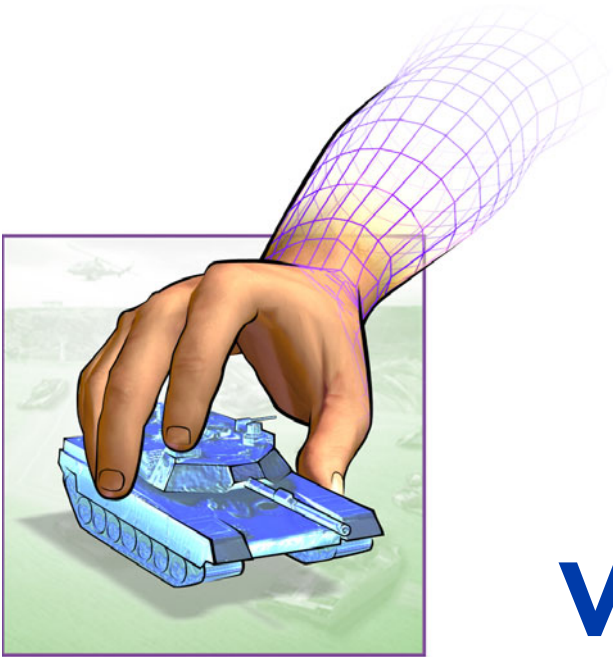
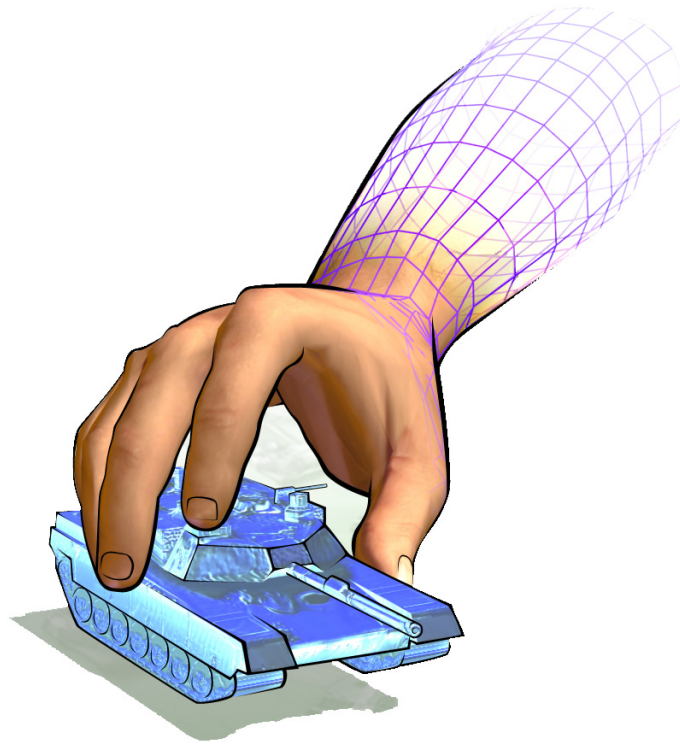




VR-Forces

Configuration Guide



VR-Forces



Configuration Guide

Copyright © 2011 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.

All other trademarks are owned by their respective companies.

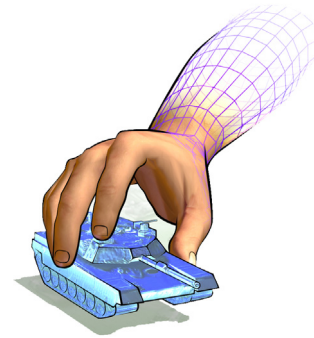
For third-party license information, please see “Third Party Licenses,” on page xx.

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-4.0-7-110117



Contents

Preface

How This Manual is Organized	xiii
VR-Forces Documentation.....	xv
MÄK Products	xvi
How to Contact Us	xviii
Document Conventions	xix
Hardware and Operating System Naming Conventions	xx
Mouse Button Naming Conventions.....	xx
Third Party Licenses	xx
Boost License.....	xx
libXML and libICONV	xxi
pThreads Library.....	xxi
EHS, PCRE, and PME	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	1-6
1.4.2. Configuring Interest Management	1-7
1.4.3. Filtering Entities by Observer Altitude	1-8
1.5. Miscellaneous Performance Configuration Options	1-8
1.5.1. Limiting Use of Spot Reports	1-8
1.5.2. Configuring Intervisibility Intersection Testing	1-8
1.5.3. Using Asynchronous I/O for DIS Exercises	1-8

1.6. Setting the Tick Rate	1-9
1.6.1. Specifying the Tick Rate for Components	1-9
1.6.2. Tuning the Network Interface	1-10
1.6.3. Tuning the State Repository Tick Rate	1-10
1.7. Balancing Visual Quality Against Network Performance	1-11
1.7.1. Enabling Configuration of Performance Options	1-12
1.8. Tuning the Target Frame Rate	1-12

Chapter 2 VR-Forces Configuration Files

2.1. Introduction	2-2
2.2. The <i>vrfSim.mtl</i> 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	2-10
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	3-6
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. Editing the Object Map File	3-11
3.5. Editing the Order of Battle File	3-11
3.5.1. Saving Default Entity Parameters to the Order of Battle File	3-12
3.6. Temporary Scenario Directories	3-12

Chapter 4 Merging Scenarios

4.1. The Scenario Merge Tool	4-2
4.2. Starting Scenario Merge	4-2
4.3. Creating a Scenario Merge Project	4-3
4.3.1. Adding Scenarios to a Project	4-3
4.3.2. Removing Scenarios from a Project	4-4
4.3.3. Specifying an Ignore List File	4-4
4.3.4. Saving a Project	4-5
4.3.5. Loading a Project	4-5

4.4. Merging Scenarios	4-6
4.5. Displaying Scenario Files	4-6
4.5.1. Editing Scenario Files	4-7
4.5.2. Closing Scenario Files	4-7
4.6. Specifying the Output Notification Level	4-7
4.7. Using the Scenario Merge Command-Line Interface	4-8

Chapter 5 Editing Entity Models

5.1. Introduction	5-3
5.1.1. Simulation Model Sets	5-4
5.1.2. Starting the Entity Editor	5-4
5.2. Creating a New Simulation Model Set	5-5
5.2.1. Creating a Completely New Simulation Model Set	5-6
5.2.2. Specifying the Default Simulation Model Set for Scenarios	5-6
5.3. Opening a Simulation Model Set	5-7
5.4. Editing an Entity Model	5-8
5.4.1. Editing an Entity's Category	5-9
5.4.2. Editing an Entity's Force	5-10
5.4.3. Changing an Entity's Entity Enumeration	5-11
5.4.4. Changing an Entity's Name	5-11
5.4.5. Specifying Whether an Entity can be Created	5-12
5.4.6. Editing Behavioral Parameters	5-12
5.4.7. Editing a Lifeform's DI-Guy Character, Appearance, and Animation	5-13
5.4.8. Undoing (Reverting) Changes to an Entity	5-14
5.5. Editing Aggregates	5-15
5.5.1. Adding Subordinates	5-17
5.5.2. Removing Subordinates	5-18
5.5.3. Replacing Subordinates	5-18
5.5.4. Changing the Order of Subordinates	5-18
5.5.5. Editing a Subordinate Aggregate	5-19
5.5.6. Adding Formations	5-20
5.5.7. Removing Formations	5-20
5.5.8. Renaming Formations	5-20
5.5.9. Specifying the Default Formation	5-21
5.5.10. Expanding and Collapsing the Display of Formations	5-21
5.5.11. Displaying Bounding Boxes	5-21
5.5.12. Copying Formations	5-22
5.5.13. Laying Out a Formation Automatically	5-23
5.5.14. Changing a Formation's Layout by Hand	5-23
5.6. Editing an Entity's Systems	5-24
5.6.1. Adding a System	5-26
5.6.2. Selecting a Damage Model	5-26
5.6.3. Selecting a Movement Model	5-26
5.6.4. Editing a System's Properties	5-27

5.6.5. Editing a System in the OPD Editor	5-27
5.6.6. Removing a System	5-27
5.7. Creating a New Entity	5-28
5.7.1. Creating a New Entity from an Existing Entity	5-28
5.8. Deleting an Entity Model from a Model Set	5-29
5.9. Navigating Among Edited Entities	5-29
5.10. Editing the Create Menu	5-29
5.10.1. Generating the Create Menu Automatically	5-30
5.10.2. Editing the Create Menu Manually	5-31
5.10.3. Undoing (Reverting) Changes to the Create Menu	5-34
5.11. Editing the Category List	5-34
5.11.1. Adding a New Category	5-34
5.11.2. Removing a Category	5-35
5.11.3. Renaming a Category	5-35
5.11.4. Undoing Category Changes	5-35
5.12. Managing the Forces List	5-36
5.12.1. Adding a Force	5-36
5.12.2. Editing a Force	5-37
5.12.3. Removing a Force	5-38

Chapter 6 Advanced Entity Model Editing

6.1. The Object Parameter Database Editor	6-2
6.2. Starting the OPD Editor	6-3
6.3. Loading an Object Parameter Database	6-3
6.3.1. Changing the Object List View	6-4
6.4. Editing an Object's Parameters	6-4
6.4.1. Editing Individual Parameters	6-5
6.4.2. Editing Parameters for Multiple Objects	6-6
6.4.3. Deleting Parameters and Components	6-6
6.4.4. Restoring Parameters	6-6
6.5. Creating a New Object from an Existing Object	6-7
6.6. Adding New Components and Parameters	6-8
6.6.1. Adding New Components	6-8
6.6.2. Editing Components	6-9
6.6.3. Adding New Resource Parameters	6-11
6.6.4. Adding New Subordinate Object Parameters	6-11
6.6.5. Adding New Soil Type Parameters	6-12
6.7. Editing Variables	6-12
6.8. Editing Systems	6-13
6.9. Saving Changes to the Object Parameter Database	6-14

Chapter 7 The Object Parameter Database

7.1. The Object Parameter Database	7-2
7.1.1. VR-Forces Classes and Object Parameters	7-6
7.1.2. Object Types	7-6
7.1.3. Parameter Types	7-9
7.1.4. Local and Remote Subcomponents	7-10
7.2. Component Descriptors and Parameters	7-13
7.2.1. Common Elements of Component Descriptors	7-14
7.2.2. Configuring the Priority of Components	7-16
7.3. Adding a New Entity to the Object Parameter Database	7-17
7.4. System Definition Files	7-18
7.4.1. Creating a System Definition File	7-19
7.4.2. Configuring System Connections	7-19
7.4.3. Adding Variable Bindings	7-22
7.5. Configuring Aggregate Entities	7-23
7.5.1. Adding an Aggregate to the Object Parameters Database	7-23
7.6. Changing the Echelon ID Category and Echelon ID Level	7-26
7.7. Configuring Tactical Graphics	7-27
7.8. Pathname Interpretation in Simulation Model Set Files	7-27
7.9. Updating Object Parameter Database Files for New Releases	7-28

Chapter 8 Configuring Object Behaviors

8.1. Configuring Sensors	8-3
8.1.1. Configuring a Sensor in an OPE File	8-3
8.1.2. Editing a Sensor System Definition File	8-4
8.1.3. Specifying a Sensor Signature	8-5
8.1.4. Adding a New Sensor Type	8-5
8.2. Detection Probability Tables	8-9
8.3. Probability and Selection Tables for Munitions	8-9
8.3.1. Ammo Select Tables	8-10
8.3.2. Hit Probability Tables	8-11
8.3.3. Damage Probability Tables	8-13
8.3.4. Configuring an Entity to Take Damage from a Munition	8-14
8.4. Configuring Formations	8-15
8.4.1. Formation Files	8-15
8.4.2. Creating a User-Defined Formation	8-16
8.5. Configuring Emitters	8-17
8.6. Configuring Radios	8-19
8.7. Configuring Sensor Volumes	8-20
8.7.1. Configuring Sensor Volume Color	8-20
8.7.2. Controlling Sensor Volume Radius	8-21
8.7.3. Configuring Sensor Volume Segments	8-22
8.8. Configuring Task Execution Rules	8-25
8.9. Configuring Joysticks	8-26

8.10. Configuring Obstruction Avoidance	8-27
8.10.1. Configuring Object Types	8-27
8.10.2. Configuring Vector Feature Types	8-28
8.11. Configuring Embarkation	8-29
8.11.1. Editing Embarkation Parameters	8-30
8.12. Configuring VR-Forces Components on Remote Entities	8-32

Chapter 9 Using VR-Forces with an External Comms Effects Server

9.1. Introduction	9-2
9.2. Configuring VR-Forces	9-2
9.2.1. Enabling Radio Transmitter Publishing	9-3
9.2.2. Enabling the External Communication Model	9-3
9.2.3. Configuring the External Communications Model	9-4
9.3. Using VR-Forces with Qualnet	9-5
9.3.1. Running VR-Forces with Qualnet	9-5
9.4. Example Scenario	9-6
9.4.1. Running the Scenario	9-6

Chapter 10 Display Configurations

10.1. The Display Engine	10-2
10.1.1. Window Types	10-2
10.1.2. Adding a Window	10-3
10.1.3. Adding a Channel to a Window	10-5
10.1.4. Removing a Window	10-5
10.1.5. Removing a Channel	10-6
10.1.6. Saving a Display Configuration	10-6
10.1.7. Loading a Display Configuration	10-6
10.2. Changing a Window's Attributes	10-6
10.3. Changing a Channel's Attributes	10-8
10.3.1. Setting the Clipping Planes	10-10
10.3.2. Specifying the Projection Resize Policy Attribute	10-11
10.3.3. Changing a Channel's Frustum (Field of View)	10-13
10.3.4. Changing the Viewport	10-14
10.4. Configuring Multichannel Displays	10-15
10.4.1. Changing the Camera's Position and Orientation Offset ..	10-17
10.5. Stereoscopic Displays	10-18
10.5.1. Configuring Anaglyphic Stereo	10-19
10.5.2. Configuring Polarized Stereo	10-19

Chapter 11 Composing Terrains

11.1. Creating a Composed Terrain	11-3
11.1.1. Considerations and Limitations for Building Terrains	11-3
11.1.2. Saving a Terrain	11-4
11.2. Adding Elevation Data (Terrain Patches) to a Terrain	11-4

