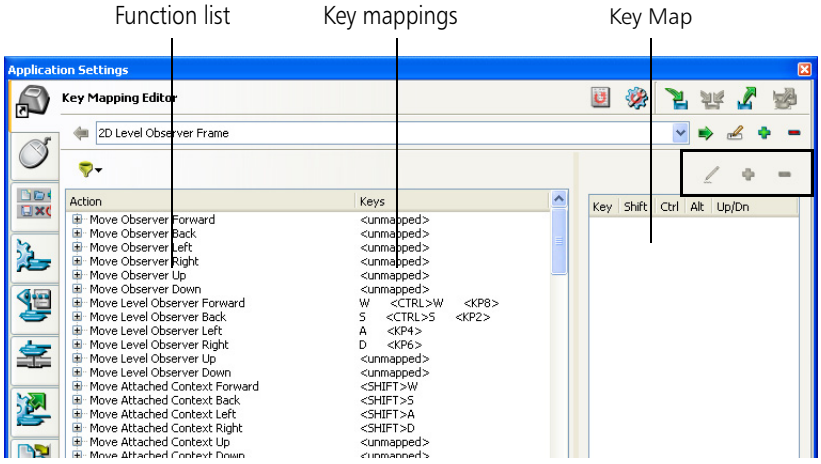


Figure E-2. Key Mapping Editor

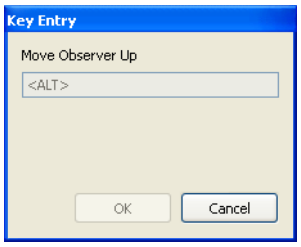


Figure E-3. Key Entry dialog box

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 E-6.) The mapping is added to the function and is listed in the Keys column of the function list and in the Key Map window.

E.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 E-2).
2. In the Key Map window, select the key mapping that you want to change.
3. Click the Edit Key Mapping icon ( ). The Key Entry dialog box opens (Figure E-4). It shows the current key mapping.

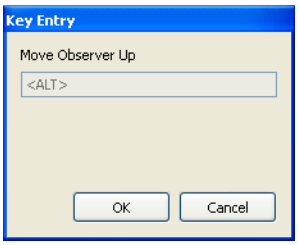


Figure E-4. Key Entry dialog box

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.

E.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 E-2).
2. In the Key Map window, select the key mapping that you want to delete.
3. Click the Delete icon (−) above the Key Map window. The mapping is deleted. The function list is updated.

E.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 entity. The **`Space Bar`** resets the observer view to the default view. Suppose that whenever you detach from an entity, 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 entity 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 E-5). It lists the functions that already use this key mapping.

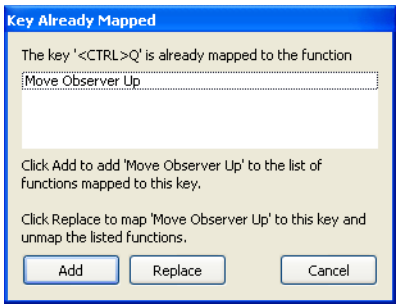


Figure E-5. Key Already Mapped dialog box

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.

E.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 icon ( ) and select an option on the list.

E.6. Creating a Key Map

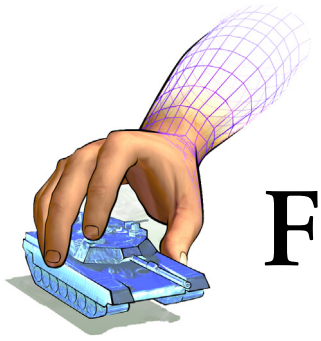
To create a key map:

1. Choose **Settings** → **Application**.The Application Settings dialog box opens.
2. Select the Keyboard Mappings page (Figure E-1).
3. Click the Add icon ( ). 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 E-4.

E.7. Deleting a Key Map

To delete a key map

1. Choose **Settings** → **Application**.The Application Settings dialog box opens.
2. Select the Keyboard Mappings page (Figure E-1).
3. In the key mapping list, select the key mapping that you want to delete.
4. Click the Delete icon ( ).
5. Confirm that you want to delete the selected key map.



F

The WRM Specification

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 appendix 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.)

i Although the WRM specification was developed for DIS, it is used in the HLA RPR FOM.

Introduction F-2
The OpenFlight File Format F-3
- Node Name and Comment Fields F-3
- External References and Instancing F-3

Modeling for DIS Interoperability F-4
- The DIS Attribute Lexicon (DAL) F-4
- DAL Keywords F-5
- Model Coordinate Systems F-6

Articulated Parts F-8
Entity Appearances F-9
Animation F-12
DIS Keyword Values F-12

F.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 appendix 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 appendix 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://local.wasp.uwa.edu.au/~pbourke/dataformats/wrm/.

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.

F.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.

F.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.

F.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.

F.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.

F.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.

F.3.2. DAL Keywords

The WRM Spec supports the following keywords:

- `articulated_part` *`part`*. Identifies the moveable 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 F-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 F-2.

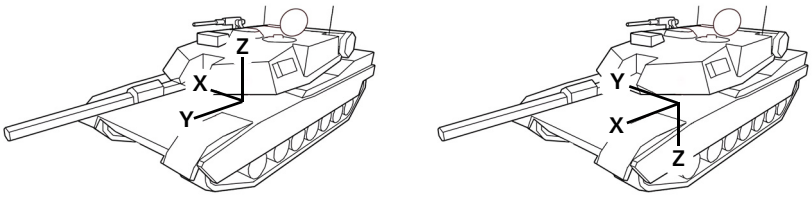
- `state` *`state`*,*`state`*... . Specifies the state of the appearance attribute with a DIS appearance state code. This keyword is placed in a Group node just below the switch node. For a list of supported values, please see Table F-2.

F.3.3. Model Coordinate Systems

OpenSceneGraph (OSG) and the DIS Standard use different right-hand coordinate systems. Table F-1 describes the two systems. Figure F-1 illustrates them.

Table F-1: Model coordinate systems

Coordinate	OpenSceneGraph	DIS
X	Right	Forward
Y	Forward	Right
Z	Up	Down



OSG coordinate system DIS coordinate system

Figure F-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.

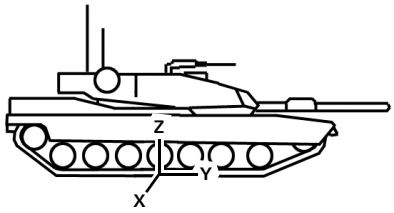
Location of the Origin for Models

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.

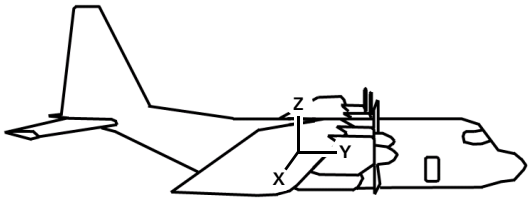
Figure F-2 illustrates the location of a model's origin for each platform domain.

Land



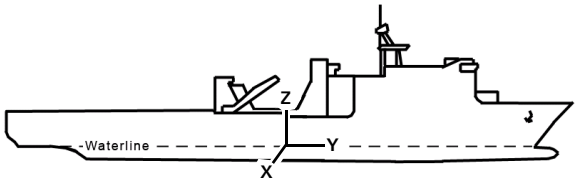
On the plane of ground contact, at the axis of yaw rotation.

Air



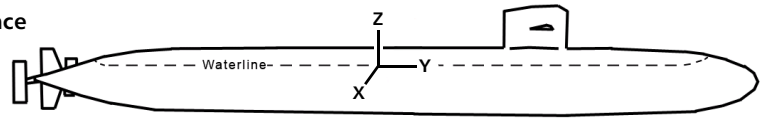
At the center of gravity.

Surface



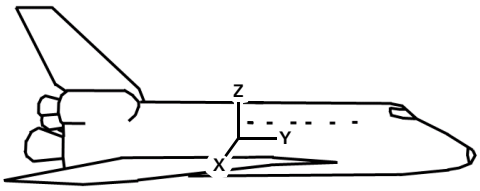
On the axis of the center of buoyancy, in the plane of the waterline for an unloaded ship.

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 F-2. Location of origin by domain

F.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.

F.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.) "Entity Appearances," on page F-9, lists the attributes you can use with VR-Forces.

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 F-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.