11.3. Adding Images to a Terrain	11-6
11.3.1. Changing the Display Order of Raster Maps	11-9
11.4. Adding a Feature Layer	11-10
11.5. Connecting to Terrain Servers	11-11
11.5.1. Adding Terrain Server Connections	11-13
11.5.2. Editing a Terrain Server Configuration	11-14
11.6. Loading MetaFlight Terrains	11-15
11.6.1. Configuring MetaFlight File Loading	11-15
11.7. Manually Paging-In Terrain	11-18
11.8. Adding Props to a Terrain	11-19
11.8.1. Extracting Props from a Terrain Patch	11-19
11.8.2. Adding Props from a Feature Layer	11-21
11.8.3. Viewing a List of Props	11-23
11.8.4. Selecting Props	11-23
11.8.5. Setting the Opacity of Props	11-24
11.8.6. Changing a Prop's Type	11-24
11.8.7. Changing a Prop's Position or Orientation	11-25
11.8.8. Changing a Prop's Model Definition	11-25
11.9. Configuring File Caching	11-26
11.9.1. Enabling Texture Compression	11-26
11.9.2. Clearing the File Cache	11-27
11.10. Displaying DDS Textures Correctly	11-27
11.10.1. Flipping DDS Textures Globally	11-27
11.11. Displaying Information About Paged and Streaming Terrains ...	11-28
11.11.1. Displaying Additional Terrain Paging Information	11-29
11.11.2. Configuring Terrain Paging for the Back-End	11-30
11.12. Building Efficient MetaFlight Terrains for VR-Forces	11-31

Chapter 12 Mapping Models and Effects

12.1. Mapping Models to Entities and other Scene Elements	12-2
12.1.1. Adding a Model Mapping	12-3
12.1.2. Editing a Model Mapping	12-4
12.1.3. Filtering the Element Definition List	12-5
12.1.4. Deleting a Model Mapping	12-6
12.1.5. How VR-Forces Maps DI-Guy Models	12-6
12.2. Configuring Model Instancing	12-7
12.2.1. Clearing the Model Instancing Cache	12-7
12.3. Compressing Model Files	12-8

Chapter 13 Model and Element Definitions

13.1. Creating and Editing Schemas	13-3
13.1.1. Creating Schemas	13-5
13.1.2. Deleting Schemas	13-6
13.1.3. Copying Schemas	13-6
13.1.4. Adding a Parameter to a Schema	13-7
13.1.5. Deleting a Parameter from a Schema	13-8
13.1.6. Editing a Schema Parameter	13-8
13.2. Creating and Editing Model Definitions	13-9
13.2.1. Creating a Model Definition	13-10
13.2.2. Deleting a Model Definition	13-11
13.2.3. Copying a Model Definition	13-11
13.2.4. Filtering the List of Model Definitions	13-12
13.2.5. Editing a Model Definition	13-12
13.2.6. Saving Model Definitions	13-15
13.3. Creating and Editing Element Definitions	13-15
13.3.1. Creating an Element Definition	13-16
13.3.2. Copying an Element Definition	13-17
13.3.3. Editing an Element Definition	13-18
13.3.4. Deleting an Element Definition	13-19
13.3.5. Reorganizing Element Definitions	13-20
13.3.6. Saving Element Definitions	13-20
13.4. Editing 2D Icons	13-21
13.5. Adding New Cockpit Display Models	13-23
13.5.1. Installing a Cockpit DLL	13-24
13.5.2. Creating a Model Definition for a Cockpit	13-24
13.6. Flipping DDS Textures for a Model	13-26
13.7. Configuring SpeedTree Trees	13-27
13.7.1. Specifying the SpeedTree Performance Profile	13-28
13.8. Best Practices for Creating Models for VR-Forces	13-29

Appendix A Editing the Create Menu

A.1. The Create Menu	A-2
A.2. The <i>vrf.ent</i> Configuration File	A-2
A.2.1. Object Blocks in <i>vrf.ent</i>	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 <i>vrf.ent</i>	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 a Terrain Patch	C-3
C.1.2. Add Imagery	C-5
C.1.3. Finding the High Resolution Image	C-7
C.1.4. Add a Feature Layer	C-8
C.1.5. Add Props	C-8
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-24
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-12

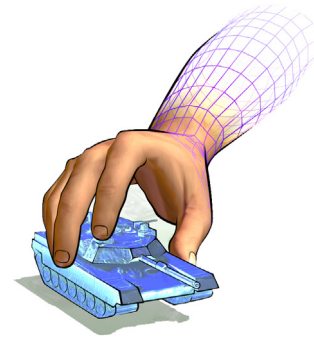
Appendix E Creating and Editing Key Mappings

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-5
E.4. Using Combined Key Mappings	E-6
E.5. Filtering the Function List	E-6
E.6. Creating a Key Map	E-7
E.7. Deleting a Key Map	E-7

Appendix F The WRM Specification

F.1. Background	F-2
F.2. General Information	F-3
F.3. Articulated Parts	F-4
F.4. Entity Appearances	F-5
F.4.1. Appearance Attributes Supported by VR-Forces Models	F-6
F.5. Animation	F-8

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 User's 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, *Merging Scenarios* explains how to use the Scenario Merge tool to merge scenarios.

Chapter 5, *Editing Entity 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 6, *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 7, *The Object Parameter Database* describes the object parameter database, which contains parameters for entities and control objects, and how to edit it.

Chapter 8, *Configuring Object Behaviors* describes additional configuration files that VR-Forces uses to manage munitions, formations, and joystick use.

Chapter 9, *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 10, *Display Configurations* explains how to configure windows and channels and how to save and load display configurations.

Chapter 11, *Composing Terrains* explains how to build terrain databases by importing elevation data, imagery, and feature data.

Chapter 12, *Mapping Models and Effects* explains how map entity enumerations to the models configured in VR-Forces.

Chapter 13, *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.

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 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 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 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.
- ♦ *VR-Forces Front-End Developers Guide* explains how to extend or modify the VR-Forces graphical user interface (GUI).
- ♦ *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.
- ♦ Class documentation. Classes are documented 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, RTIsby®, 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 three end-user applications — VR-Vantage Stealth, VR-Vantage XR, and VR-Vantage IG — and the VR-Vantage Toolkit. 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.

The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MÄK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.

- ♦ MÄK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- ♦ MÄK Plan View Display. The Plan View Display (PVD) provides a 2D, "big picture" of a virtual environment. It shows simulated entities, such as vehicles, soldiers, and tanks moving over a 2D-map display.
- ♦ 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.

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/licenses.php
Product version and platform information:	www.mak.com/support/productversions.php
For the free, unlicensed MÄK RTI:	www.mak.com/products/rti.php
Support FAQ:	www.mak.com/support/faq.php

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.
<i>Monospaced Italic</i>	Indicates command variables that you replace with appropriate values.
{ }	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.
<i>Italic</i>	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 .
i	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*.

Hardware and Operating System Naming Conventions

Unless otherwise specified, references to UNIX include Linux, regardless of the type of computer it is running on. References to PCs refer to Intel processor-compatible computers running a version of Microsoft Windows.

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/>

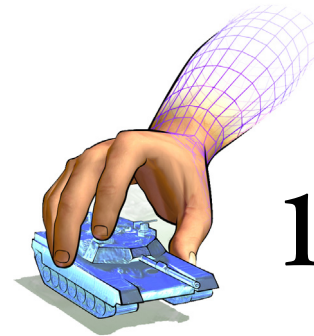
pThreads Library

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>



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	1-6
Configuring Interest Management	1-7
Filtering Entities by Observer Altitude	1-8
Miscellaneous Performance Configuration Options	1-8
Limiting Use of Spot Reports	1-8
Configuring Intervisibility Intersection Testing	1-8
Using Asynchronous I/O for DIS Exercises	1-8
Setting the Tick Rate	1-9
Specifying the Tick Rate for Components	1-9
Tuning the Network Interface	1-10
Tuning the State Repository Tick Rate	1-10
Balancing Visual Quality Against Network Performance	1-11
Enabling Configuration of Performance Options	1-12
Tuning the Target Frame Rate	1-12

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
- ♦ Turning off processor-intensive features
- ♦ Adjusting the network processing time
- ♦ Specifying the tick rate for an entity or component
- ♦ Specifying a tick rate for the network interface
- ♦ Specifying the VR-Forces sleep time
- ♦ Specifying the number of objects that you expect to simulate in the exercise
- ♦ Optimizing the terrain. (For details, please see *MAK Terrain Database Tool Users Guide*.)
- ♦ 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
- ♦ 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 .

The following actions can affect performance:

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

The following sections describe options for optimizing performance:

- ♦ [“Configuring SpeedTree Trees,”](#) on page 16-29
- ♦ [“Configuring File Caching,”](#) on page 5-29
- ♦ [“Configuring Model Instancing,”](#) on page 15-7
- ♦ [“Setting the Clipping Planes,”](#) on page 6-14
- ♦ [“Enabling Texture Compression,”](#) on page 5-29
- ♦ [“Considerations and Limitations for Building Terrains,”](#) on page 5-3
- ♦ [“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](#)).

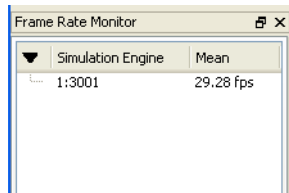


Figure 1-1. Frame Rate Monitor panel

- 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](#)).

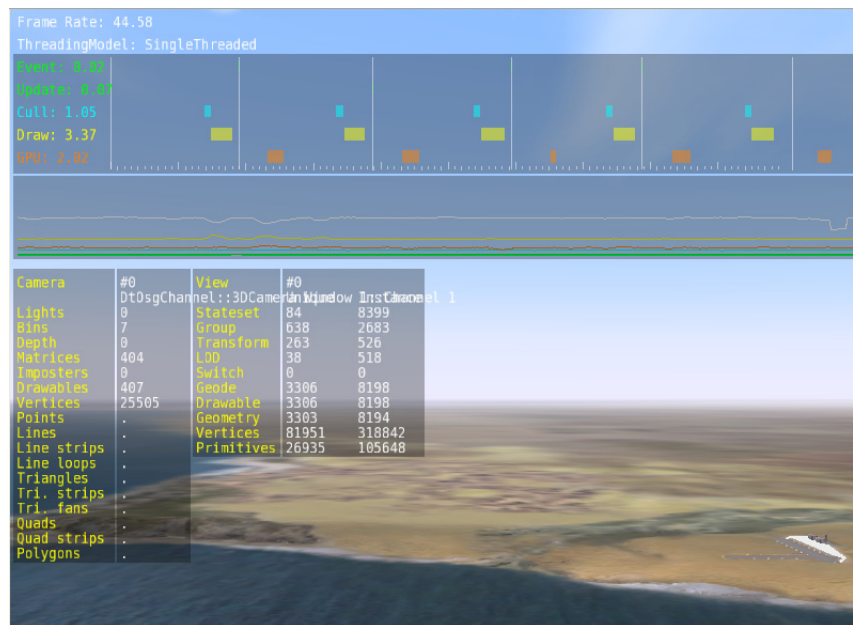


Figure 1-2. OSG performance statistics



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).

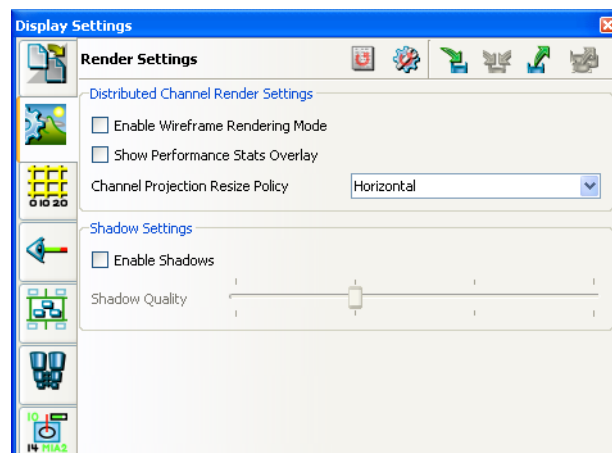


Figure 1-4. Render Settings page

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.

To configure interest management:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Interest Management Settings page ([Figure 1-5](#)).

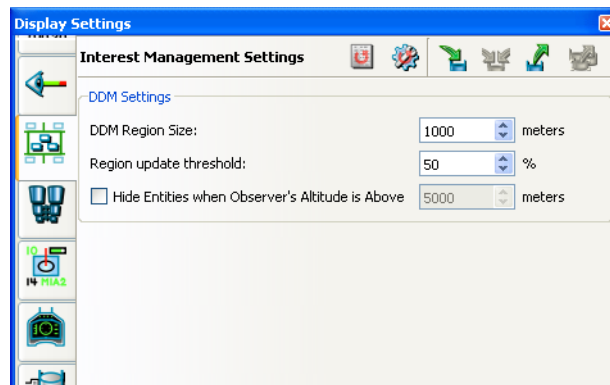


Figure 1-5. Interest Management Settings page

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 Altitude Higher Than check box.
 - b. Specify the observer altitude at which to hide entities.

For more information about filtering entities based on observer altitude, please see [“Filtering Entities by Observer Altitude,”](#) on page 1-8.

6. Click OK.

1.4.3. Filtering Entities by Observer Altitude

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.

1.5. Miscellaneous Performance Configuration Options

You can improve front-end performance by limiting use of display features, as follows:

- ♦ Turn off track histories
- ♦ Limit use of object transparency
- ♦ Limit use of spot reports
- ♦ Minimize the types of intervisibility intersections you test for
- ♦ 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. Configuring Intervisibility Intersection Testing

When it calculates intervisibility, VR-Forces can check for intersections of polygons, features, and entities. Depending on the complexity of the terrain database and the number of entities in a simulation, increasing the types of intersections for which VR-Forces tests can degrade performance. Therefore, if performance is an issue, minimize the intersection testing being done.

1.5.3. 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:

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

If you insert some sample values (in seconds), you get:

$$t_{\text{next}} = .1 + .05 +/\text{- rand}(.05/2)$$

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).

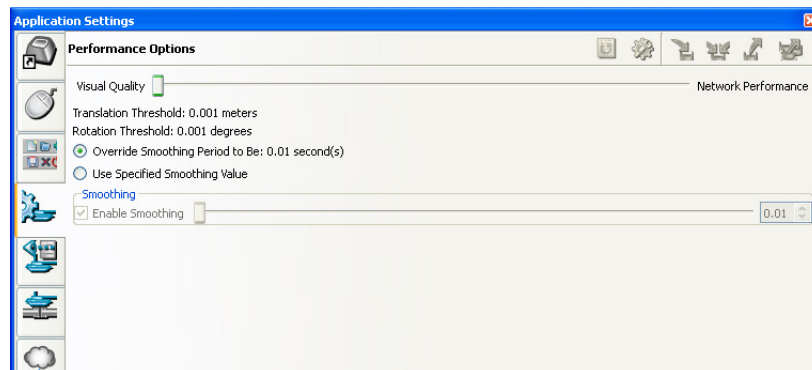


Figure 1-6. Performance Options dialog box

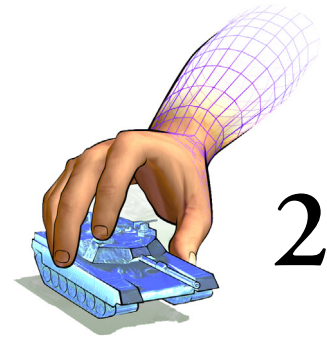
3. Adjust the Visual Quality slider to your desired preference for visual quality versus network performance.
4. To override the default smoothing period and set it to a value calculated to work best with the performance setting, select Override Smoothing Period.
5. Click OK.