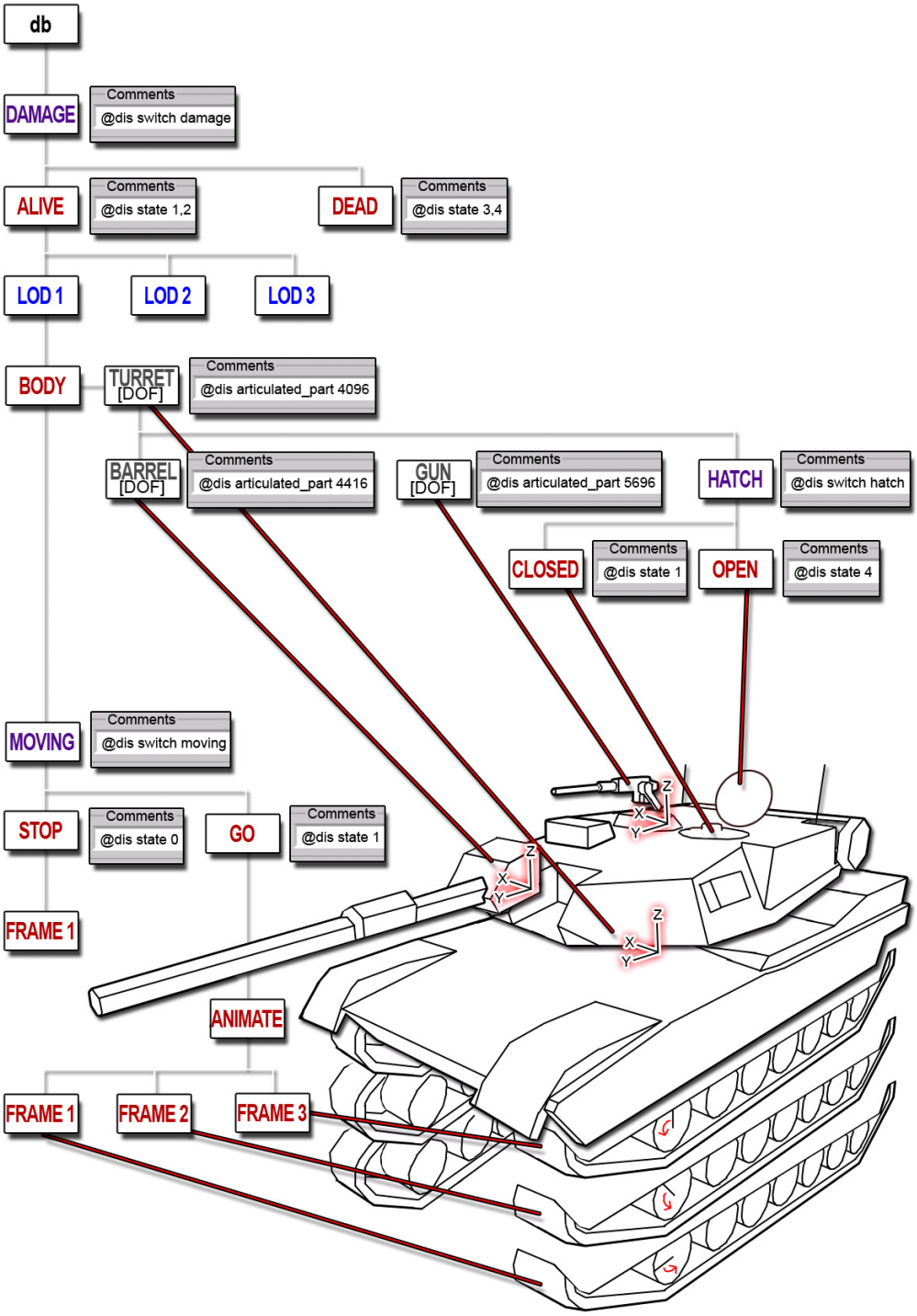


Figure F-3. 3D model structure

F.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.

F.7. DIS Keyword Values

Table F-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 F-2: DIS Appearance and States (Sheet 1 of 5)

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

Table F-2: DIS Appearance and States (Sheet 2 of 5)

Values	Description	State/ Description	Bit
compliance	Compliance of lifeform.	0 = Lifeform None 1 = Detained 2 = Surrender 3 = Using Fists 4 = Verbal Abuse 1 5 = Verbal Abuse 2 6 = Verbal Abuse 3 7 = Passive Resistance 1 8 = Passive Resistance 2 9 = Passive Resistance 3 10 = Non-Lethal Weapon 1 11 = Non-Lethal Weapon 2 12 = Non-Lethal Weapon 3 13 = Non-Lethal Weapon 4 14 = Non-Lethal Weapon 5 15 = Non-Lethal Weapon 6	
concealed	Type of concealment.	0 = Not Concealed 1 = In Concealed Position	19
damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed	3-4
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

Table F-2: DIS Appearance and States (Sheet 3 of 5)

Values	Description	State/ Description	Bit
flaming	State of flames are rising from an entity.	0 = No 1 = Yes	15
flashlight	State of lights.	0 = Off 1 = On	
formation_lights	Describes whether formation lights are on or off.	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	
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

Table F-2: DIS Appearance and States (Sheet 4 of 5)

Values	Description	State/ Description	Bit
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
running_lights	State of running lights.	0 = Off 1 = On	12
smoke	State or location of smoke and vapor emanating.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume	5-6
spot_lights	Describes whether spot-lights are on or off.	0 = Off 1 = On	28

Table F-2: DIS Appearance and States (Sheet 5 of 5)

Values	Description	State/ Description	Bit
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 F-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 F-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.
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.

Table F-3: DIS Articulations (Sheet 2 of 5)

Value	String	Description
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.
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.

Table F-3: DIS Articulations (Sheet 3 of 5)

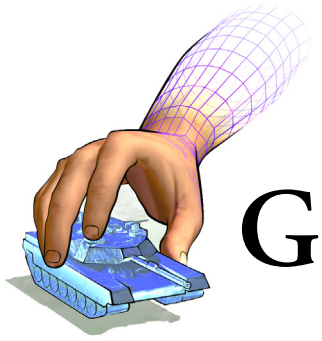
Value	String	Description
4960	primary_launcher_8	Position of primary launcher 8.
4992	primary_launcher_9	Position of primary launcher 9.
5024	primary_launcher_10	Position of primary launcher 10.
5376	primary_radar_1	Position of the primary radar 1.
5408	primary_radar_2	Position of the primary radar 2.
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.
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.

Table F-3: DIS Articulations (Sheet 4 of 5)

Value	String	Description
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.
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.

Table F-3: DIS Articulations (Sheet 5 of 5)

Value	String	Description
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.



G

Processing MetaFlight Files

To display MetaFlight files in VR-Forces, you must first process them. This appendix explains how to do so.

Processing MetaFlight Files for Use in VR-Forces G-2
Splitting Datasets into Individual MetaFlight Files G-2
Convert Virtual Texture Datasets to Tiled Textures G-2
The mftTool G-3
Using the mftTool to Process MetaFlight Files G-5
Convert Geometry Grid Datasets with Tiled Textures G-5
Convert Source Data into MEDF Format G-6
Create an MTF File for the MetaFlight Terrain G-6

G.1. Processing MetaFlight Files for Use in VR-Forces

Before you can load a MetaFlight file in a VR-Forces application, you must process it as described in this appendix. If you try to load an unprocessed MetaFlight file, VR-Forces displays an error message. When running our processing tools, a MetaFlight database which does not conform will cause an error message to be 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.

G.2. Splitting Datasets into Individual MetaFlight Files

Our tools work on one Geometry Grid Dataset 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 will identify which 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).

G.3. Convert 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. This is done with the mftTool. It creates the highest resolution textures possible from the virtual texture dataset and binds them to individual tiles.

G.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. 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 G-1 describes the command-line options.

Table G-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 will be 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 of this option is indicated if the terrain disappears when you zoom out, particularly in PVD mode. Increasing the range causes the terrain to page in at a greater distance.
(-f | --file) *filename*	The name of the MetaFlight file which contains the GGDS
(-g | --generateFiles)	Generate tiled textures from the Virtual Texture and generate tiles in MEDF format from the FLT tiles which use the new textures. Also an MFT file will be 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. Converting a named *YourTerrain.mft* would create a new MetaFlight file named *yourTerrain_medf.mft* in the same directory as *yourTerrain.mft*.If the generated tiled texture already exists the file will not be regenerated.
(-h | --help)	Displays usage information.
(-l | --level) *index*	The level to convert, using a zero-based index.
(-m | --missing) *string*	The missing texture placeholder.

Table G-1: mftTool command-line options

Option	Description
(-p | --threads) *threads*	Specifies the number of threads to run. Specifying zero will result 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 will be 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 which contains the VTDS. The arguments -s and -t are mutually exclusive.
(-u | --publishMftOnly)	Publish a new MetaFlight file which 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 named *YourTerrain.mft* would create a new MetaFlight file named *yourTerrain_medf.mft* in the same directory as *yourTerrain.mft*.* The -u and the -g arguments are mutually exclusive.
(-v | --version)	Display version information and exit.
(-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 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.

G.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 Flight 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.

G.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 11-7.

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 is described in "Compressing Model Files," on page 11-7.

The medfTool should never be run on a directory which contains GGDS tiles that use Virtual Textures.