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 `vrSim.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.



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	2-10
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	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.	
aggregateSpatialModelTickPeriod		
aggregateSpatialModelTickVariance		
aggregateSpatialModelTickAsFastAsPossibleWhilePaused		
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 <i>VR-Forces Developers Guide</i> .	
batchScenarioFileName	Specifies a batch file to run.	-B
cgfDispatcherReceivePort	Specifies the port to be used by the <i>DtCgfDispatcher</i> ’s file transporter to receive order of battle and plan files from the <i>DtVrfRemoteController</i> .	
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.	
ignoreAdvisories	Turns off HLA advisory messages.	
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.) 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, <i>./plugins/</i> .	
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 1-12. 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 5.9.5, “Command Line Options for HLA Federations” in <i>VR-Forces Users Guide</i> .	
destroyFedExec	Specifies whether (1) or not (0) the <i>DtExerciseConn</i> destructor should try to destroy the federation execution. Default: 1.	
fomMapperLib	A FOM Mapper library.	-f
fomMapperInitData	An initialization string.	--fomMapperInitData
execName	The federation execution name.	-x

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

Parameter	Description	Option
federateName	The federate name.	-N
fedFileName	Specifies the FED file name.	-F
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 5.9.6, “Command Line Options for DIS Exercises” in <i>VR-Forces Users Guide</i> .	
addMulticastAddr	The multicast address.	-S
appldRange	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
exerciseld	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 5-13.) 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 the 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 make up new types of 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.

i

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 9, *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))
```

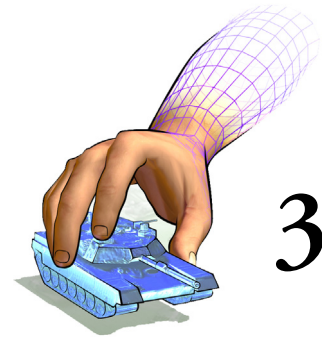
2.5.3. Using Conditional Statements

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)
  )
)
```

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.....	3-6
Changing the Terrain Database for a Scenario	3-8
Editing the Plan Files	3-8
The Structure of an Entity Plan File	3-9
Editing the Object Map File	3-11
Editing the Order of Battle File.....	3-11
Saving Default Entity Parameters to the Order of Battle File	3-12
Temporary Scenario Directories	3-12

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, named *scenario_name.gpl*, stores the scenario's global plans
- The hostility relationships file, named *scenario_name.hst*, lists the forces that each force is hostile to.

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 option to attach components to remote entities. However, you can edit them in a text editor.

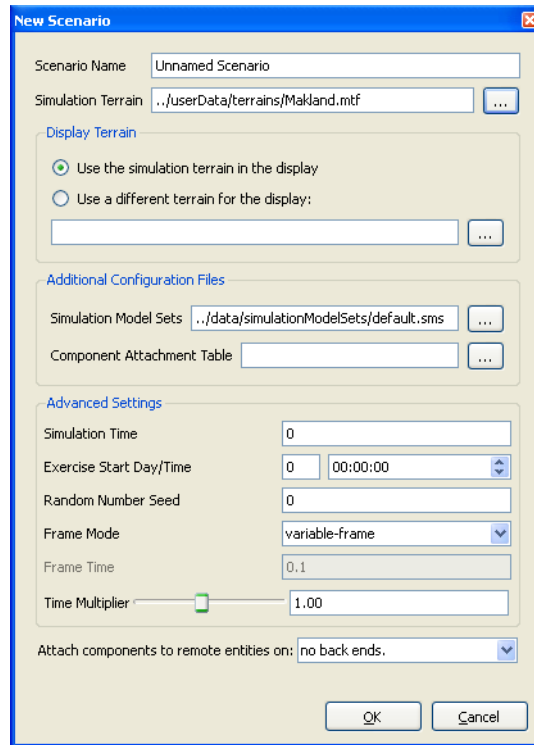


Figure 3-1. Choose Terrain dialog box

The default values for scenario settings in the New Scenario dialog box are drawn from the *default.scn* file. 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 "Makland.gpl")
  (Overlay "example.ovl")
  (SelectionGroups "example.sgr")
  (Object-Map "example.omp")
  (Force-Hostility "example.hst")
  (time-multiplier 1.000000)
  (auto-reorganize False)
  (random-number-seed 0)
  (frame-mode "variable-frame")
  (frame-time 0.000000)
  (sim-time 0.000000)
  (exercise-start-time 0.000000)
  (additional-opds )
  (Simulation-Model-Set-Files "../data/simulationModelSets/default.sms")
  (scenario-name "Scenario created on ../userData/terrains/Makland.mtf")
  (scenario-data )
  (version 3.100000)
)
```

i

- ♦ Although you can edit the `sim-time` and `exercise-start-time` parameters in scenario files, we strongly recommend that you not change these parameters for scenarios that were saved with a simulation time other than 0. Some of the state information saved in the scenario may depend on these values and if you change them, the behavior of the scenario may not be what you expect.
 - ♦ Paths in the scenario files are relative to the application directory, for example, `./bin`.
-

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 file¹.
- ♦ **SelectionGroups** specifies the selection groups file¹.
- ♦ **Object-Map** specifies the object mapping file.
- ♦ **Force-Hostility** specifies the force hostility file.
- ♦ **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 Management, it is strongly recommended that you set **time-multiplier** to 1.

- ♦ **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*.

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.

- ♦ **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 3.13.2, “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 3.13.2, “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.
- ♦ **sim-time** specifies the starting simulation time, in seconds. By default, a new scenario starts with simulation time set to 0.0. A scenario saved as a checkpoint saves the simulation time of the checkpoint.
- ♦ **exercise-start-time** specifies the starting exercise time for the scenario, in seconds. By default, an exercise starts at day 1, zero hours, minutes, and seconds. However, you can set the scenario to start running with a different exercise time.
- ♦ **additional-opds** specifies custom OPD files to load (for backwards compatibility.)
- ♦ **Simulation-Model-Set-Files** specifies a list of SMS files to load.
- ♦ **version** specifies versioning information for the scenario file. Do not edit this parameter.

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 *vrSim.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/bin/vrSim*).
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):

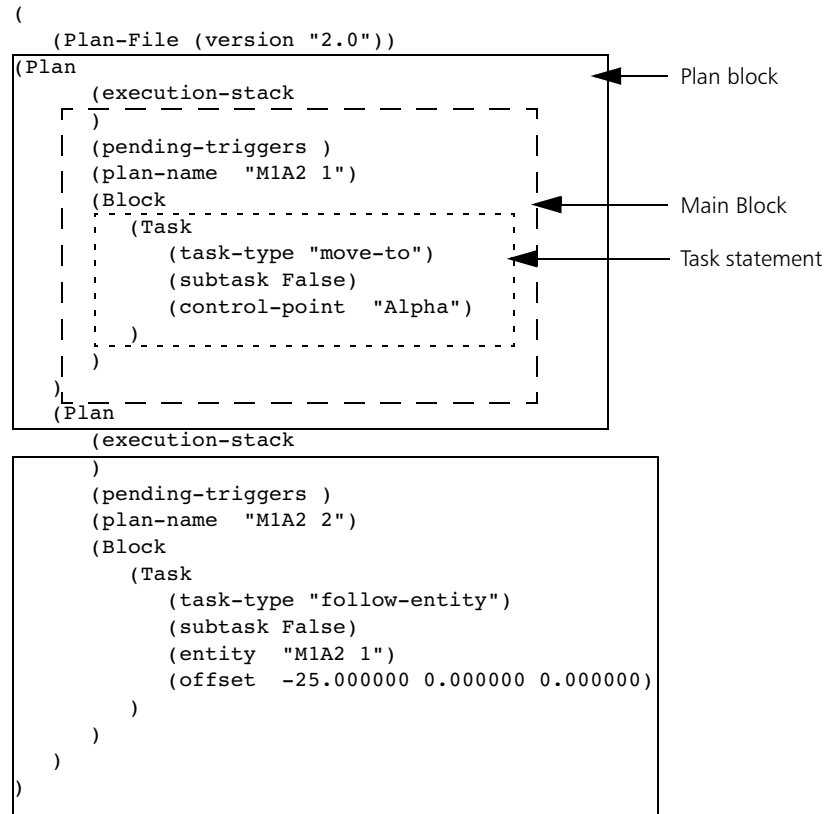


Figure 3-2. Simple plan

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.

```

(Block ← Main block
  (If ← If statement
    (Condition
      (EntInArea
        (Entity "M1A2 1")
        (Area "Blue")
        (Modifier ",")
      )
    )
    (Then-Block ← Then block
      (Task
        (task-type "move-to")
        (subtask False)
        (control-point "Bravo")
      )
    )
    (Else-Block ← Else block
      (Task
        (task-type "move-to")
        (subtask False)
        (control-point "Alpha")
      )
    )
  )
  (Task ← Task statement
    (task-type "follow-entity")
    (subtask False)
    (entity "M1A2 1")
    (offset -25.000000 0.000000 0.000000)
  )
)

```

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. Editing the Object Map File

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 control object 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

VR-Forces saves entity state information and graphical object information to the order of battle file.

If you save a scenario after the simulation has run, for example, as a checkpoint, the order of battle file stores the state information for the entities at the time of the save. 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.

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:

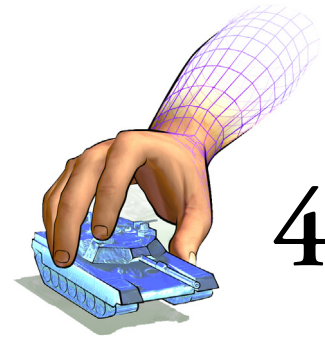
```
(parameter-name USE-DEFAULT)
```

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. 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 *vrfbe*. 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.



Merging Scenarios

This chapter explains how to merge scenarios with the Scenario Merge tool.

The Scenario Merge Tool	4-2
Starting Scenario Merge	4-2
Creating a Scenario Merge Project.....	4-3
Adding Scenarios to a Project	4-3
Removing Scenarios from a Project.....	4-4
Specifying an Ignore List File.....	4-4
Saving a Project	4-5
Loading a Project.....	4-5
Merging Scenarios.....	4-6
Displaying Scenario Files	4-6
Editing Scenario Files	4-7
Closing Scenario Files.....	4-7
Specifying the Output Notification Level.....	4-7
Using the Scenario Merge Command-Line Interface.....	4-8

4.1. The Scenario Merge Tool

VR-Forces users often assign scenario development to multiple persons or teams. Then they merge the separately-developed scenarios into a master scenario. Merging scenarios by hand can be a difficult task due to the requirement to reconcile entity and object names, plans, and other scenario elements that could be duplicated, particularly if objects are named using the default conventions.

The Scenario Merge tool allows you to combine two or more VR-Forces scenarios into a single scenario. Scenario Merge detects and resolves conflicts so that the simulated objects can co-exist in a single scenario, without losing their unique identity. Likewise, references to conflicting names are resolved, so that the plans from both scenarios work properly.

Scenario Merge has a graphical user interface and a command-line interface. It is an off-line tool for combining existing scenarios.

Scenario Merge has an API that lets you merge scenarios programmatically. For details, please see the *DtScenarioMergeDirector* class (*scenarioMergeDirector.h*) in the class documentation.

4.2. Starting Scenario Merge

The graphical user interface is the primary way to use the Scenario Merge tool.

- To start Scenario Merge in Windows, on the Start menu, choose **Programs** → **MAK Technologies** → **VR-Forces 4.0** → **Tools** → **Scenario Merge**.

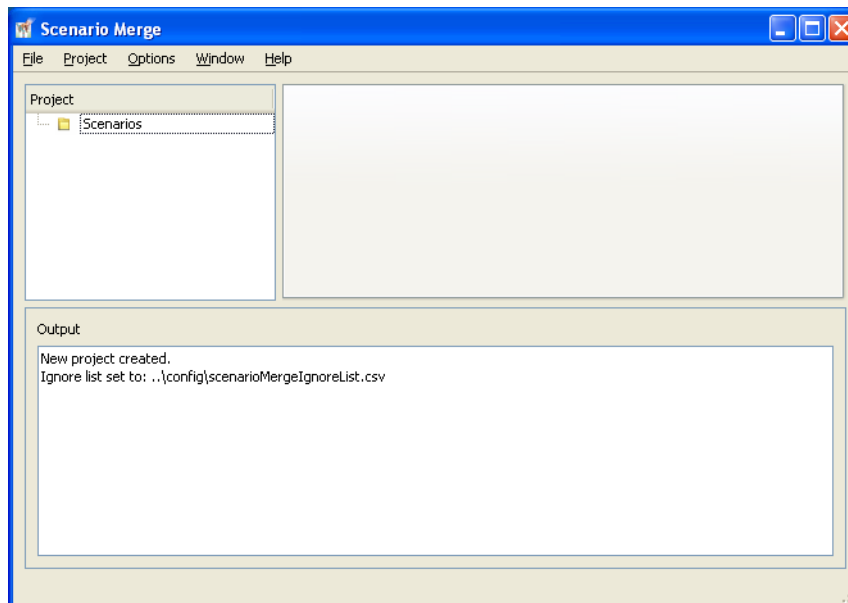


Figure 4-1. Scenario Merge window at startup

- To start the Scenario Merge from the command line, enter the following:

```
./bin/scenarioMerge [projectFile]
```

If you start Scenario Merge from the command line and specify *projectFile*, Scenario Merge loads the project file. Otherwise, it opens a new project. Command-line syntax is described in [“Using the Scenario Merge Command-Line Interface,”](#) on page 4-8.

4.3. Creating a Scenario Merge Project

Before you can merge scenarios, you must create a Scenario Merge project. A project identifies the scenarios to merge, where to write the merged scenario, and optionally, a list of attributes to ignore during the merge process.

When you start Scenario Merge, it automatically creates a new, empty project. However, you can also start a new project manually.

- To open a new project, choose **File** → **New Project**.

4.3.1. Adding Scenarios to a Project

You can add as many scenarios as you want to a project. However, the first scenario you add is the key scenario. It determines the parameters of the final scenario, including, terrain database, object parameter database, exercise start time, and so on. If any of the scenarios that you merge into the first scenario have different scenario parameters, they are ignored and a diagnostic message is recorded in the project log file.

If you add scenarios from different terrains, Scenario Merge uses only geocentric coordinates for entity position data. This is accomplished using the Ignore List File, which is explained in [“Specifying an Ignore List File,”](#) on page 4-4.



Before you add a scenario created in VR-Forces 3.10.x or earlier to a project, open it in the current version of VR-Forces and save it.

To add a scenario to a project, use one of the following methods:

- ♦ Choose **Project** → **Add Scenario**, and select a scenario to add.
- ♦ Drag a scenario file (*.scn*) from Windows Explorer to the Scenario Merge Project tree view.
- ♦ Drag a scenario folder from Windows Explorer to the Scenario Merge Project tree view. If you drag a folder, Scenario Merge adds all scenario files in the folder to the project.

4.3.2. Removing Scenarios from a Project

To remove a scenario from a project:

1. In the Project window, select the scenario that you want to remove.
2. Choose **Project** → **Remove Scenario**.

4.3.3. Specifying an Ignore List File

Sometimes it is useful to suppress values when writing to the final scenario. For example, Scenario Merge suppresses any reference to a local coordinate system in the Order of Battle file (*.oob*) file. You can specify values to ignore in an ignore list file. The ignore list file is a comma-delimited text file, for example:

```
local-kinematics-state,IGNORE  
local-vertices,IGNORE
```

The first value is the string to search for during a merge operation. The second value is a text string that will replace the key in the merged scenario.

If a substitution is specified in the ignore list, it is applied to the following files in the scenario:

- ♦ Order of Battle (*.oob*)
- ♦ Plan (*.pln*)
- ♦ Global Plan (*.gpl*)
- ♦ Object Map (*.omp*).

To create an ignore list file, you must use a text editor. Scenario Merge just lets you specify which file to use. You must include the following lines from the default ignore list:

```
local-kinematics-state,IGNORE  
local-vertices,IGNORE
```

These entries are needed when merging scenarios that use different terrains.

To specify the Ignore List file:

1. Choose **Project** → **Specify Ignore List**. The Specify Ignore File dialog box opens.
2. Type a filename or click the Browse button and select the Ignore List file.
3. Click OK.

4.3.4. Saving a Project

When you save a project, Scenario Merge creates a project file. You can load the project at a later date to merge the scenario again or change which scenarios you want to merge. You can also use a project file to merge files from the command-line. For details about running from the command-line, please see [“Using the Scenario Merge Command-Line Interface,”](#) on page 4-8.

To save a project:

1. Choose **File** → **Save Project**. The Scenario Merge dialog box opens.
2. Select a directory and type a name for the project.
3. Click Save. The project gets saved with the extension *.smp*.

The Scenario Merge Project File

A merge project is a comma-delimited text file that lists the scenarios to merge, the desired location for the final result, and merge options. The following is an example of a Scenario Merge project file:

```
OUT,..\\data\\scenarios\\demoMerge
IGN,..\\config\\scenarioMergeIgnoreList.csv
SCN,..\\data\\scenarios\\scenarioOne\\scenarioOne.scn
SCN,..\\data\\scenarios\\scenarioTwo\\scenarioTwo.scn
SCN,..\\data\\scenarios\\scenarioThree\\scenarioThree.scn
SCN,..\\data\\scenarios\\scenarioFlour\\scenarioFlour.scn
```

Each line starts with one of the following keywords:

- ♦ OUT specifies the location for the output scenario files.
- ♦ IGN specifies an Ignore List file. For details, please see [“Specifying an Ignore List File,”](#) on page 4-4.
- ♦ SCN specifies the path of a scenario to merge in.



Paths are relative to the location of *scenarioMerge.exe*.

4.3.5. Loading a Project

If you have saved a project, you can load it, edit it as necessary, and regenerate the merged scenario.

To load a project:

1. Choose **File** → **Load Project**. An Open dialog box opens.
2. Select the Scenario Merge project file that you want to load.
3. Click Open.

4.4. Merging Scenarios

- To merge the scenarios in a project, choose **Project** → **Merge**. The scenarios are merged and informational messages are displayed in the console.



Scenario files created in VR-Forces 4.0 or later have entries for selection group (.sgr) or overlay (.ovl) files. However, these files are no longer supported and are not created. When you merge scenarios, Scenario Merge prints messages indicating it cannot find these files. You can safely ignore these messages.

4.5. Displaying Scenario Files

You can display the text of one or more of the scenario files in a project.

- To display a scenario file, in the Project window, double click the file. It is displayed to the right of the Project window.

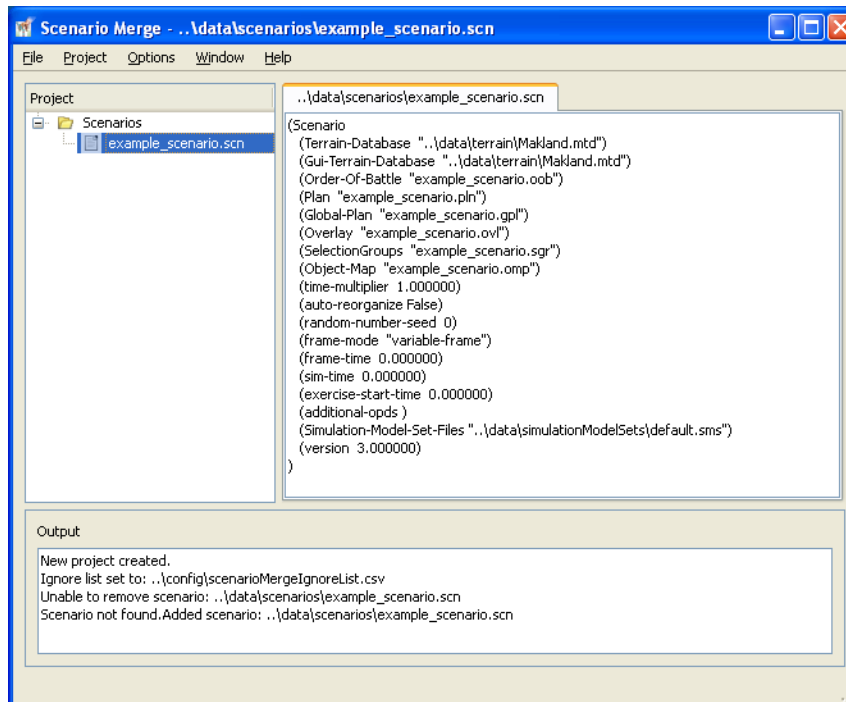


Figure 4-2. Scenario file display

4.5.1. Editing Scenario Files

If you display a scenario file, you can edit it. Any edits you make are saved back to the original scenario. You cannot edit the scenario file that gets created when you merge scenarios.



Editing scenario files is risky to scenarios and is advisable only for experienced VR-Forces users. Be sure you fully understand the contents of the file.

To edit a scenario file:

1. In the scenario file display window, make any edits desired.
2. Choose **File** → **Save Scenario** or **Save All**.

4.5.2. Closing Scenario Files

To close a scenario file that is displayed by Scenario Merge, use one of the following methods:

- ♦ Choose **Window** → **Close**. If multiple files are open, this closes the most recently opened one.
- ♦ Choose **Window** → **Close All**.
- ♦ Choose **File** → **Close Scenario**.

4.6. Specifying the Output Notification Level

You can specify the notification level used for sending messages to the output console. The notification levels correspond to the notification levels used throughout MÄK products and described in Chapter 2, [VR-Forces Configuration Files](#).

- To set the notification level, choose **Options** → **Set Notify Level** → *level*, where *level* is one of the following:
 - ♦ Fatal
 - ♦ Warning
 - ♦ Info
 - ♦ Verbose
 - ♦ Debug.

4.7. Using the Scenario Merge Command-Line Interface

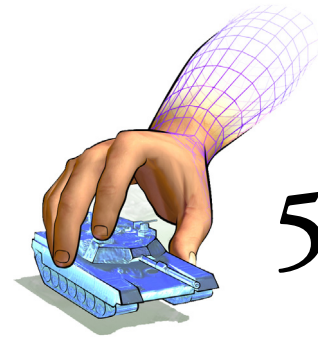
To run the Scenario Merge tool from the command line, use the following syntax:

```
scenarioMerge -c projectFile
```

where:

-c specifies that Scenario Merge should run in command-line mode.

projectFile specifies a Scenario Merge project file to load. For details about creating project files, please see [“Adding Scenarios to a Project,”](#) on page 4-3.



Editing Entity 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	5-3
Simulation Model Sets.....	5-4
Starting the Entity Editor	5-4
Creating a New Simulation Model Set	5-5
Creating a Completely New Simulation Model Set.....	5-6
Specifying the Default Simulation Model Set for Scenarios	5-6
Opening a Simulation Model Set	5-7
Editing an Entity Model	5-8
Editing an Entity's Category	5-9
Editing an Entity's Force.....	5-10
Changing an Entity's Entity Enumeration	5-11
Changing an Entity's Name	5-11
Specifying Whether an Entity can be Created.....	5-12
Editing Behavioral Parameters	5-12
Editing a Lifeform's DI-Guy Character, Appearance, and Animation....	5-13
Undoing (Reverting) Changes to an Entity.....	5-14
Editing Aggregates	5-15
Adding Subordinates	5-17
Removing Subordinates.....	5-18
Replacing Subordinates	5-18
Changing the Order of Subordinates	5-18
Editing a Subordinate Aggregate.....	5-19
Adding Formations.....	5-20
Removing Formations	5-20

Renaming Formations	5-20
Specifying the Default Formation.....	5-21
Expanding and Collapsing the Display of Formations	5-21
Displaying Bounding Boxes	5-21
Copying Formations	5-22
Laying Out a Formation Automatically	5-23
Changing a Formation's Layout by Hand	5-23
Editing an Entity's Systems	5-24
Adding a System.....	5-26
Selecting a Damage Model	5-26
Selecting a Movement Model	5-26
Editing a System's Properties	5-27
Editing a System in the OPD Editor	5-27
Removing a System	5-27
Creating a New Entity	5-28
Creating a New Entity from an Existing Entity	5-28
Deleting an Entity Model from a Model Set	5-29
Navigating Among Edited Entities.....	5-29
Editing the Create Menu	5-29
Generating the Create Menu Automatically	5-30
Editing the Create Menu Manually	5-31
Undoing (Reverting) Changes to the Create Menu.....	5-34
Editing the Category List.....	5-34
Adding a New Category	5-34
Removing a Category	5-35
Renaming a Category	5-35
Undoing Category Changes	5-35
Managing the Forces List	5-36
Adding a Force	5-36
Editing a Force	5-37
Removing a Force.....	5-38

5.1. Introduction

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. To change entity models, icons, or other visualization characteristics, you must edit model definitions and model mappings. For more information, please see Chapter 12, *Mapping Models and Effects* and Chapter 13, *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 6, *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 5-27. Use one tool, save the changes, and close the tool before opening the other.

5.1.1. 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 5-5.

5.1.2. Starting the Entity Editor

- To start the Entity Editor in Windows, choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.0** → **Tools** → **Entity Editor**. The Entity Editor opens ([Figure 5-1](#)).

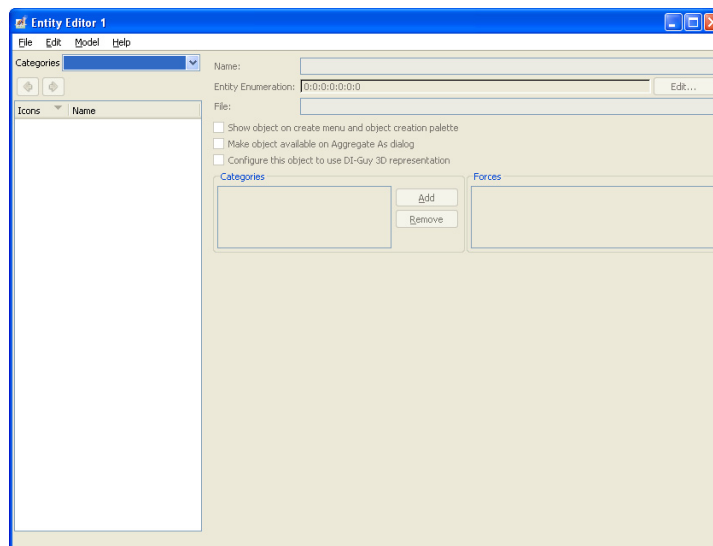


Figure 5-1. Entity Editor

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

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