G.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 our fast loading MEDF (MAK Encrypted Data Format) format. This should be done on terrains without virtual textures (it is done automatically when creating Geometry Grid Datasets (GGDS) from virtual texture datasets) and on cultural feature datasets. This should be done on all data files being loaded, but is required by 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
```

G.5. Create an MTF File for the MetaFlight Terrain

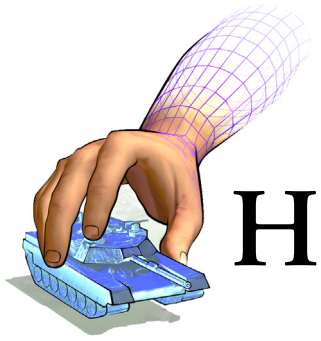
Once you have processed all of the MetaFlight files, create an MTF file for the terrain.



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.



H

Editing Earth Files

This chapter provides an introduction to the syntax of Earth files.

Earth Files H-2

A Simple Earth File H-2

H.1. Earth Files

The Earth file is an XML configuration file used to display terrain databases in osgEarth. The file format is documented at http://osgearth.org/wiki/Documentation. 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.

H.2. A Simple Earth File

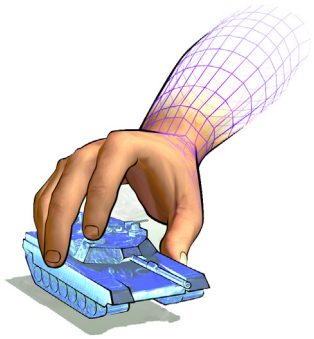
The *VR-TheWorldOnline.earth* file is an example of a simple Earth file. It has one image layer, one elevation layer, and five placemarks:

```
<!--
VR-TheWorld
VR-TheWorld base map includes 30m global imagery with 90m global
 elevation, along with high resolution insets.
-->
<map name="VR-TheWorld" type="globe" version="2">
  <image name="BaseImagery" driver="tms">
    <url>http://vrtw-devel.mak.local/tiles/1.0.0/1/</url>
  </image>
  <heightfield name="BaseElevation" driver="tms">
    <url>http://vrtw-devel.mak.local/tiles/1.0.0/2/</url>
  </heightfield>
  <external>
    <viewpoint height="0" lat="30.3229522705" long="-97.7872467041"
      name="Austin, TX" heading="0.0" pitch="-89.0"
      range="220731.221463"/>
    <viewpoint height="0" lat="31.4875030518" long="65.8475875855"
      name="Kandahar, Afghanistan" heading="0.0" pitch="-89.0"
      range="55182.8053662"/>
    <viewpoint height="0" lat="34.9411582947" long="69.259185791"
      name="Bagram Air Base, Afghanistan" heading="0.0" pitch="-89.0"
      range="27591.4026831"/>
    <viewpoint height="0" lat="42.3234558105" long="-71.0976791382"
      name="Boston, MA" heading="0.0" pitch="-89.0"
      range="55182.8053662"/>
    <viewpoint height="0" lat="38.893661499" long="-77.0114135742"
      name="Washington, DC" heading="0.0" pitch="-89.0"
      range="55182.8053662"/>
  </external>
</map>
```

Table H-1 describes the XML elements in the file and their attributes.

Table H-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 - globe or projected.
	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.
heightfield		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.
external		Specifies map data that is not part of a layer, such as placemarks.
	viewpoint	Specifies the location of the item, the location from which to view it, and its name.



Index

-- command-line option G-3
--col command-line option G-3
--encryptImages command-line option G-3
--extendLevelZero command-line option G-3
--file command-line option G-3
--generateFiles command-line option G-3
--help command-line option G-3
--ignore_rest command-line option G-3
--level command-line option G-3
--missing command-line option G-3
--noVsync command-line option 1-13
--publishMftOnly command-line option G-4
--regenerateAll command-line option G-3
--row command-line option G-4
--searchDir command-line option G-4
--subDir command-line option G-4
--textureSize command-line option G-4
--threads command-line option G-4
--version command-line option G-4
--vtFile command-line option G-4
-f command-line option G-3
-g command-line option G-3
-h command-line option G-3
-l command-line option G-3
-m command-line option G-3
-p command-line option G-4
-r command-line option G-4
-s command-line option G-4
-t command-line option G-4
-u command-line option G-4
-v command-line option G-4
-w command-line option G-4
-x command-line option G-4

Numerics

-1 (wildcard) 6-8
2525B icon 12-19
2D
  icon, editing 12-19
2D icon, configuring 12-19

A

-A command-line option G-3
--targetFrameRate command-line option 1-13
absolute, pathname 3-7
active stereo 9-20
add image attribute 10-7
Add Model Definition dialog box 12-9
Add Soil Type dialog box 5-12
Add Terrain Patch dialog box 10-5, C-3, C-17
Add Terrain Server dialog box C-15
Add Visualizer Definition Attribute window 12-17
adding
  attribute, visualizer 12-17
  category 4-44
  channel 9-5
  cockpit model 12-27
  component to object in OPD Editor 5-9
  DI-Guy character 2-8
  entities to Create menu A-5
  entity bitmap image 12-26
  entity image 12-26
  entity to object parameter database 6-17
  feature layer 10-10
  formation 4-29
  ingress or egress points 7-30
  key mapping E-4

adding (continued)
model mapping 11-3
new preconfigured aggregate 6-25
object in OPD Editor 5-8
parameter
model definition 12-11
schema 12-5
prop 10-17
from feature layer 10-19
sensor signature 7-5
sensor type 7-5
soil type parameter 5-12
subordinate object 5-12
subordinate to aggregate 4-26
system to entity 4-37
terrain patch to terrain 10-5
terrain server 10-13, C-13
visualizer 12-15
window 9-3
additional-opds parameter 3-6
addMulticastAddr parameter 2-6
advisory message, enabling and disabling 2-6
aggregate B-8
adding new preconfigured type 6-25
adding preconfigured to object parameter database 6-23
bounding box, Entity Editor 4-31
connection mode 2-9
creating 4-24
editing 4-24
editing preconfigured 6-24
formation
adding 4-29
automatic layout 4-33
copying 4-32
default 4-30
editing 4-24
manual layout 4-33
removing 4-29
renaming 4-29
snap to grid 4-34
formations 7-14
parameter-type 6-23
subordinate
adding 4-26
changing order 4-27
editing 4-28
removing 4-27
replacing 4-27
aggregate-formation-controller parameter 7-14
aggregateSpatialModelTickAsFastAsPossibleWhilePaused parameter 2-2
aggregateSpatialModelTickPeriod parameter 2-2
aggregateSpatialModelTickPeriodUsesRealTime parameter 2-2
aggregateSpatialModelTickVariance parameter 2-2
all, connection mode 2-9
all-tasks-exclusive parameter 7-25
alpha 12-34
altitude
observer, filtering entities by 1-7
ammo parameter 7-9
ammo select table 7-9
anaglyphic stereo 9-20
configuring 9-21
angle, damage 7-12
Angle of Attack 12-30
angle-of-incidence parameter 7-12
angleSegmentMax parameter 7-22
animation
DI-Guy, choosing 11-6
model F-12
AOA updater 12-30
appearance
DI-Guy, random 4-20
specifying for DI-Guy model 4-19
appIdRange parameter 2-6
application number 3-11
configuring 2-2
Application Settings dialog box 1-12
applying updated model definitions 12-11
appNumber parameter 2-2
arc, segment 7-22
arrow, forward and back 4-28, 4-40
articulated part F-2, F-8
assertOnBlockingTerrainCalls parameter 2-3
asynchronous I/O 2-7
using for performance 1-9
attribute
cockpit updater 12-27
destroyedsymbol 12-25
imagesymbolmap 12-24
modeldefinitionschema 12-23
parent 12-23
type 12-23
visualizer 12-15
adding 12-17
WidgetTheme 12-24
window 9-7
attributes, image 10-7

Auto Generate 10-17
auto-reorganize parameter 3-6
avoiding, objects 7-27

B

back arrow 4-28, 4-40
back-end
  console 2-3
  heartbeat 2-5
  mapping objects to 3-11
  multiple 3-11
  reloading OPD 2-4
ballistic gun, controller descriptor 7-9
batch file, configuring 2-3
batchScenarioFileName parameter 2-3
bead
  DOF F-8
  switch F-9
binary key mapping E-3
binding, system definition file variable 6-22
bitmap
  adding new 12-26
  displaying 10-6
  icon 12-21
blend image attribute 10-7
bounding box, aggregate 4-31
building
  adding to terrain C-21
  prop type 10-22
  terrain 10-5

C

-c command-line option G-3
cache
  clearing 10-25
  clearing model instancing 11-6
  model instance 11-6
caching
  file 10-24
  terrain server data 10-24
camera, position and orientation 9-19
Camera Orientation Offset 9-8
Camera Position Offset 9-8, 9-19
catastrophic-kill parameter 7-12
category
  adding 4-44
  changing 4-11
  changing echelon 6-26
category (continued)
  removing 4-44, 4-45
  renaming 4-44, 4-45
  undoing changes 4-45
category parameter 6-17, 6-26
cgfDispatcherReceivePort parameter 2-3
changing
  channel attributes 9-8
  component priority 5-10
  echelon ID category and echelon ID level 6-26
  entity
    category 4-11
    enumeration 4-14
    force 4-13
    name 4-15
    parameter 4-18
    view in OPD Editor 5-5
  field of view 9-14
  frame rate mode 3-6
  grid scale factor 4-34
  image display order 10-9
  key mapping E-5
  model definition, prop C-20
  prop
    model definition 10-23
    position or orientation 10-23
    type 10-22
  site ID 3-11
  subordinate order 4-27
  terrain database for scenario 3-8
  viewport 9-15
  window attributes 9-7
channel
  adding 9-5
  changing attributes 9-8
  keyword 9-8
  multiple 9-16
  projection resize policy 9-11
  property 9-8
  removing 9-6
Channel Name 9-8
ChannelKeyword parameter 12-30
character
  DI-Guy, adding 2-8
  specifying for DI-Guy model 4-19
character_animations_table.mtl 2-8
character_digest.xml 2-8
check box, Marker Labels Visible 4-38
child, element definition 12-13
clamp image attribute 10-7

clamp to border image attribute 10-7
clamp to edge image attribute 10-7
class
  DtSimComponentManager 6-10
  DtSimObjectNetInterface 6-10
  DtSimObjectStateRepository 6-10
  DtSimTaskManager 6-10
  DtVrfObject 6-6, 6-10
  DtVrfObjectBaseStateRepository 6-6
clearing
  file cache 10-25
  model instancing cache 11-6
clipping plane 12-32
  dynamic 9-10
clock mode 3-6
cockpit display 11-2, 12-27
  adding new 12-27
  element definition 12-13
  installing 12-28
  model definition, creating 12-29
  updater 12-27
    attribute 12-27
coefficient probability 7-11
collateral damage, probability table 7-12
collision avoidance, configuring 7-27
color
  represents sensor volume frequency 7-20
  sensor volume, pulse rate 7-20
combining, key mapping E-6
command-line option
  -- G-3
  -A G-3
  -c G-3
  --col G-3
  -e G-3
  --encryptImages G-3
  --extendLevelZero G-3
  -f G-3
  --file G-3
  -g G-3
  --generateFiles G-3
  -h G-3
  --help G-3
  --ignore_rest G-3
  -l G-3
  --level G-3
  -m G-3
  --missing G-3
  --noVsync 1-13
  -p G-4
command-line option (continued)
  --publishMftOnly G-4
  -r G-4
  --regenerateAll G-3
  --row G-4
  -s G-4
  --searchDir G-4
  --subDir G-4
  -t G-4
  --targetFrameRate 1-13
  --textureSize G-4
  --threads G-4
  -u G-4
  -v G-4
  --version G-4
  --vtFile G-4
  -w G-4
  -x G-4
comment, MTL parameter 2-10
comm-model-name parameter 7-19
commModelParams.mtl file 2-9, 8-3, 8-4
communication model 2-9, 7-19
  custom 2-10
  setting 8-3
  simple-radio-comm-model 2-9
component 6-9
  adding to object in OPD Editor 5-9
  attaching to remote entity 7-32
  creating connection in OPD Editor 5-10
  deleting in OPD Editor 5-7
  descriptor, elements of 6-14
  editing in OPD Editor 5-10
  entity 6-13
  occupancy-director-controller 7-30
  priority 5-10
  systems 6-18
  tuning 1-10