```
entityEditor
```

5.2. 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 5-2).

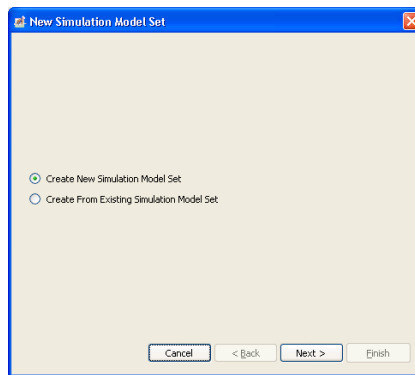


Figure 5-2. New Simulation Model Set dialog box

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**.

5.2.1. 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 5-2](#)).
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 5-28.
7. Choose **File** → **Save Simulation Model Set**.

5.2.2. 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`.

5.3. 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.

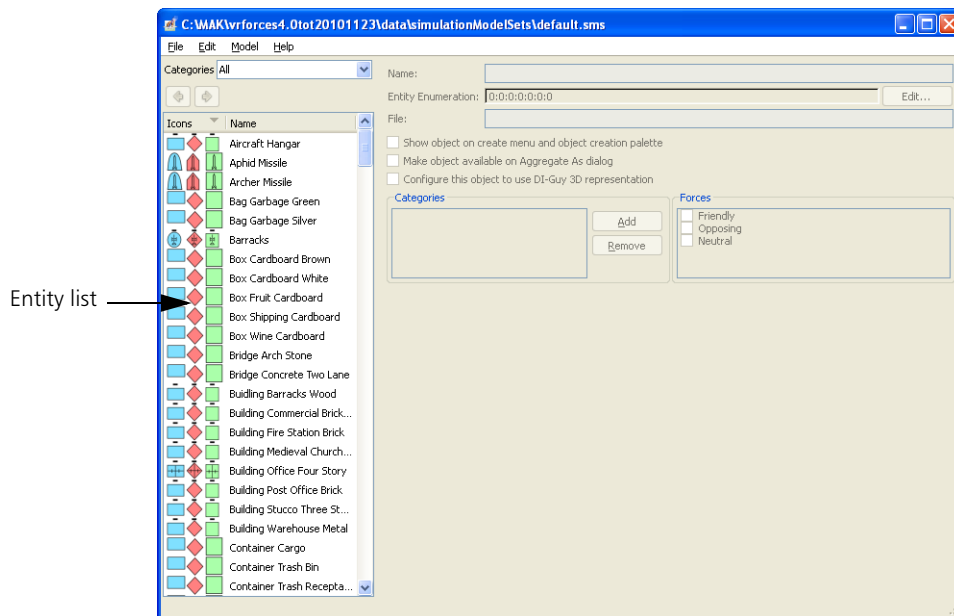


Figure 5-3. Entity Editor with model set loaded

5.4. Editing an Entity 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.



The 3D models and 2D icons used to display entities are configured as part of model definitions and entity definitions. For information about changing models and icons, please see Chapter 13, *Model and Element Definitions*.



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.

5.4.1. 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.

To change an entity's category:

1. In the Categories group box, click the category that you want to change. The field becomes editable.
2. Select a different category from the drop-down list.

To add a category:

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

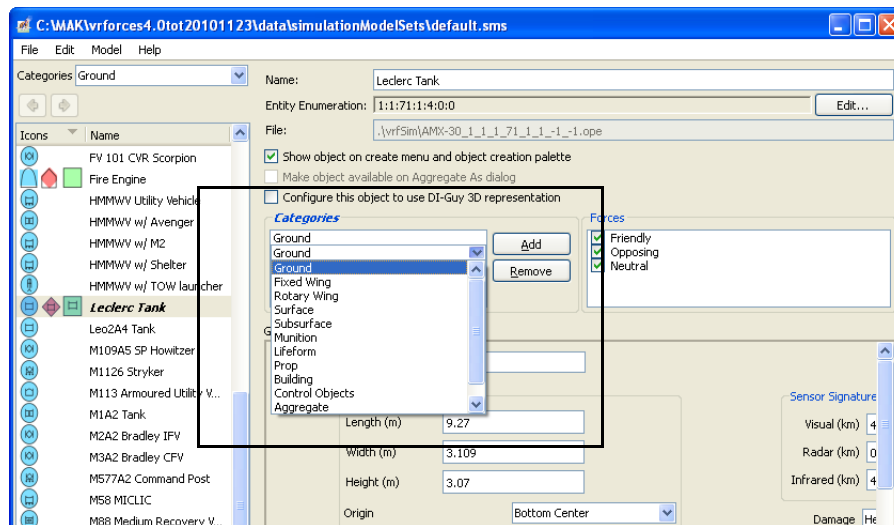


Figure 5-4. Adding a category

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

5.4.2. 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 5-5).

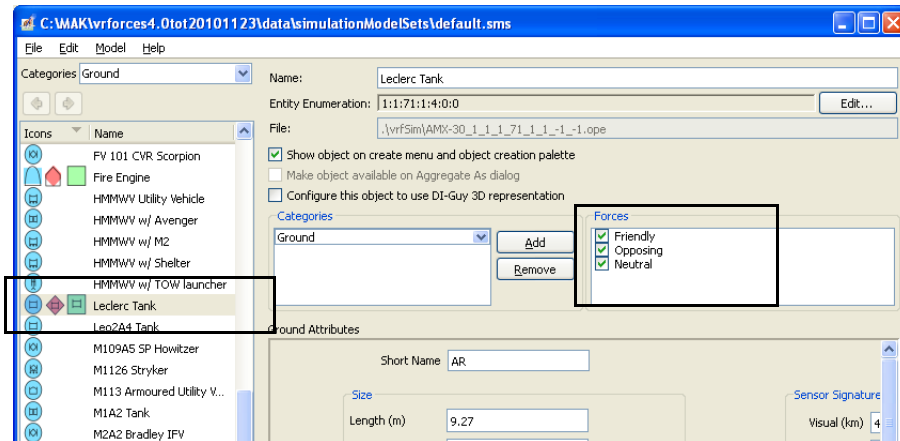


Figure 5-5. Entity (civilian vehicle) assigned to multiple forces

- 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.

5.4.3. 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. The Change Enumeration dialog box opens (Figure 5-6).

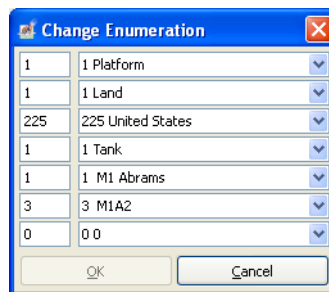


Figure 5-6. Change Enumeration dialog box

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.

5.4.4. 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.

5.4.5. 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 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 5-7).

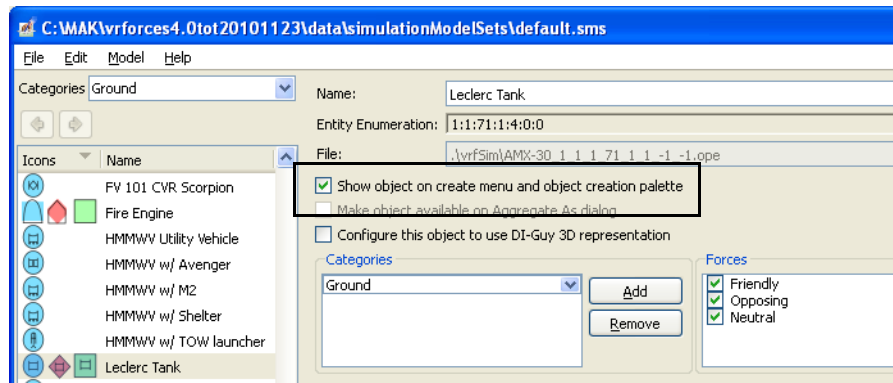


Figure 5-7. Show Object on Create Menu and Object Creation Palette check box

5.4.6. 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 5-24.

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

5.4.7. 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 the 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.

To specify the default DI-Guy characteristics for a lifeform:

1. In the Categories drop-down list, select the Lifeform category.
2. Select an entity in the entity list.
3. In the DI-Guy Character group box, select the default character from the Character Type drop-down list ([Figure 5-8](#)).

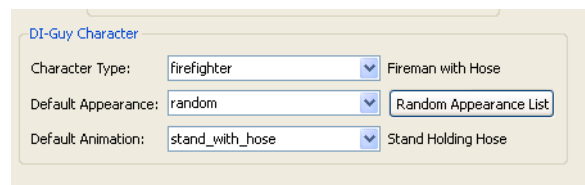


Figure 5-8. DI-Guy Character group box

4. Select the default appearance from the Default Appearance drop-down list. One of the default options is to use a random appearance.
5. 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 5-14.
6. 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 DI-Guy Character group box, click the Random Appearance List button. The Random Appearance List dialog box opens (Figure 5-9). The appearances available for the character type are listed.

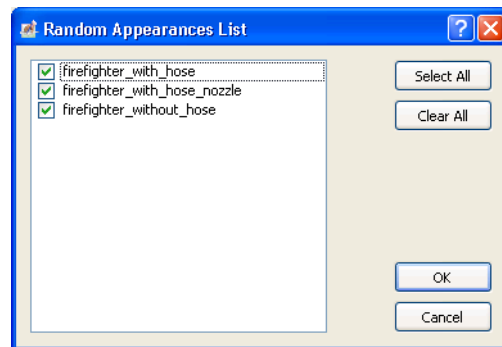


Figure 5-9. Random Appearances List dialog box

2. Select the appearances that you want to include in the random appearance list.
3. Clear any appearances that you do not want to include in the random appearance list.
4. Click OK.

5.4.8. 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**.

5.5. 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.

i

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 5-10 illustrates the Entity Editor with an aggregate selected. Area 1 is generic, as mentioned earlier in this section. Area 2 is the Composition Editor. Area 3 is the Formation Editor.

i

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

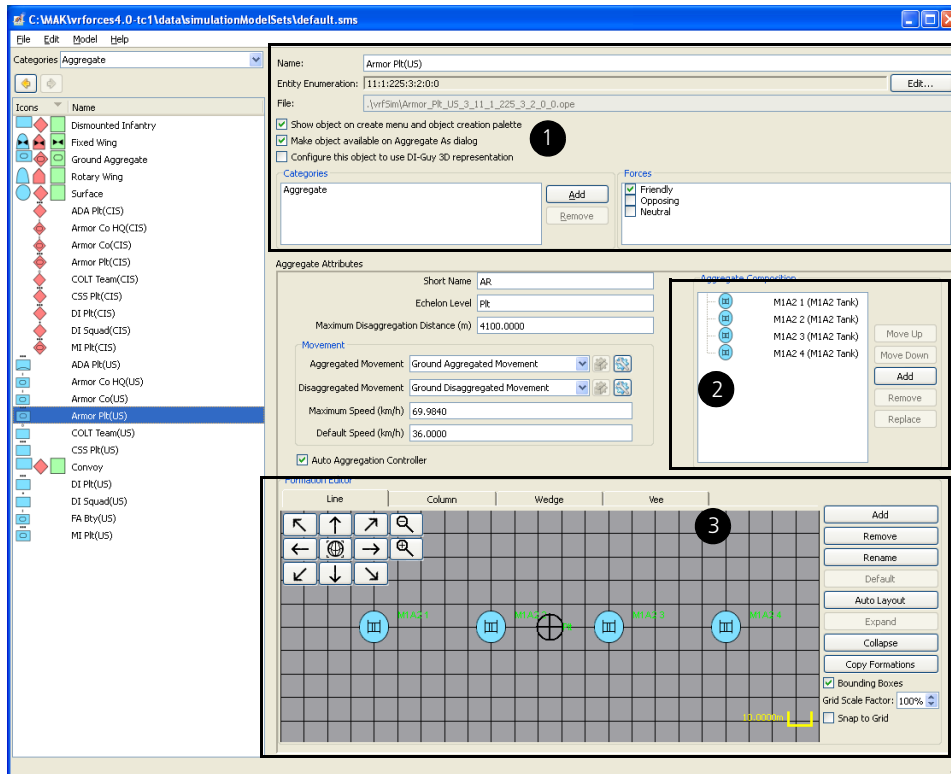


Figure 5-10. Aggregate page in Entity Editor

5.5.1. 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 5-10](#)), click the Add button. The Add to Hierarchy dialog box opens ([Figure 5-11](#)).

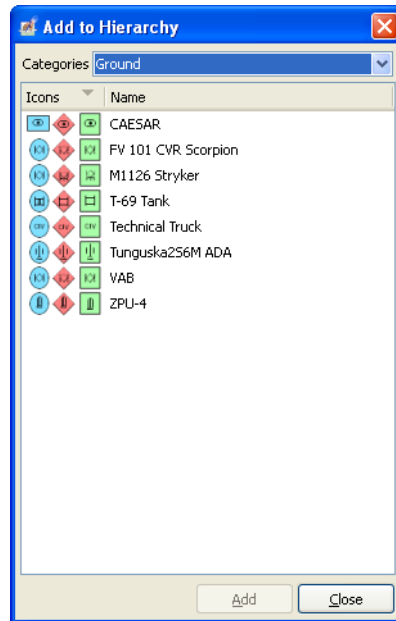


Figure 5-11. Add to Hierarchy dialog box

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.

5.5.2. 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.

5.5.3. 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 5-11](#)).
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.

5.5.4. 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.

5.5.5. 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 redisplay 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 5-12](#)).

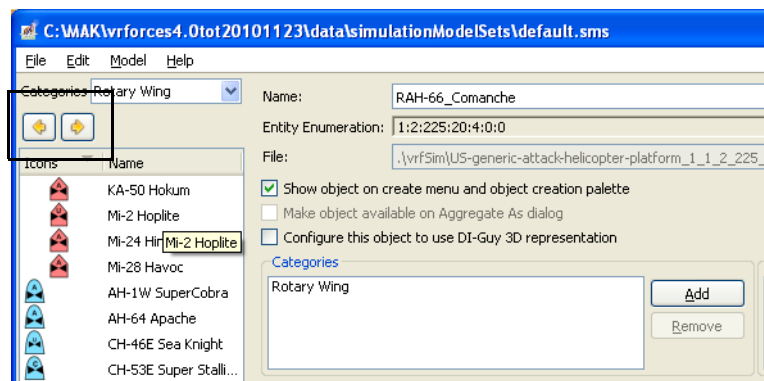


Figure 5-12. Forward and Back arrows

5.5.6. Adding Formations

To add a formation:

1. In the Formation Editor group box, click Add. The Add Formation dialog box opens ([Figure 5-13](#)).

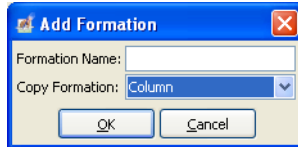


Figure 5-13. Add Formation dialog box

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 5-23.

5.5.7. 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.

5.5.8. 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.

5.5.9. 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.

5.5.10. 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.

5.5.11. 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.

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

5.5.12. 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 5-14).

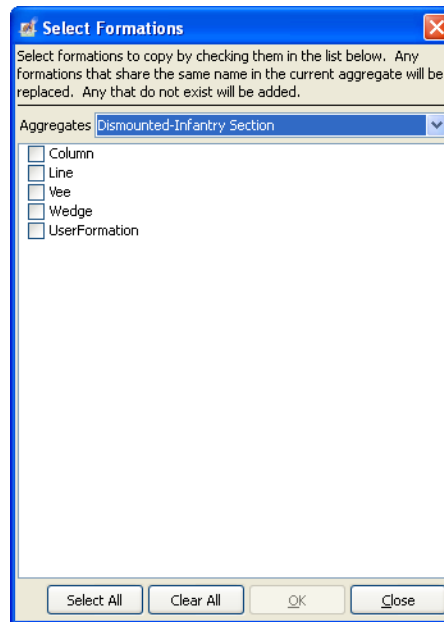


Figure 5-14. Select Formations dialog box

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.

5.5.13. 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.

5.5.14. 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 5-15). This is where the aggregate icon is placed when the aggregate is collapsed or aggregated.

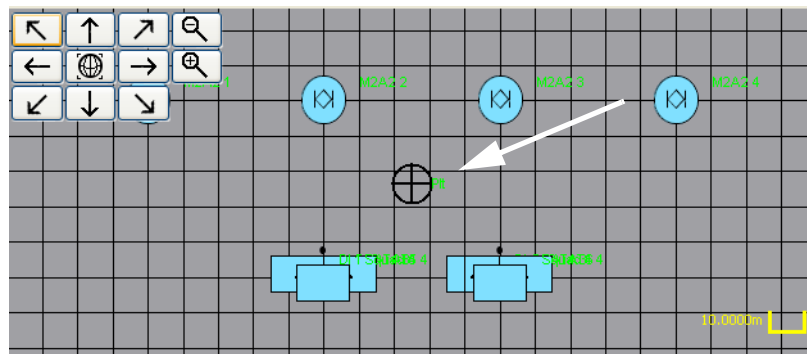


Figure 5-15. Aggregate formation editor crosshair

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.

5.6. 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 5-16). For example, a lifeform can have an M-16 or an RPG, but not a cruise missile.

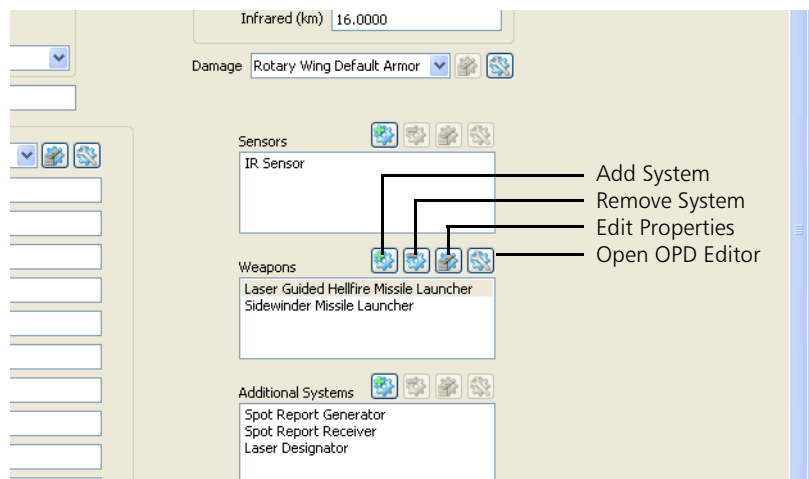


Figure 5-16. Entity systems

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.

5.6.1. 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 5-16). The Add System dialog box opens (Figure 5-17). 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.

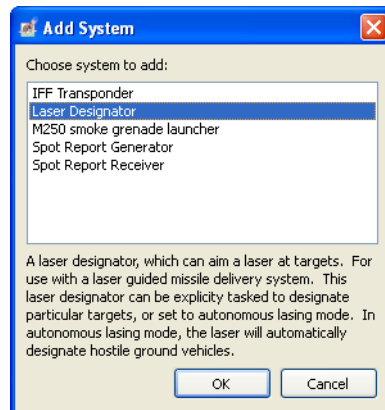


Figure 5-17. Add System dialog box

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

5.6.2. 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.

5.6.3. 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.

5.6.4. Editing a System's Properties

Some systems have properties that you can edit.

To edit a system's properties:

1. 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.
2. Click the Properties button ([Figure 5-16](#)). A Properties dialog box opens.
3. Edit the properties as desired.
4. Click OK.

5.6.5. 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 5-16](#)).

5.6.6. 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 5-16](#)).

5.7. 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.

5.7.1. 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.

5.8. 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**.

5.9. 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.

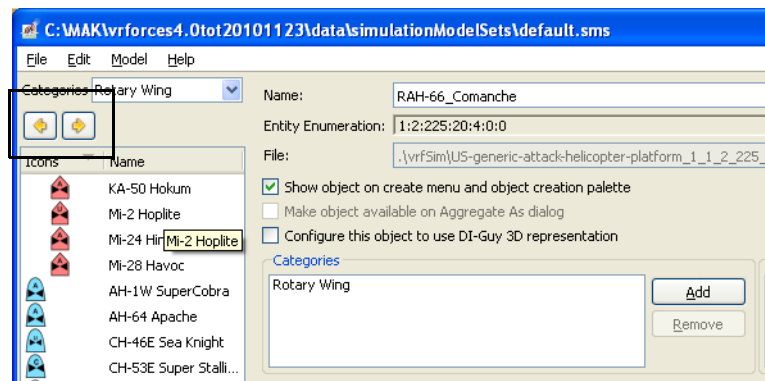


Figure 5-18. Forward and Back arrows

5.10. 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.

i

- ♦ 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.

5.10.1. Generating the Create Menu Automatically

To generate the Create menu automatically:

1. Choose **Edit** → **Edit Create Menu**. The Menu Editor dialog box opens ([Figure 5-19](#)).

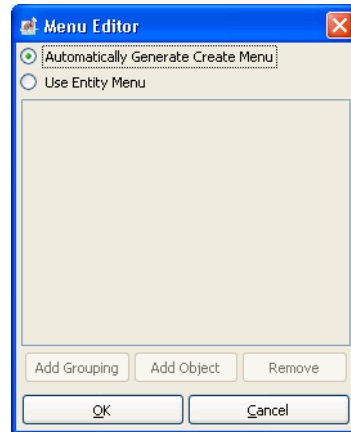


Figure 5-19. Menu Editor

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

5.10.2. 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 5-20](#)).

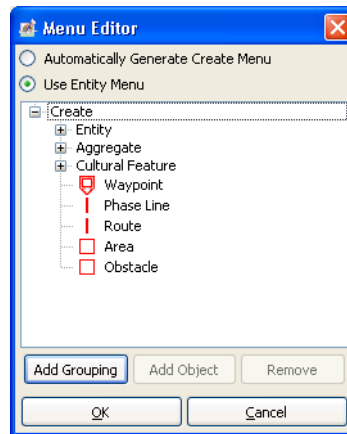


Figure 5-20. Menu Editor

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 5-21](#)).

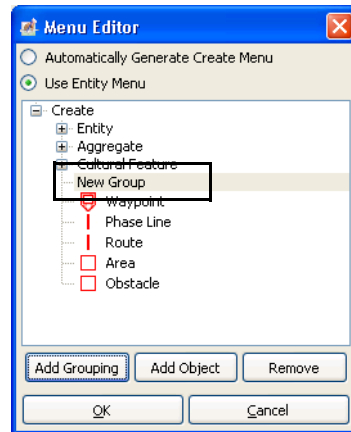


Figure 5-21. Menu Editor with new group added

- 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 5-22](#)).

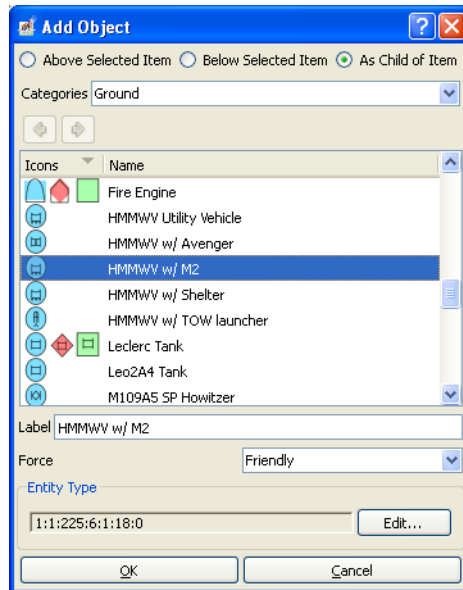


Figure 5-22. Add Object dialog box

- 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 5-23).

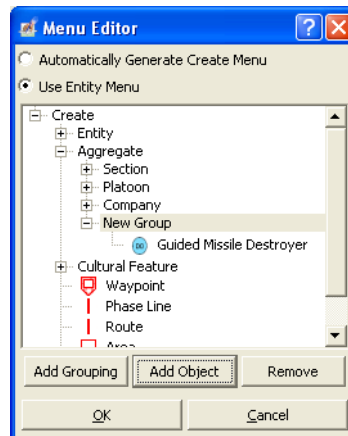


Figure 5-23. Menu Editor with new object added

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.

5.10.3. 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**.

5.11. 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.

5.11.1. Adding a New Category