    performance 1-10
component attachment table 7-32
Component Manager B-5
Component-Attachment parameter 3-5
Component-Attachment-Table parameter 3-5
componentAttachmentTable.mtl 7-32
component-descriptor-type 6-14
component-manager parameter B-5, B-7
component-type 6-14
compressing, model and image files 11-7
compression, texture 10-25
conditional expression 3-9

configuration
  display engine
    loading 9-6
    saving 9-6
configuration file
  character_animations_table.mtl 2-8
  character_digest.xml 2-8
  commModelParams.mtl 2-9, 8-3, 8-4
  for object parameter database 4-4
  sysdef 6-18
  vrf.ent A-2, A-5
  vrfSim.mtl 2-2
  vrfSimSettings.xml 1-12
configuring
  2D icon 12-19
  back-end
    console 2-3
    heartbeat 2-5
  batch file 2-3
  communications model 2-9
  destination address 2-6
  DI-Guy 2-3, 2-5
  DI-Guy character and appearance 2-8, 4-19
  DIS 2-6
  drainInput 2-4
  drainInput for DIS 2-6
  drainInput for HLA 2-6
  Earth file H-2
  emitter 7-17
  exercise ID 2-6
  External Communications Model 8-4
  feature avoidance 7-28
  FED file name 2-5
  federation execution 2-5
    name 2-5
  FOM Mapper 2-5
  formations 7-14
  frequency of data sends 2-4
  heartbeat threshold 2-6
  HLA 2-5
  interest management 1-7
  joysticks 7-26
  map datum 2-5
  model definition 12-8
  multicast address 2-6
  multichannel display 9-16
  mutually exclusive tasks 7-25
  object avoidance 7-27
  obstruction avoidance 7-27
  path to OPD file 2-3
configuring (continued)
  port number 2-6
  projection resize policy 9-11
  pulse rate 7-20
  radio message model 2-9
  radios 7-19
  receive port 2-3
  scenario file to use 2-4
  schema 12-3
  sensor volume, segments 7-22
  sensor volumes 7-19
  sensors 7-3
  session ID 2-4
  simulation management message 2-5
  site ID 2-5
  start mode 2-5
  stereo
    anaglyphic 9-21
    polarized 9-22
  terrain database
    path 2-3
    to load 2-5
  terrain server 10-15
  time stamp order 2-6
  visual quality slider 1-13
  VR-Forces for external communications effects server 8-2
configuring slider 1-13
connection
  component 5-10
  configuring in system definition file 6-19
  sensor 7-3
connection mode 2-9
  aggregate 2-9
  all 2-9
  force 2-9
connection model, external 2-9
console, configuring for back-end 2-3
constructing, terrain 10-5
control object B-7
  list of for scenario 3-12
  parameter-type 6-27
coordinates, UV 10-8
copying
  formation 4-32
  model definition 12-10
  schema 12-5
CPU time, giving up 1-13
Create Connection dialog box 5-10
Create Entry dialog box 5-8

Create menu A-2
  adding entity to A-5
  assigning or removing entity 4-15
  configuring A-2
  editing 4-40, A-4
    manually 4-41
  reorganizing A-4
  reverting 4-44
Create Menu Tree, OPD Editor 5-5
Create New Component dialog box 5-9
Create New Entity Type Model Mapping dialog box 11-3
Create Resource dialog box 5-11
create-entity-menu-item A-5
creating
  aggregate 4-24
  component connection 5-10
  element definition 12-14
  entity A-5
    from existing 4-39
    new 4-39
  key map E-7
  MEDF and MEIF files 11-7
  model, guidelines for 12-34
  model definition 12-8, 12-9, D-2
    cockpit display 12-29
  schema 12-3
  simulation model set 4-6, 4-7
  system definition file 6-19
  terrain 10-3
  user-defined formation 7-16
cultural-feature-param parameter 6-12, B-8
cultural-feature-state-repository 6-12
custom, communication model 2-10
customizing, object parameter database 6-28

D

daemon, running vrfSim as 2-3
daemon parameter 2-3
damage
  configuring entity to take 7-13
  probability table 7-12
damage actuator 7-12
damage file 4-4, 6-27
damage model 4-35, 4-36
  specifying 4-37
data
  caching terrain 10-24
  configuring frequency of sends 2-4
data (continued)
  input 6-13
data distribution management 1-6, 1-7
  See also DDM
database, object parameter 6-2
dataset, virtual and tiled G-2
DDM 1-6, 1-7
  region size 1-7
DDS texture 10-25
  flipping 10-5, 12-31
decal effects 11-2
decal image attribute 10-7
default
  formation 4-30
  value, parameter 3-12
DefaultEllipsoidArc, model definition 7-19
defaultHeartbeatThreshold parameter 2-6
defaultObjectTimeout parameter 2-6
defaultParameterDatabasePath parameter 2-3, 3-7
default.sms file 4-4, 4-7
defaultTerrainDatabasePath parameter 2-3, 3-7
definition
  element 12-13
  parameter 5-5
degrees of freedom F-8
deleting
  category 4-44, 4-45
  channel 9-6
  element definition 12-18
  entity from simulation model set 4-40
  key map E-7
  key mapping E-6
  model definition 12-10
  model mapping 11-5
  parameter
    model definition 12-13
    schema 12-7
  parameters and components in OPD Editor 5-7
  schema 12-4
  terrain, cached 10-25
  window 9-6
description, parameter 5-5
destAddrString parameter 2-6
destination address 2-6
destroyedsymbol, attribute 12-25
destroyFedExec parameter 2-5
destroying, federation execution 2-5
detonation, effects 11-2
dialog box
  Add Model Definition 12-9

dialog box (continued)
Add Soil Type 5-12
Add Terrain Patch 10-5, C-3, C-17
Add Terrain Server C-15
Application Settings 1-12
Create Connection 5-10
Create Entry 5-8
Create New Component 5-9
Create New Entity Type Model Mapping 11-3
Create Resource 5-11
Display Settings
Interest Management Settings page 1-7
Render Settings page 1-5
Duplicate Model Definition 12-10
Entity Type Mappings 11-2
New Scenario 3-3
Open Model Set 4-8
Performance Options, configuring slider 1-13
Performance options 1-12
Visual Model Editors 12-8
DI-Guy
adding new characters 2-8
choosing animation 11-6
configuring 2-3
character and appearance 2-8
configuring character and appearance 4-19
configuring use of 2-5
model definition 12-9
diGuyAnimationsFile parameter 2-3
diGuyCharacterDataFile parameter 2-3
DirectX 10-25
DIS
configuring 2-6
encoding information in OpenFlight format F-2
enumerations document 11-2
heartbeat, threshold 2-6
WRM specification F-1
DIS Enumeration Document 6-6
disabling
advisory messages 2-6
creation of an entity type 4-15
frame rate statistics 2-4
texture compression 10-25
display
field of view 9-14
perspective 9-14
display engine 9-2
configuration
loading 9-6
display engine (continued)
configuration
saving 9-6
window, adding 9-3
Display Settings dialog box
Interest Management Settings page 1-7
Render Settings page 1-5
displaying
performance statistics 1-4
raster maps 10-6
segment outlines 7-23
disPort parameter 2-6
DLL
cockpit display 12-27
installing cockpit display, installing 12-28
DOF bead F-8
doNotUseConsole parameter 2-3
drainInput
configuring 2-4
configuring for DIS 2-6
configuring for HLA 2-6
dr-allow-gui-overrides parameter 1-13
DTED
loading 10-4
support for 10-3
DtNotifyLevel parameter 2-3
DtSimComponentManager class 6-10
DtSimObjectNetInterface class 6-10
DtSimObjectStateRepository class 6-10
DtSimTaskManager class 6-10
DtVrfObject class 6-6, 6-10
DtVrfObjectBaseStateRepository class 6-6
duplicate, key mapping E-6
Duplicate Model Definition dialog box 12-10
dynamic clipping plane 9-10
Dynamic Near Far Planes attribute 9-8, 9-10

E

-e command-line option G-3
.earth, file 10-24
Earth file H-2
echelon, changing level 6-26
echelon ID, category and echelon ID level 6-26
echelon-level parameter 6-23, 6-26
editing
aggregate 4-24
aggregate subordinate 4-28
channel attributes 9-8
components in OPD Editor 5-10

editing (continued)
Create menu 4-40, 4-41, A-4
element definition 12-14
embarkation ingress and egress points 7-29
embarkation parameters 7-30
entity model 4-9
formation 4-24
frustum 9-14
icon, 2D 12-19
key maps E-2, E-3
model definition 12-8, 12-11
model mapping 11-4
object map file 3-11
parameter
model definition 12-12
schema 12-7
parameters 4-18, 5-5
parameters for multiple objects 5-7
plan file 3-8
preconfigured aggregate 6-24
scenario file 3-3
sensor 7-4
sensor signature 7-5
system 4-38
system properties 4-38
systems 5-13
terrain server configuration 10-15
visualizer 12-15
parameter 12-16
window attributes 9-7
effect, mapping 11-2
egress point 7-29, 7-30
egress-points parameter 7-29
electromagnetic emissions, segment angle 7-22
element, parent 12-15
element definition 12-13
adding visualizer 12-15
creating 12-14
deleting 12-18
editing 12-14
filtering list in Entity Type Mappings dialog box 11-5
saving 12-18
embarkation
editing ingress and egress points 7-29
occupancy-director-controller 7-29
slots 7-29
embarkation-slot-entry parameter 7-29
embarkation-slots parameter 7-29
embedded window 9-2
EM-Emission update message 7-21
emitter, configuring 7-17
Emitter Power 7-21
enabling
advisory messages 2-6
creation of an entity type 4-15
External Communications Model 8-4
frame rate statistics 2-4
radio transmitter 8-3
texture compression 10-25
entity
adding new to parameter database 6-17
adding to Prop Palette 12-8
appearance F-2
category, changing 4-11
changing
force 4-13
name 4-15
undoing changes 4-21
component, attaching to remote 7-32
components 6-13
configuring
damage by munitions 7-13
emitter 7-17
obstruction avoidance 7-27
creating from existing 4-39
creating in Entity Editor 4-39
creation, enabling and disabling 4-15
deleting from simulation model set 4-40
editing 4-3
element definition 12-13
filtering 1-6, 1-7
icon, adding new image 12-26
model, editing 4-9
models 4-4
storing list of 3-12
systems 4-35
adding 4-37
tick rate 1-10
type 11-2
enumeration 11-3
vrf.ent file A-2, A-5
Entity Editor 4-3
configuring DI-Guy character and appearance 4-19
force
adding 4-46
editing 4-47
removing 4-48
formation, bounding box 4-31

Entity Editor (continued)
managing forces 4-46
navigating edited entities 4-40
opening OPD Editor from 4-38
starting 4-5
entity enumeration, changing 4-14
Entity Image Symbol, visualizer 12-19, 12-21, 12-26
Entity List, OPD Editor 5-5
Entity Mapping Settings page 11-3