To add a new category:

1. Choose **Edit → Edit Categories**. The Categories Editor opens ([Figure 5-24](#)).

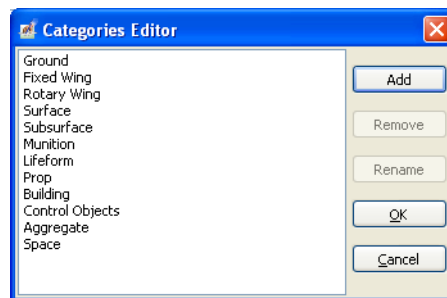


Figure 5-24. Categories Editor

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.

5.11.2. 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.

5.11.3. Renaming a Category

When you rename a category, all entities that are assigned to the old name are re-assigned 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.

5.11.4. 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.

5.12. 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.

5.12.1. Adding a Force

Adding new forces may be useful if you are simulating participants from multiple countries, or if you want to easily model and change force relationships for non-governmental forces such as militias or insurgent groups.

To add a force:

1. Choose **Edit** → **Edit Forces**. The Forces Editor opens (Figure 5-25).

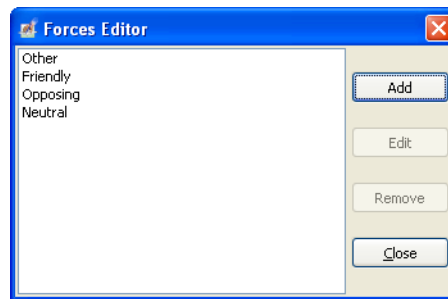


Figure 5-25. Forces Editor

2. Click Add. The Force Hostility Editor opens (Figure 5-26). The new force is assigned a force number. You cannot edit the force number.

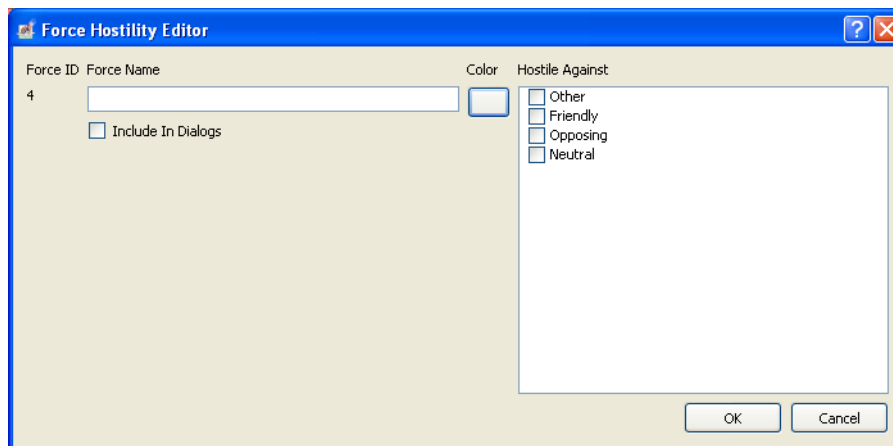


Figure 5-26. Force Hostility Editor

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.

5.12.2. Editing a Force

You can edit a force’s name, color, hostility, and use. Each force has a force number. When you edit a force, you are changing the attributes of that numbered force.

To edit a force:

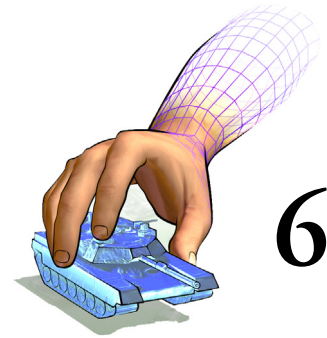
1. Choose **Edit** → **Edit Forces**. The Forces Editor opens ([Figure 5-25](#)).
2. Select the force that you want to edit.
3. Click Edit. The Force Hostility Editor opens ([Figure 5-26](#)).
4. To edit the name, change the text in the Force Name box.
5. To edit the color associated with the force, 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.

5.12.3. 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 5-25](#)).
2. Select the force that you want to remove.
3. Click Remove. The force is removed from the list.
4. Click Close.



Advanced Entity Model Editing

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

The Object Parameter Database Editor	6-2
Starting the OPD Editor	6-3
Loading an Object Parameter Database	6-3
Changing the Object List View	6-4
Editing an Object's Parameters	6-4
Editing Individual Parameters	6-5
Editing Parameters for Multiple Objects	6-6
Deleting Parameters and Components	6-6
Restoring Parameters	6-6
Creating a New Object from an Existing Object	6-7
Adding New Components and Parameters	6-8
Adding New Components	6-8
Editing Components	6-9
Adding New Resource Parameters	6-11
Adding New Subordinate Object Parameters	6-11
Adding New Soil Type Parameters	6-12
Editing Variables	6-12
Editing Systems	6-13
Saving Changes to the Object Parameter Database	6-14

6.1. 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 5-27. Use one tool, save the changes, and close the tool before opening the other.
-

6.2. Starting the OPD Editor

- To start the OPD Editor, choose **Start** → **Programs** → **MÄK Technologies** → **VR-Forces 4.0** → **Tools** → **OPD Editor**. The OPD Editor opens (Figure 6-1).

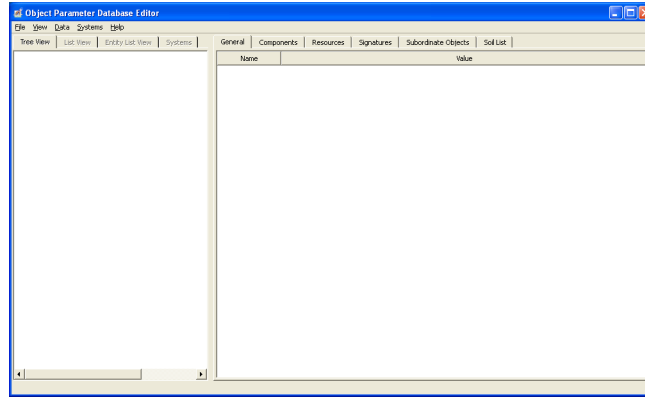


Figure 6-1. OPD Editor

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

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

6.3. 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.

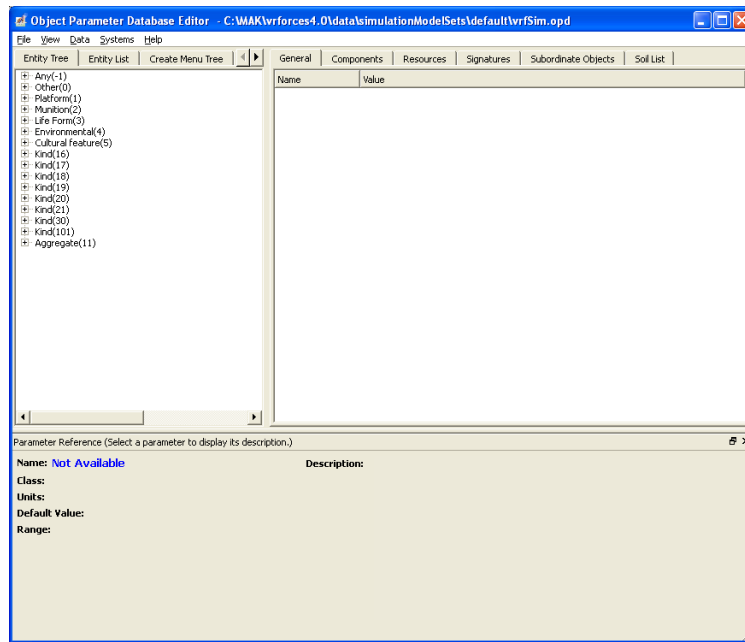


Figure 6-2. OPD Editor with SMS file loaded

6.3.1. 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.

6.4. 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.

6.4.1. 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 6-3).

Name	Value
Controllers	
orq-controller	Priority: 1
component-descript...	component-descriptor
component-type	organization-controller
min-tick-period	-1
min-tick-period-varia...	-1
tick-period-uses-real...	False
process-state-repos...	
process-state-repos...	
is-enabled	True
set-data	Priority: 2
embarkation-controller	Priority: 3
send-radio-message-con...	Priority: 4
contact-fusion	Priority: 5
target-selection	Priority: 6
Systems	
movement	
variable-bindings	
fuel-amount	465
system-definition	\$(system-dir)/movement/ground-tracked.sysdef
is-enabled	True
damage	
sensor	
weapon	
other	
other-2	
other-3	
weapon-2	
Sensors	
Actuators	

Figure 6-3. 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 6-4), 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.

Systems	
movement	
system-definition	\$(system-dir)/movement/ground-tracked.sysdef
variable-bindings	
fuel-amount	465
damage	
system-definition	\$(system-dir)/damage/ground-heavy-armor.svsdef
sensor	

File Chooser

Edit

Restore

Figure 6-4. Parameter value icons

6.4.2. 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.

6.4.3. 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 6-5).

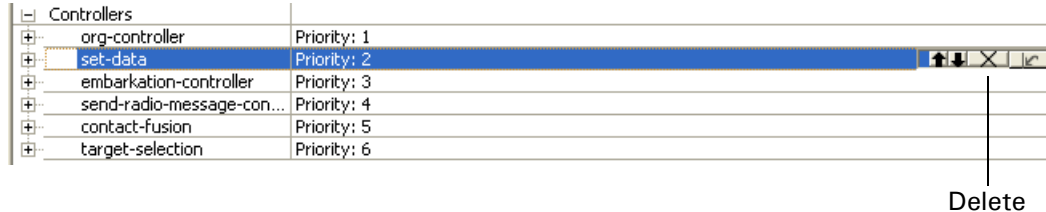


Figure 6-5. Delete parameter icon

6.4.4. 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 6-3). 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 6-4).

6.5. 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 6-6). It lists the object enumeration for the base object in the Base Entry box.

Base Entry		Entry to Create	
1	Platform	1	Platform
2	Air	2	Air
225	United States	225	United States
2	Attack/Strike	2	Attack/Strike
11	Lockheed F-117A	11	Lockheed F-117A
-1	Any	-1	Any
-1	Any	-1	Any

File Name Start: Any

OK Cancel

Figure 6-6. 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.

6.6. 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.

6.6.1. 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 6-7). The Create New Component dialog box opens. The Component list lists all components that are available for this object.

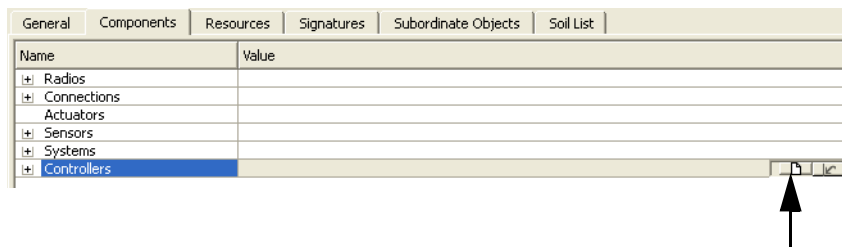


Figure 6-7. Add icon

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

6.6.2. 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 6-8).

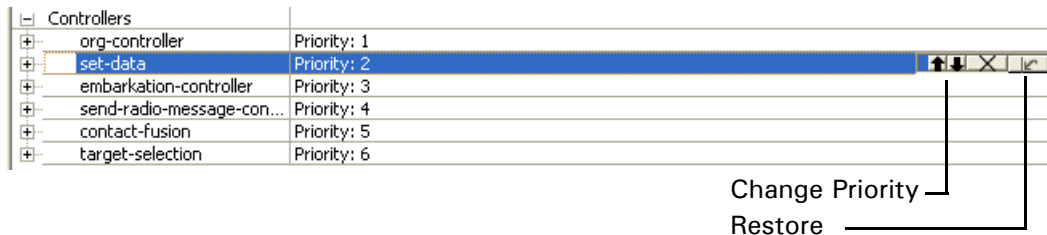


Figure 6-8. Change Priority and Delete Parameter icons

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 6-7). The Create Connection dialog box opens (Figure 6-9).

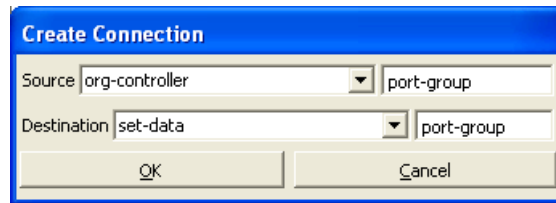


Figure 6-9. Create Connection dialog box

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.

6.6.3. 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 6-7). The Create Resource dialog box opens.

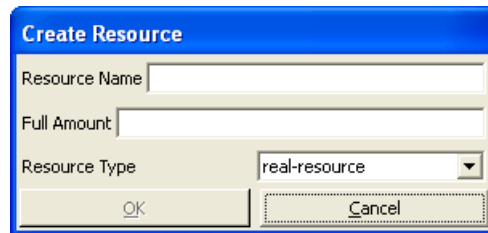


Figure 6-10. Create Resource dialog box

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.

6.6.4. 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 6-7). A subordinate parameter is added.
4. Expand the Subordinate Objects list and edit the parameters for the new subordinate object.

6.6.5. 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 6-7). The Add Soil Type dialog box opens (Figure 6-11).

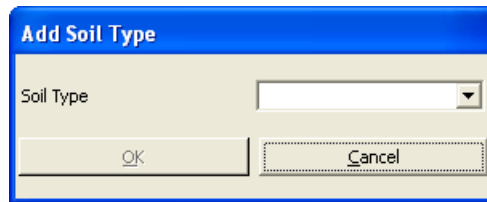


Figure 6-11. Add Soil Type dialog box

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.

6.7. 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 7-22.)

To edit variables:

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

6.8. 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 6-12). Systems are also listed on the Systems List tab of the OPD Editor window.

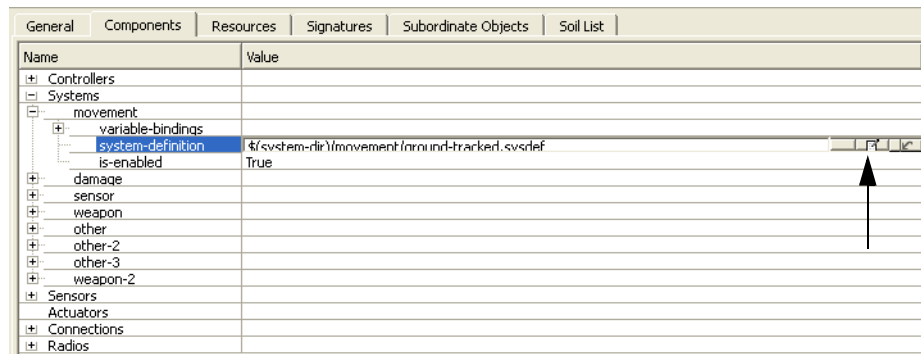


Figure 6-12. Systems list on a Components tab

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.

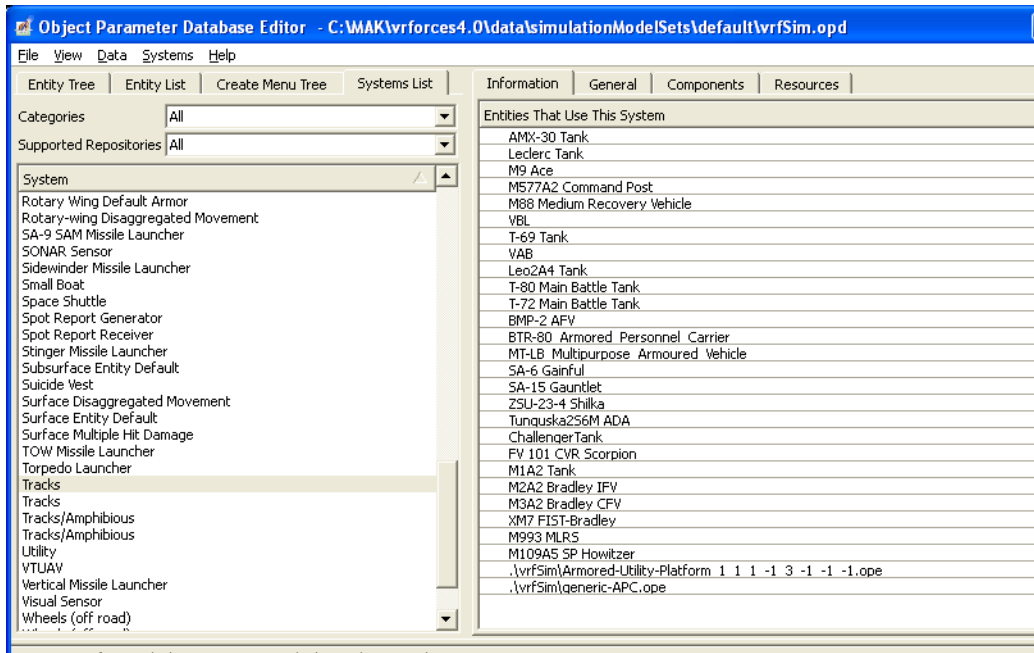


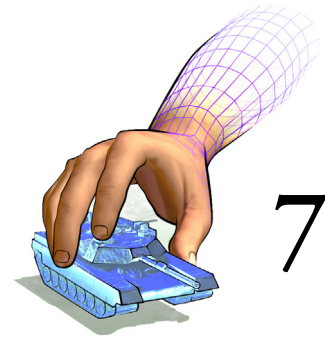
Figure 6-13. System entry in Systems List

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 6-7 and [“Adding New Components and Parameters,”](#) on page 6-8.

6.9. Saving Changes to the Object Parameter Database

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



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.....	7-2
VR-Forces Classes and Object Parameters	7-6
Object Types	7-6
Parameter Types	7-9
Local and Remote Subcomponents.....	7-10
Component Descriptors and Parameters	7-13
Common Elements of Component Descriptors.....	7-14
Configuring the Priority of Components.....	7-16
Adding a New Entity to the Object Parameter Database	7-17
System Definition Files	7-18
Creating a System Definition File.....	7-19
Configuring System Connections	7-19
Adding Variable Bindings	7-22
Configuring Aggregate Entities	7-23
Adding an Aggregate to the Object Parameters Database	7-23
Changing the Echelon ID Category and Echelon ID Level	7-26
Configuring Tactical Graphics	7-27
Pathname Interpretation in Simulation Model Set Files	7-27
Updating Object Parameter Database Files for New Releases	7-28

7.1. 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 *vrjSim.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 7.1 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 7.1

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

```

8      )
9      (formation "Other")
10     (force ForceOther)
11     (category "M1A2");; "Changing the Echelon ID Category and Echelon
12     ID Level," on page 7-26
13     (echelon-level "")
14     (is-organized True)
15     (min-tick-period 0.000000);; 15 Hz ;; "Specifying the Tick Rate
16     for Components," on page 1-9
17     (min-tick-period-variance 0.000000)
18     (dr-algorithm Rvw)
19     (radios
20       (main-radio
21         (radio-descriptor-type "radio-descriptor")
22         (radio-type "default-radio")
23         (comm-model-name "default-radio-model")
24         (publish-transmitter False)
25         (transmitter-params
26           (radio-type 7 1 225 2 1 20)
27           (initial-frequency 3000000.000000)
28           (power 30.000000)
29         )
30       )
31     )
32     (systems
33       (movement
34         (variable-bindings
35           (DtRwReal
36             (fuel-amount 465.000000)
37           )
38         )
39       )
40       (system-definition ;; "System Definition Files," on page 7-18
41         (filename "$(system-dir)/movement/ground-tracked.sysdef")
42       )
43       (is-enabled True)
44     )
45     (sensor
46       (variable-bindings
47         (DtRwVector
48           (sensor-position -1.140000 0.570000 -2.900000)
49         )
50       )
51       (DtRwReal
52         (max-range 4000.000000)
53       )
54     )
55     (system-definition
56       (filename "$(system-dir)/sensors/visual-sensor.sysdef")
57     )
58     (is-enabled True)
59   )
60   (weapon
61     (variable-bindings
62       (DtRwInt
63         (num-ap-rounds 20)
64       )
65     )
66     (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 8-3
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 7-13
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 7-14
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 7-10
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 1-10
158     (net-interface-min-tick-period-variance -1.000000)
159     (task-manager "local-task-manager")
160     (component-manager "component-manager")
161     (plan-manager "vrfoject-plan-manager")
162   )
163   (remote-objects
164     ... )
165   (subordinate-objects )
166   (attached-objects )
167   (user-extension )
168 )

```

7.1.1. 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.

7.1.2. 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 7-1](#) 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*).

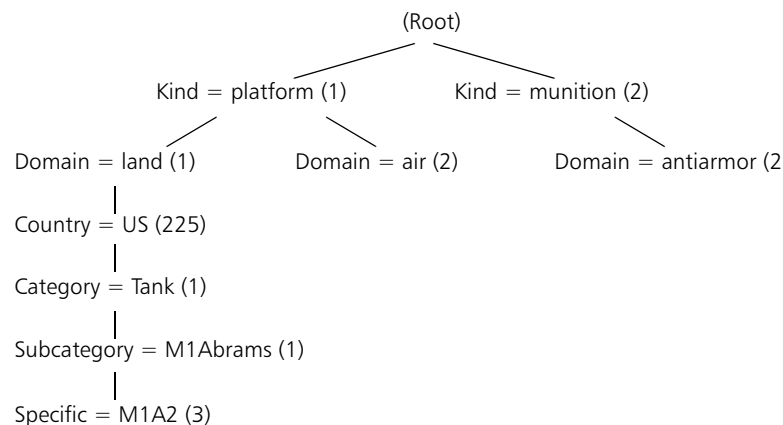


Figure 7-1. Entity type enumeration tree

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 7-2](#)). 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).

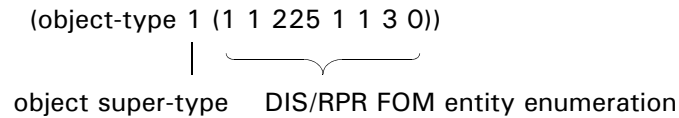


Figure 7-2. VR-Forces object-type parameter

[Table 7-1](#) lists the values VR-Forces can use for the object super-type. For more information, please see *objTypeEnum.h* or “[Object Type Enumerations](#),” on page B-9.

Table 7-1: 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.

7.1.3. 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 7-3](#) illustrates the object parameter type hierarchy.

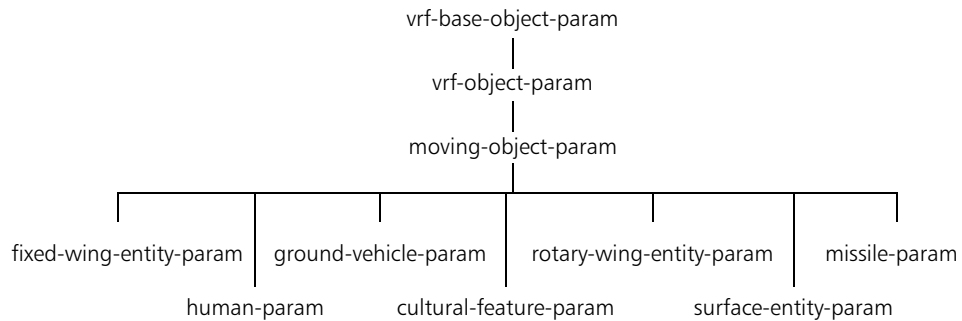


Figure 7-3. Object parameter hierarchy

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 7-12.

7.1.4. Local and Remote Subcomponents

As explained in Section 3.4.2, “[Local Objects and Remote Objects](#)” of *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 7-2](#) maps the subcomponent classes to the parameters that specify them.

Table 7-2: Subcomponent classes and parameters

Class	Parameter
<i>DtSimObjectStateRepository</i>	state-repository
<i>DtSimObjectNetInterface</i>	net-interface
<i>DtOrganizationManager</i>	organization-manager
<i>DtSimTaskManager</i>	task-manager
<i>DtSimComponentManager</i>	component-manager
<i>DtVrfObjectPlanManager</i>	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 7-3](#) lists the permissible entries for each parameter listed in [Table 7-2](#).

Table 7-3: 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 "vrfoobject-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 7-4](#) lists parameter types and the state repository type required.

Table 7-4: 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.

7.2. 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 7.2). 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 7-14 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 7-16.

7.2.1. 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 7.2, 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 7.2, 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 7.2, 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 7.2, 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 7.2 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
```

i

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 7.2 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 7.2

```
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.

7.2.2. 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*.

7.3. 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 5-28) or the OPD Editor (see “[Creating a New Object from an Existing Object](#),” on page 6-7.)



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 7-26.)
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")  
)
```

7.4. 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.