Entity Mappings Settings page 11-4, 11-5
Entity Palette, assigning or removing entity 4-15
Entity Resource Manager 7-9
Entity Tree, OPD Editor 5-5
entity type, changing 4-14
Entity Type Mappings dialog box 11-2
filtering element definition list 11-5
entity-type parameter 7-10, A-5
entVrfDataTimeThreshold parameter 2-4
enumeration 6-6, 11-3
document 11-2
entity, changing 4-14
object type B-9
Enumeration and Bit Encoded Values for Use with Protocols for DIS Applications 6-6
environment variable
MAK_GEOID_GRID 10-4
OpenSceneGraph, anaglyphic stereo 9-21
OSGEARTH_CACHE_PATH 10-24
execName parameter 2-5
exercise ID 2-6
exerciseId parameter 2-6
exercise-start-time parameter 3-6
external communication model 2-9
enabling 8-3
external communications effects server
configuring VR-Forces for 8-2
QualNet 8-5
External Communications Model
configuring 8-4
enabling 8-4
external communications model 8-3
external reference 12-34
extracting
prop C-17
props from terrain patch 10-17

F

far clipping plane 9-10
feature
adding to terrain C-21
avoiding 7-27
configuring avoidance of 7-28
extracting as prop 10-18
feature layer
adding 10-10
adding prop from 10-19
FED file, name 2-5
federate, name 2-5
federateName parameter 2-5
federation execution
destroying 2-5
name 2-5
fedFileName parameter 2-5
Feet Above Ground Level 12-30
Feet Above Mean Sea Level 12-30
FeetAGL updater 12-30
FeetMSL updater 12-30
FeetPerMinute updater 12-30
field of view 7-21, 9-14, 9-15
changing 9-14
file
cache, clearing 10-25
caching 10-24
compressed 11-7
default.sms 4-4
.earth 10-24
formation 7-15
hostility relationships 3-12
joyParam.dat 7-26
MTL 2-10
object map 3-11
order of battle 3-12
plan 3-8
scenario 3-2, 3-3, 3-7
simulation model set 4-4
vrf.ent A-2, A-5
vrfSim.mtl 2-2
vrfSim.opd 6-2
vrfSimSettings.xml 1-12
File Caching Settings page 10-24, 10-25, 11-6, 11-7
Filename parameter 12-29
filtering
element definition list 11-5
entities
by observer altitude 1-7
in scene 1-6
function list E-7

filtering (continued)
model definition list 12-11
firing effects 11-2
fixed-frame 3-6
Fixed-Frame Best-Effort 3-6
Fixed-Frame Run-To-Complete 3-6
fixed-frame-run-to-complete 3-6
fixed-wing-entity-param 6-12
fixed-wing-entity-param parameter B-8
fixed-wing-entity-state-repository 6-12
flaming effects 11-2
flipping, DDS textures 10-5, 12-31
FLT file, mapping to models 11-2
FOM Mapper
configuring 2-5
initialization string 2-6
fomMapperInitData parameter 2-6
fomMapperLib parameter 2-5
force
adding 4-46
color, changing 4-47
connection mode 2-9
editing 4-47
entity, changing 4-13
hostility, changing 4-47
name, changing 4-47
removing 4-48
forceParameterDbReload parameter 2-4
force-type parameter A-4, A-5
format, OpenFlight F-2
formation
adding 4-29
bounding box 4-31
configuring 7-14
copying 4-32
default 4-30
editing 4-24
expanding and collapsing in Formation Editor 4-30
file 4-4, 7-15
layout
automatic 4-33
manual 4-33
removing 4-29
renaming 4-29
snap to grid 4-34
user-defined 7-16
Formation Editor 4-29, 4-30, 4-32, 4-33
forward arrow 4-28, 4-40
frame, length of 3-6
frame rate 1-3, 1-13
mode, specifying 3-6
statistics, enabling and disabling 2-4
Frame Rate Monitor panel 1-4
frame-mode parameter 3-6
frame-time parameter 3-6
frequency, color of 7-20
frustum 9-8, 9-19
editing 9-14
full screen window 9-2
function, movement E-2
function list, filtering E-7

G

GDB terrain, shapefile 10-10
geocentric terrain 10-11
geographic DDM 1-7
geoid grid file 10-4
geometry 12-34
sharing 11-6
Geometry Grid Dataset G-2, G-6
GeoTIFF image 10-8
GGDS G-6
GL Studio 12-27
Global-Plan parameter 3-5
grid scale factor, changing 4-34
ground vehicle, configuring obstruction avoidance 7-27
ground-entity-state-repository 6-12
ground-vehicle-param 6-12
ground-vehicle-param parameter 6-6, B-8
guidelines 12-34
creating models 12-34
GuiScenarioSettings parameter 3-6
Gui-Terrain-Database parameter 3-5

H

header file*, objTypeEnums.h* B-9
Headlight updater 12-31
heartbeat
back-end, configuring 2-5
DIS 2-6
relationship to thresholds 1-12
hierarchy, element definition 12-13
hit probability file 6-27
hit probability table 7-10
HLA 1-6, 1-7
parameters 2-5

horizontal navigation E-2
hostility relationships file 3-12
HUD. *See* cockpit display
human-param parameter 6-12, B-8
human-state-repository 6-12

I

icon
  2525B 12-19
  2D 12-21
    bitmap 12-21
    editing 12-19
  bitmap, adding new 12-26
  configuring 2D 12-19
  image 12-21
    adding new 12-26
ID, site 2-5
image
  adding new 12-26
  attributes 10-7
  compressing 11-7
  display order 10-9
  displaying 10-6
  GeoTIFF 10-8
  icon 12-21
  texture coordinate system 10-8
imagesymbolmap, attribute 12-24
imagesymbolmap parameter 12-26
individual-lifeform-param 6-12
individual-lifeform-state-repository 6-12
information message 2-3
ingress point 7-29, 7-30
ingress-points parameter 7-29
inheritance, element definition 12-13
initialization string, FOM Mapper 2-6
input
  of data 6-13
  port and port group 6-14
interest management 1-6, 1-7
  configuring 1-7
Interest Management Settings page 1-7
IVE 10-15

J

joy-device-type parameter 7-26
joyParam.dat file 7-26
joy-rot-gain parameter 7-26
joy-rot-invert parameter 7-26
joy-rot-sensitivity parameter 7-26
joystick, configuring 7-26
joy-trans-gain parameter 7-26
joy-trans-invert parameter 7-26
joy-trans-sensitivity parameter 7-26

K

key map E-2
  creating E-7
  deleting E-7
  editing E-3
key mapping
  adding E-4
  binary E-3
  changing E-5
  combining E-6
  deleting E-6
Key Mapping Editor E-2
  adding mapping E-4
  binary mapping E-3
  changing mapping E-5
  deleting mapping E-6
  filtering function list E-7
keyboard E-2
  mappings E-2
  navigation E-2
key-names, location of configuration files 7-33
keyword, channel 9-8
KilometersPerHour updater 12-30
KnotsPerHour updater 12-30

L

label parameter A-5
lamppost, prop type 10-22
lateral-clearance parameter 7-27
layer
  feature
    adding 10-10
    adding prop from 10-19
layout
  formation
    automatic 4-33
    manual 4-33
leader-promotion-id parameter 7-15
level-of-detail 12-32
  See also LOD
lifeform
  appearance, random 4-20

lifeform (continued)
configuring DI-Guy appearance and character 2-8
limits, terrain 10-3
lisp B-2
list, props 10-21
listing, components in parameters database 6-16
load-entry parameter 7-29
load-entry-point parameter 7-29
loading
display engine configuration 9-6
DTED 10-4
MetaFlight 10-15
models 10-24
terrain 10-24
loadPluginIfVersionMismatch parameter 2-4
local
object 6-10
network interface 6-13
local-objects parameter 6-10, 6-11, B-4
location, window 9-7
location image attribute 10-8
LOD 12-32, 12-34
logFrameRateStatistics parameter 2-4

M

MÄK Encrypted Data Format G-6
MÄK Technologies LISP 2-10
MÄK Terrain Database Tool C-21
MAK_GEOID_GRID, environment variable 10-4
map datum, configuring 2-5
mapping
keys E-2
model, to entity type 11-2, 11-3
marker, system 4-38
Marker Labels Visible, check box 4-38
maxDrainInputReads parameter 2-6
maxDrainInputTime parameter 2-4
mcastTtl parameter 2-6
MEDF G-6
creating 11-7
medfTool G-5
MEIF, creating 11-7
memory 1-3
improving use of 11-6
menu, adding to Create A-5
message
frequency of 2-4
message (continued)
information and warning 2-3
sending and receiving 7-19
message-timeout parameter 8-4
metadata, in system definition file 6-18
MetaFlight
building efficient terrains 10-30
loading 10-15
processing G-2
mftTool G-3, G-5
migrating, object parameter database to new release 6-28
MilesPerHour updater 12-30
Military Symbol Icon, visualizer 12-19
minDrainInputTime parameter 2-6
minimum tick period 1-10
minimum tick period variance 1-10
min-tick-period parameter 1-10
min-tick-period-variance parameter 1-10
mirror image attribute 10-7
missile-entity-state-repository 6-12
missile-param 6-12
missile-param parameter B-8
model
adding, tutorial D-2
animation F-12
communication 2-9
compressing 11-7
creating 12-34
DI-Guy, choosing 11-6
editing model mapping 11-4
entity 4-4
editing 4-3, 4-9
file caching 10-24
flipping DDS textures 12-31
instance, caching 11-6
mapping
adding 11-3
deleting 11-5
editing 11-4
to FLT files 11-2
object 4-4
schema 12-8
set 12-14, 12-15
sharing resources 11-6
simulation 4-4
model definition 11-2
applying updated 12-11
copying 12-10

model definition (continued)
creating 12-8, 12-9, D-2
cockpit display 12-29
DefaultEllipsoidArc 7-19
deleting 12-10
DI-Guy character 12-9
editing 12-11
filtering list of 12-11
parameter
adding 12-11
deleting 12-13
editing 12-12
prop, changing 10-23, C-20
saving 12-13
schema 12-3
SpeedTree 12-9
Model Definition Editor page 12-8, 12-9, 12-10, 12-11, 12-12, 12-13, D-2
model instancing 11-6
cache, clearing 11-6
Model Specification for DIS Interoperability F-2
modeldefinitionschema, attribute 12-23
monitor, multiple 9-16
movement model 4-35
specifying 4-37
movement system 4-36, 6-18
moving object parameter B-8
moving-object-param 6-12
moving-object-param parameter 6-23, B-8
moving-object-param parameter type B-8
MTL file 2-10
vrfSim.mtl 2-2
multicast address, configuring 2-6
multicast time-to-live 2-6
multichannel, configuring 9-16
MultiGen-Paradigm, OpenFlight format F-2
multiple
back-ends 3-11
parameter values 12-16
munition, configuring entity for damage by 7-13
munition-type parameter 7-9
mutually exclusive task, configuring 7-25
mutually-exclusive-task-groups parameter 7-25

N

name
changing, entity 4-15
component 6-14
FED file 2-5
name (continued)
federate 2-5
navigating, aggregate hierarchy 4-28
navigation
function, editing key map E-3
key map E-2
keyboard E-2
near clipping plane 9-10
net-interface-min-tick-period parameter 1-11, B-5, B-7
net-interface-min-tick-period-variance parameter 1-11, B-5, B-7
network interface B-5
hierarchy 6-13
network-interface parameter B-5, B-7
new, terrain 10-3
New Scenario dialog box 3-3
notification level 2-3

O

object
adding, resources in OPD Editor 5-11
adding to Prop Palette 12-8
adding to terrain C-21
configuring, avoidance of 7-27
count 1-3
editing, multiple 5-7
list of 3-12
models 4-4
parameter B-3
default value 3-12
subcomponents 6-10
super-type 6-7
timeout threshold 2-6
type 6-6
hierarchy 6-8
object map file 3-11
object parameter database 4-4, 6-2
adding
component in OPD Editor 5-9
new entity 6-17
new object in OPD Editor 5-8
preconfigured aggregate 6-23, 6-25
soil type parameter 5-12
subordinate object 5-12
configuring path to 2-3
configuring sensors 7-3
loading in OPD Editor 5-4
migrating to new release 6-28
pathname interpretation 6-27

object parameter database (continued)
reloading 2-4
saving 5-14
system definition file 6-18
Object Parameter Database Editor. *See* OPD Editor
object type enumeration B-9
objectConsoleNotifyLevel parameter 2-4
Object-Map parameter 3-5
object-type parameter 6-8, B-3
objTypeEnums.h header file B-9
observer
altitude, filtering entities by 1-7
camera, changing 9-19
movement, functions E-2
Observer Name 9-8
obstacle, configuring avoidance of 7-27
obstruction, configuring avoidance of 7-27
occupancy-director-controller, role in embarkation 7-29
occupancy-director-controller component 7-30
OffsetX parameter 12-30
OffsetY parameter 12-30
OffsetZ parameter 12-30
opacity
prop 10-22
setting 10-22
OPD, reloading 2-4
OPD Editor 5-2
adding
new object 5-8
resources 5-11
adding subordinate object 5-12
changing
component priority 5-10
list views 5-5
compared to Entity Editor 4-3
connecting components 5-10
deleting parameters and components 5-7
editing
components 5-10
multiple objects 5-7
preconfigured aggregate 6-24
editing parameters 5-5
editing systems 6-18
loading object parameter database 5-4
opening from Entity Editor 4-38
restoring values 5-7
saving object parameter database 5-14
starting 5-3
OPD editor, ingress and egress points 7-30
OPE file 6-27
variable bindings 6-22
Open Model Set, dialog box 4-8
OpenFlight, format F-2
OpenGL 10-25
opening, simulation model set 4-8
OpenSceneGraph, environment variables for anaglyphic stereo 9-21
optimizing, performance 1-3
order of battle 3-12
file 3-12
default values in 3-12
Order-Of-Battle parameter 3-5
orientation
camera 9-19
prop, changing 10-23
osgEarth 10-11, H-2
adding terrain servers 10-13
OSGEARTH_CACHE_PATH, environment variable 10-24
outline
segment 7-23
sensor volume segment 7-23
output, port group 6-14
out-the-window view 9-16
Overlay parameter 3-5

P

paged terrain
flipping DDS textures 10-26
preprocessing 10-15
panel, Frame Rate Monitor 1-4
parameter
adding, to model definition 12-11
additional-opds 3-6
addMulticastAddr 2-6
aggregate-formation-controller 7-14
aggregateSpatialModelTickAsFastAsPossibleWhilePaused 2-2
aggregateSpatialModelTickPeriod 2-2
aggregateSpatialModelTickPeriodUsesRealTime 2-2
aggregateSpatialModelTickVariance 2-2
all-tasks-exclusive 7-25
ammo 7-9
angle-of-incidence 7-12
angleSegmentMax 7-22
appIdRange 2-6
appNumber 2-2

parameter (continued)
assertOnBlockingTerrainCalls 2-3
auto-reorganize 3-6
batchScenarioFileName 2-3
catastrophic-kill 7-12
category 6-17, 6-26
cgfDispactherReceivePort 2-3
ChannelKeyword 12-30
commenting out 2-10
comm-model-name 7-19
component list order 6-16
Component-Attachment 3-5
Component-Attachment-Table 3-5
component-manager B-5, B-7
cultural-feature-param 6-12, B-8
daemon 2-3
defaultHeartbeatThreshold 2-6
defaultObjectTimeout 2-6
defaultParameterDatabasePath 2-3, 3-7
defaultTerrainDatabasePath 2-3, 3-7
definitions 5-5
deleting, from model definition 12-13
deleting in OPD Editor 5-7
destAddrString 2-6
destroyFedExec 2-5
diGuyAnimationsFile 2-3
diGuyCharacterDataFile 2-3
DIS 2-6
disPort 2-6
doNotUseConsole 2-3
dr-allow-gui-overrides 1-13
DtNotifyLevel 2-3
echelon-level 6-23, 6-26
editing in Entity Editor 4-18
editing in OPD Editor 5-5
egress-points 7-29
embarkation, editing 7-30
embarkation-slot-entry 7-29
embarkation-slots 7-29
entity-type 7-10, A-5
entry B-2
entVrfDataTimeThreshold 2-4
execName 2-5
exerciseId 2-6
exercise-start-time 3-6
federateName 2-5
fedFileName 2-5
Filename 12-29
fixed-wing-entity-param B-8
fomMapperInitData 2-6

parameter (continued)
fomMapperLib 2-5
forceParameterDbReload 2-4
force-type A-4, A-5
frame-mode 3-6
frame-time 3-6
Global-Plan 3-5
ground-vehicle-param 6-6, B-8
GuiScenarioSettings 3-6
Gui-Terrain-Database 3-5
HLA 2-5
human-param 6-12, B-8
imagesymbolmap 12-26
ingress-points 7-29
joy-device-type 7-26
joy-rot-gain 7-26
joy-rot-invert 7-26
joy-rot-sensitivity 7-26
joy-trans-gain 7-26
joy-trans-invert 7-26
joy-trans-sensitivity 7-26
label A-5
lateral-clearance 7-27
leader-promotion-id 7-15
load-entry 7-29
load-entry-point 7-29
loadPluginIfVersionMismatch 2-4
local-objects 6-10, 6-11, B-4
logFrameRateStatistics 2-4
maxDrainInputReads 2-6
maxDrainInputTime 2-4
mcastTtl 2-6
message-timeout 8-4
minDrainInputTime 2-6
min-tick-period 1-10
min-tick-period-variance 1-10
missile-param B-8
model definition, editing 12-11, 12-12
moving-object-param 6-23, B-8
mutually-exclusive-task-groups 7-25
net-interface-min-tick-period 1-11, B-5, B-7
net-interface-min-tick-period-variance 1-11, B-5, B-7
network-interface B-5, B-7
object B-3
objectConsoleNotifyLevel 2-4
Object-Map 3-5
object-type 6-8, B-3
OffsetX 12-30
OffsetY 12-30

parameter (continued)
OffsetZ 12-30
Order-Of-Battle 3-5
Overlay 3-5
parameter-type 6-6, 6-9, 6-27, B-2, B-3
Perspective 12-29
PerspectiveHeight 12-29
PerspectiveWidth 12-29
Plan 3-5
plan-manager B-5, B-7
pluginsDirectory 2-4
position-offset 7-15
promotion-id 7-15
publishSimulationTimeForStealth 2-4
publish-transmitter 8-3
random-number-seed 3-6
range-determinant 7-10, 7-12
range-list 7-10
recvBufferSize 2-7
remote-objects 6-11, B-6
rotary-wing-entity-param B-8
rprFomVersion 2-6
RSOName 12-29
runInTimeManagementMode 2-6
scenario, file 3-4
scenario-data 3-6
ScenarioDescription 3-6
scenarioFileName 2-4
scenario-name 3-6
schema
adding 12-5
deleting 12-7
editing 12-7
SelectionGroups 3-5
sendBackendLogToNetwork 2-4
sendBufferSize 2-7
sendFedTime 2-6
sessionId 2-4
set-data-request-type 3-10
showLines 7-23
simMgmtPduProcessLevel 2-5
simulationModelSet 2-5, 4-7
Simulation-Model-Set-Files 3-6
siteId 2-5
startInRunMode 2-5
state-repository B-4, B-6
state-repository-min-tick-period 1-11
state-repository-min-tick-period-variance 1-11
statusUpdateHeartbeat 2-5
stopping-distance-factor 7-27
parameter (continued)
structure of entries B-2
substitute-message-size 8-4
surface-entity-param B-8
targetFrameRate 2-5
task-manager B-5, B-7
task-type 3-10
Terrain-Database 3-5
terrainDatabase 2-5
time-multiplier 3-5
useAbsoluteTimeStamps 2-6, 2-7
use-actual-message-size 8-4
useAdvisories 2-6
useAsyncIO 2-7
useCustomMapDatum 2-5
useDIGuy 2-5
value, restoring in OPD Editor 5-7
vector-obstruction-type 7-28
vector-type 7-28
visualizer, editing 12-16
vrf-base-object-param 6-27, B-3
warhead 7-9
parameters database, component list order 6-16
parameter-type
aggregate 6-23
control object 6-27
parameter-type parameter 6-6, 6-9, 6-27, B-2, B-3
parent
attribute 12-23
element 12-15
element definition 12-13
Parent Channel 9-8
pathname
interpreting by object parameter database 6-27
scenario files 3-7
pause mode 2-5
PDU
siman, configuring 2-5
performance 1-6, 1-7
asynchronous I/O 1-9
compressing texture 10-25
factors affecting 1-2
frame rate 1-13
improving 11-6
interest management 1-6
network, tuning 1-12
optimizing 1-2, 1-3
SpeedTree profile 12-33
statistics, displaying 1-4
terrain 10-3

performance (continued)
tuning
network interface 1-11
state repository 1-11
tuning visual for 3D application 1-12
visual 1-13
Performance Options dialog box 1-12, 1-13
Performance Options page 1-12
perspective, in display 9-14
Perspective parameter 12-29
PerspectiveHeight parameter 12-29
PerspectiveWidth parameter 12-29
Pitch updater 12-30
pixel 12-34
plan
file, editing 3-8
plan manager B-5
Plan parameter 3-5
plan-manager parameter B-5, B-7
pluginsDirectory parameter 2-4
point-to-point communications, configuring 2-6
polarized stereo 9-20
configuring 9-22
polygon 12-34
count 10-3
port 6-14
for scenario files 2-3
group 6-14
port number 2-6
position
camera 9-19
prop, changing 10-23
window attribute 9-7
position-offset parameter 7-15
precedence, components in parameters database 6-16
preprocessing
terrain, paged 10-15
priority
component, changing 5-10
probability table, direct and collateral damage 7-12
probability-coefficient list 7-10
process, object mappings to 3-11
processing, MetaFlight G-2
profile, SpeedTree performance 12-33
Projection Resize Policy 9-8
configuring 9-11
Projection Units 9-8
promotion-id parameter 7-15
prop
adding
from feature layer 10-19
to terrain 10-17, C-21
adding to Prop Palette 12-8
extracting 10-17, C-17
model definition, changing 10-23, C-20
opacity 10-22
opaque 10-22
transparent 10-22
type 10-22
viewing list of 10-21
Prop Palette, adding objects to 12-8
properties, editing system 4-38
property
channel 9-8
window 9-7
publishing, simulation time 2-4
publishSimulationTimeForStealth parameter 2-4
publish-transmitter parameter 8-3
pulse repetition rate 7-20

Q

QualNet
example scenario 8-6
using with VR-Forces 8-5
Qualnet, scenario 8-6

R

radio
configuring 7-19
message, configuring 2-9
radio transmitter, enabling 8-3
radius, sensor volume 7-21
random, DI-Guy appearance 4-20
random-number-seed parameter 3-6
range-determinant parameter 7-10, 7-12
range-list parameter 7-10
raster map
display order 10-9
displaying 10-6
receive buffer size 2-7
receiving, radio messages 7-19
recvBufferSize parameter 2-7
reference ellipsoid, configuring 2-5
refresh rate 1-13
region
DDM 1-7

region (continued)
update threshold 1-7
relative path 3-7
remote
object 6-10, B-6
network interface 6-13
remote attachment, key-names 7-33
remote entity, attaching components 7-32
remoteAttachment example scenario 7-32
remoteConfigurations directory 7-33
remote-configurations entry 7-33
remote-objects parameter 6-11, B-6
removing
category 4-44, 4-45
channel 9-6
force 4-48
formation 4-29
subordinate 4-27
system 4-38
window 9-6
renaming
category 4-44, 4-45
formation 4-29
terrain server 10-14, C-13
Render Settings page 1-5, 9-12
reorganizing, Create menu A-4
repeat image attribute 10-7
replace image attribute 10-7
replacing, subordinate 4-27
repository B-4
resizeable window 9-2
resizing, windows 9-11
resolution, tree 12-32
resource, adding to object 5-11
restoring, parameter values in OPD Editor 5-7
reusable software object 12-27
reverting
category changes 4-45
Create menu 4-44
Roll updater 12-30
rotary-wing-entity-param 6-12
rotary-wing-entity-param parameter B-8
rotary-wing-entity-state-repository 6-12
RPR FOM 11-2
specifying version 2-6
rprFomVersion parameter 2-6
RSO 12-27
RSOName parameter 12-29
run mode 2-5
runInTimeManagementMode parameter 2-6

S

saving
display engine configuration 9-6
element definition 12-18
model definition 12-13
object parameter database 5-14
schemas 12-3
terrain 10-4
scenario
file
editing 3-3
parameters 3-4
pathnames 3-7
specifying which to use 2-4
files 3-2
order of battle 3-12
parameter, **version** 3-6
Qualnet 8-6
remoteAttachment 7-32
temporary directory 3-13
terrain database 3-8
scenario-data parameter 3-6
ScenarioDescription parameter 3-6
scenarioFileName parameter 2-4
scenario-name parameter 3-6
schema 12-3
copying 12-5
creating 12-3
deleting 12-4
model definition 12-3
parameter
adding 12-5
deleting 12-7
editing 12-7
saving 12-3
visual definition 12-3
Schema Editor page 12-3, 12-4, 12-5, 12-7
Screen Number attribute 9-7
segment
angle, setting 7-22
arc size 7-22
edge 7-23
outline, displaying 7-23
sensor volume 7-22
SelectionGroups parameter 3-5
send buffer size 2-7
sendBackendLogToNetwork parameter 2-4
sendBufferSize parameter 2-7
sendFedTime parameter 2-6

sending
  data, configuring 2-4
  radio messages 7-19
sensor 7-3
  adding new 7-5
  attaching to remote entity 7-32
  configuring 7-3
  connecting 7-3
  signature 7-5
  signature-object-sensor 7-3
  system definition file, editing 7-4
sensor signature 4-36
sensor system 4-35
sensor volume
  color 7-20
  configuring 7-19
  outlines 7-23
  pulse rate 7-20
  radius 7-21
  segment angle 7-22
  segment size 7-22
sensors 6-18
server
  terrain, caching 10-24
session ID, configuring 2-4
sessionId parameter 2-4
set statement 3-9
set-data-request-type parameter 3-10
setting
  communication model 8-3
  prop opacity 10-22
  segment arc size 7-22
  sensor volume radius 7-21
sharing, resources 11-6
Show Performance Stats Overlay check box 1-6
showHuds 9-8
showLines parameter 7-23
signature, sensor 7-5
signature-object-sensor 7-3
simMgmtPduProcessLevel parameter 2-5
simple-radio-comm-model 2-9
simulation
  objects 1-3
  start mode 2-5
simulation management message, configuring 2-5
simulation model set 4-4
  creating 4-6
  deleting entity 4-40
  file 4-4
  new, creating 4-7
simulation model set (continued)
  opening 4-8
simulation time, publishing 2-4
simulationModelSet parameter 2-5, 4-7
Simulation-Model-Set-Files parameter 3-6
site ID 3-11
  configuring 2-5
siteId parameter 2-5
size
  region 1-7
  window 9-7
slider
  visual quality, configuring 1-13
slot, embarkation 7-29
snap to grid, formation 4-34
soil type, adding to object 5-12
specifying
  damage model 4-37
  default *default.sms* 4-7
  frame length 3-6
  frame rate mode 3-6
  movement model 4-37
  segment angle 7-22
specifying default 4-7
SpeedTree
  adding to terrain C-21
  configuring 12-32
  model definition 12-9
  performance profile 12-33
SpeedTree Settings page 12-32, 12-33
spot report, affect on performance 1-9
starting
  Entity Editor 4-5
  OPD Editor 5-3
startInRunMode parameter 2-5
state repository B-4
  hierarchy 6-11, 6-12
statement
  conditional 3-9
  set 3-9
state-repository parameter B-4, B-6
state-repository-min-tick-period parameter 1-11
state-repository-min-tick-period-variance parameter 1-11
statistics, displaying 1-4
statusUpdateHeartbeat parameter 2-5
Stealth, simulation clock 2-4
stereo display, types of 9-20
stopping-distance-factor parameter 7-27

subcomponent B-4
  classes 6-10
subordinate
  adding to aggregate 4-26
  editing 4-28
  order, changing 4-27
  removing from aggregate 4-27
  replacing 4-27
subordinate object, adding 5-12
substitute-message-size parameter 8-4
super-type 6-7
surface, damage specification 7-12
surface-entity-param 6-12
surface-entity-param parameter B-8
surface-entity-state-repository 6-12
switch bead F-9
synchronization, vertical 1-13
synchronizing, VR-Forces and Stealth simulation time 2-4
sysdef file 6-18
system 6-18
  adding to entity 4-37
  connection 6-19
  editing 4-38, 5-13
  editing properties 4-38
  entity 4-35
  marker in 3D model view 4-38
  removing 4-38
  sensor 7-3
system definition file 6-18
  configuring connections 6-19
  creating 6-19
  sensor 7-4
  variable bindings 6-22
system-definition block 6-20
Systems List, OPD Editor 5-5
Systems List tab 5-13

T

tactical graphics 11-2
  element definition 12-13
targetFrameRate parameter 2-5
target-type parameter 7-9
task, configuring mutually exclusive 7-25
task-manager parameter B-5, B-7
task-type parameter 3-10
TDB Tool C-21
temporary directory, scenario 3-13
terrain
  building 10-5
  creating 10-3
  deleting cached 10-25
  displaying raster image on 10-6
  DTED support 10-3
  file caching 10-24
  MetaFlight
    designing 10-30
    loading 10-15
    processing G-2
  paged
    flipping DDS textures 10-26
    preprocessing 10-15
  paging, MetaFlight 10-15
  patch
    adding to terrain 10-5
    extracting props from 10-17
  prop
    adding 10-17
    extracting C-17
    list 10-21
  saving 10-4
  server 10-11
    adding 10-13
    caching data 10-24
    editing 10-15
    editing configuration 10-15
  tutorial C-2
terrain database
  changing for a scenario 3-8
  configuring 2-5
  configuring path to 2-3
Terrain Database Tool C-21
terrain server
  adding C-13
  renaming 10-14, C-13
Terrain-Database parameter 3-5
terrainDatabase parameter 2-5
TerrraPage 10-15
texture 12-34
  compression, enabling and disabling 10-25
  coordinate system for image 10-8
  DDS 10-25
  flipping, paged terrain 10-26
  sharing 11-6
  tiled and virtual G-2
threshold
  data updates 2-4
  DIS heartbeat 2-6

threshold (continued)
object timeout 2-6
tick
rate, tuning 1-10
tiled texture G-2
Time Multiplier toolbar 3-5
time stamp order, configuring 2-6
time-multiplier parameter 3-5
timeout, object 2-6
trailing effects 11-2
transparency 12-34
prop, setting 10-22
tree
prop type 10-22
SpeedTree 12-32
TrueType 12-21
TrueType font, 2D icon 12-21
tuning
components 1-10
network performance 1-12
performance
component 1-10
network interface 1-11
state repository 1-11
visual performance 1-12
tutorial
add SpeedTree D-13
add terrain server C-13
adding feature as a prop C-21
adding model D-2
composing terrain C-2
extract feature as prop D-13
extract props C-17
type
attribute 12-23
prop 10-22

U

undoing
category changes 4-45
changes to Create menu 4-44
entity edits in Entity Editor 4-21
update, frequency of 2-4
updater 12-27
AOA 12-30
attribute 12-27
FeetAGL 12-30
FeetMSL 12-30
FeetPerMinute 12-30
updater (continued)
Headlight 12-31
KilometersPerHour 12-30
KnotsPerHour 12-30
MilesPerHour 12-30
Pitch 12-30
Roll 12-30
Value 12-31
Yaw 12-30
Use Extracted Geometry 10-17
useAbsoluteTimeStamps parameter 2-6, 2-7
use-actual-message-size parameter 8-4
useAdvisories parameter 2-6
useAsyncIO parameter 2-7
useCustomMapDatum parameter 2-5
USE-DEFAULT keyword 3-12
useDISGuy parameter 2-5
user-defined, formation 7-16
UV, coordinates 10-8

V

Value updater 12-31
variable
bindings 6-22
tick rate 1-10
variable frame mode 1-13
variable-frame 3-6
Variable-Frame Run-To-Complete 3-6
vector
feature, configuring avoidance of 7-28
vector-obstruction-type parameter 7-28
vector-type parameter 7-28
version, RPR FOM 2-6
version parameter, scenario file 3-6
video card, processor speed 1-3
viewing, list of props 10-21
Viewport 9-8
viewport, changing 9-15
virtual texture G-2
Virtual Texture Datasets G-2
visual definition, schema 12-3
Visual Model Editors dialog box 12-8
visual quality slider, configuring 1-13
visualizer 12-3, 12-13
adding 12-15
attribute 12-15
adding 12-17
editing 12-14, 12-15, 12-16
Entity Image Symbol 12-19, 12-21, 12-26

visualizer (continued)
Military Symbol Icon 12-19
vrf A-2
vrf-base-object-param parameter 6-27, B-3
vrf-base-object-param parameter type B-7
vrf.ent file A-2, A-5
editing A-4
reorganizing A-4
vrf-moving-object-state-repository 6-12
vrf-object-param 6-12
vrf-object-param parameter type B-7
vrf-object-state-repository 6-12
vrf-overlay-object-state-repository 6-12
vrfSim, running as daemon 2-3
vrfSim.mtl file 2-2, 4-7
vrfSim.opd, remote-configurations block 7-33
vrfSim.opd file 6-2
vrfSimSettings.xml file 1-12
VSync 1-13
VTDS G-2

W

warhead parameter 7-9
warning message 2-3
weapon system 4-35, 4-36
weapons 6-18
WidgetTheme, attribute 12-24
wildcard 6-8
window
Add Visualizer Definition Attribute 12-17
adding 9-3
attributes, editing 9-7
channel, adding 9-5
location, changing 9-7
property 9-7
removing 9-6
size, changing 9-7
types 9-2
Window Name attribute 9-7
Window Type attribute 9-7
WRM Entity Model Specification F-1, F-10

X-Y-Z

XML H-2
xtr file 3-12
Yaw updater 12-30
Z Far 9-8, 9-10
Z Near 9-8, 9-10



Link - Simulate - Visualize

68 MOULTON STREET CAMBRIDGE, MA 02138 617.876.8085 www.mak.com