7.4.1. 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.

7.4.2. 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 7-14.) 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.

```
1  (weapon
2    (variable-bindings
3      (DtRwInt
4        (num-ap-rounds 20
5          )
6        )
7      ...
8      (DtRwInt
9        (gun-art-part-type 4416
10          )
11        )
12      ...
13    (system-definition
14      (filename "$(system-dir)\weapons/120mm-gun.sysdef")
15    )
16  )
```

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 7-22.

The system definition file has the following relevant code:

```
1  (weapon-120mm-gun
2    (controllers
3      ...
4    )
5    (actuators
6      (main-gun-joy
7        (component-descriptor-type "ballistic-gun-descriptor")
8        ...
9        (is-enabled True)
10       (art-part-list
11         (primary-gun
12           (art-part-type $gun-art-part-type)
13           (parent-part-type $turret-art-part-type)
14           (attach-point $turret-attach-point)
15           (art-part-param-list
16             (part-type 13)
17             (part-type 14)
18           )
19         )
20       )
21     )
22   )
23   (connections
24     (connect system:target-list turret-acquire:target-list)
25     (connect main-gun-control-joy:gun-control main-gun-joy:gun-
26       control)
27     ...
28   )
29   (meta-data
30     ...
31     (parameter-data-list
32       (art-part-parameter-data
33         (parameter-name "gun-art-part-type")
34         (variable-type "DtRwInt")
35       )
36     )
37   )
```



```

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 7-4) illustrates these connections.

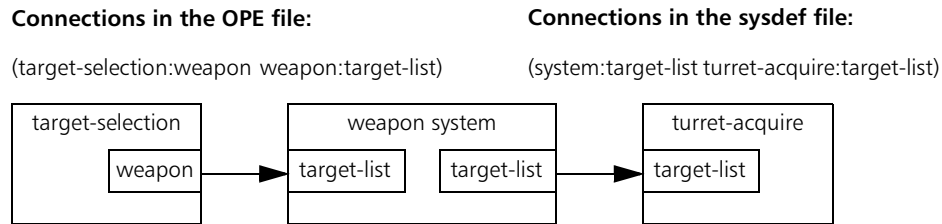


Figure 7-4. Flow of data in component connection

7.4.3. 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.

7.5. 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.

7.5.1. 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 6, *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 *vrSim.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*.

7.6. 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.



You can also add new entries with the OPD Editor. For details, please see Chapter 6, *Advanced Entity Model Editing*.



- ♦ 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.
-

7.7. 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.

7.8. 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):

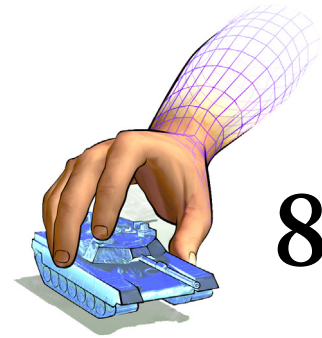
```
(systems
  (tracked-movement
    (variable-bindings
      (DtRwReal
```

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

7.9. 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.



Configuring Object Behaviors

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

Configuring Sensors.....	8-3
Configuring a Sensor in an OPE File.....	8-3
Editing a Sensor System Definition File.....	8-4
Specifying a Sensor Signature	8-5
Adding a New Sensor Type.....	8-5
Detection Probability Tables	8-9
Probability and Selection Tables for Munitions	8-9
Ammo Select Tables	8-10
Hit Probability Tables.....	8-11
Damage Probability Tables	8-13
Configuring an Entity to Take Damage from a Munition.....	8-14
Configuring Formations.....	8-15
Formation Files	8-15
Creating a User-Defined Formation	8-16
Configuring Emitters	8-17
Configuring Radios.....	8-19
Configuring Sensor Volumes.....	8-20
Configuring Sensor Volume Color	8-20
Controlling Sensor Volume Radius.....	8-21
Configuring Sensor Volume Segments.....	8-22
Configuring Task Execution Rules	8-25
Configuring Joysticks	8-26
Configuring Obstruction Avoidance	8-27

Configuring Object Behaviors

Configuring Object Types	8-27
Configuring Vector Feature Types	8-28
Configuring Embarkation	8-29
Editing Embarkation Parameters	8-30
Configuring VR-Forces Components on Remote Entities.....	8-32

8.1. 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*.

8.1.1. 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)
```

8.1.2. 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.

8.1.3. 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.

8.1.4. 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 8-3. 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.

8.2. 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 8-1](#), 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	I	2	1	1	2	2	3	3	3	4	
6	V	2.5	1	2	2	3	3	3	4	4	
7		3	1	2	3	3	3	4	4	4	
8		4	2	3	3	3	4	4	4	4	
9											

Figure 8-1. 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*.

8.3. 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).

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

8.3.2. 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”.

i

- ♦ 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)
    )
   )
  )
 )
)
```

8.3.3. 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 8-11. The damage actuator provided by MÄK looks for a probability entry with the name **catastrophic-kill**.

i

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.

8.3.4. 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.

8.4. 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")
   )
  )
 )
 ...
)
```

8.4.1. 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.

8.4.2. 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.

8.5. Configuring Emitters

All fixed-wing aircraft have emitters as part of their radar sensor. 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 is as follows:

```
(active-radar-sensor-system
  (systems )
  (sensors
    (radar-sensor
      (component-descriptor-type "radar-signature-sensor-descriptor")
      (component-type "radar-signature-sensor")
      . . .
      (emitter-system
        (system-name-enum 5570)
        (system-function-enum 5)
        (beam-list
          (beam-0
            (index 0)
            (type "track")
            (frequency 4999999488.000000)
            (frequency-range 5000.000000)
            (power 70.000000)
            (pulse-repetition-frequency 0.000000)
            (pulse-width 0.000000)
            (azimuth-center 0.000000)
            (azimuth-sweep 3.142857)
            (elevation-center 0.000000)
            (elevation-sweep 1.000000)
            (sweep-sync 0.000000)
            (beam-function-enum 5)
            (beam-param-index 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 beams on an emitter, add additional beam-list blocks and change the name of each block. For example:

```
(radar-sensor
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type "radar-signature-sensor")
  . . .
  (emitter-system
    (system-name-enum 5570)
    (system-function-enum 5)
    (beam-list
      (beam-0
        ...
      )
      (beam-1
        ...
      )
    )
  )
)
```

8.6. 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)
  )
)
```

8.7. 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.

8.7.1. 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 8-1](#) describes the sensor volume visualizer attributes that affect sensor volume color.

Table 8-1: 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.

8.7.2. 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 );
```

8.7.3. 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°.

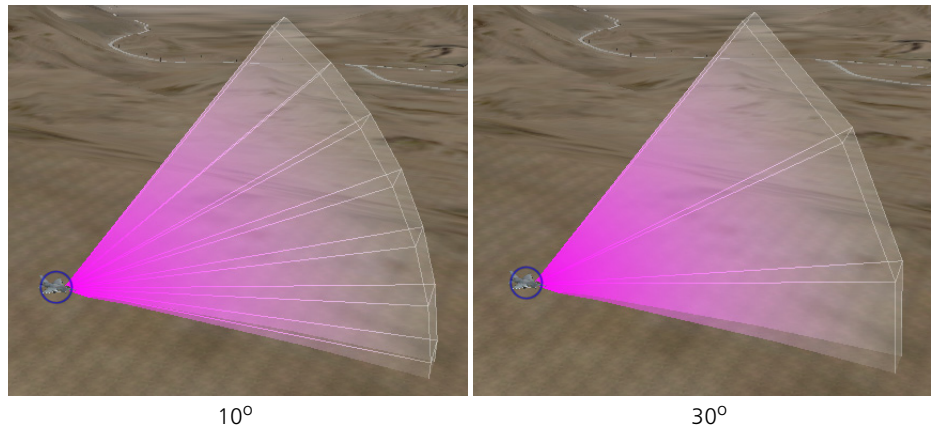


Figure 8-2. Sensor volume segments

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.

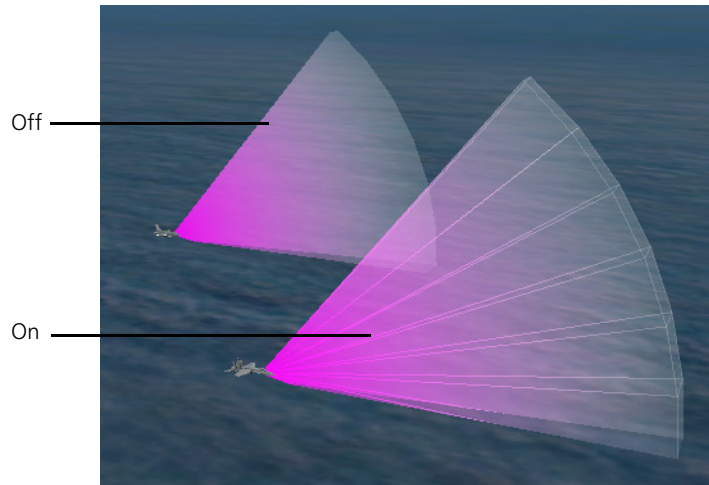


Figure 8-3. Segment outlines

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 Sensor Volume Visualizer for the entity.
12. Click the Add Attribute icon (). The Add Visual Attribute dialog box opens.
13. Select modelDefinition.
14. Click OK. The modelDefinition attribute is added to the Sensor Volume Visualizer.
15. Click the Value column for the new modelDefinition attribute. The Model Definition for Visualizer dialog box displays a list of model definitions.
16. Select the model definition that you just created.
17. Click OK.
18. If you are connected to an exercise, disconnect and then reconnect to enable the new model definition.

8.8. 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/simulationModelSets/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.

8.9. 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 2000 or XP
 - `pre-direct-input-joydevice` – used on Windows NT, for joysticks that do not use DirectInput.
 - `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.

i

- ♦ 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)
)
```

8.10. 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.

8.10.1. 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
)
```

8.10.2. 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 Chapter 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)
  )
))
```

8.11. 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.

8.11.1. 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 8-4).

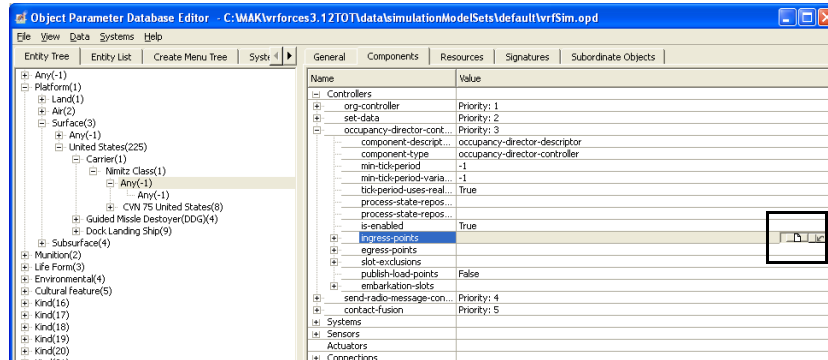


Figure 8-4. occupancy-director-controller component

3. Click the New button. The Set Load Points dialog box opens (Figure 8-5).

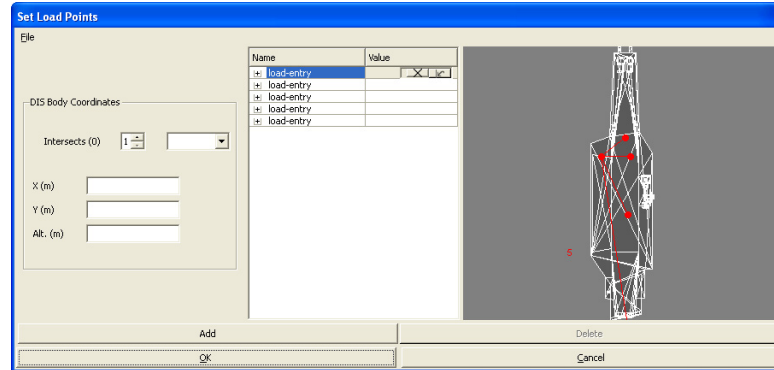


Figure 8-5. Set Load Points dialog box

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

8.12. 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")
  )
)
```

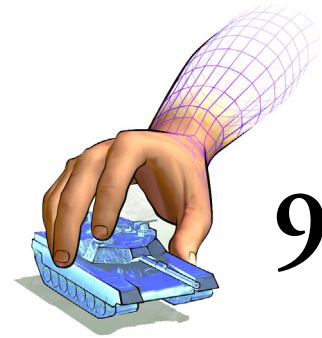
The entries in this table associate the key-names mig29-config and f18-config with appropriate entity type enumerations. You can also associate a key-name with marking text, as follows:

```
(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.



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	9-2
Configuring VR-Forces.....	9-2
Enabling Radio Transmitter Publishing	9-3
Enabling the External Communication Model	9-3
Configuring the External Communications Model.....	9-4
Using VR-Forces with Qualnet	9-5
Running VR-Forces with Qualnet	9-5
Example Scenario.....	9-6
Running the Scenario	9-6

9.1. 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.

9.2. 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 through the communications effects server
- ♦ Set the external communications model as the communications model to be used by VR-Forces
- ♦ Configure the external communications model.

9.2.1. Enabling Radio Transmitter Publishing

The external communications model requires that any entities sending or receiving messages through 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 9-3.

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)
    )
  )
)
```

9.2.2. 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/simulationModelSets/<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.

9.2.3. 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.

9.3. 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 9-2. 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.

9.3.1. 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.
-

9.4. Example Scenario

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

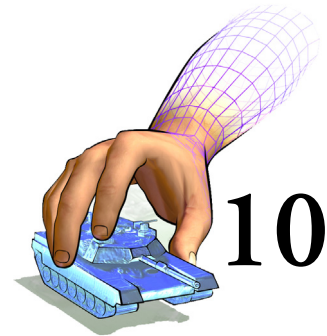
In the scenario, the M1A2 1 tank commands the UAV to fly over the area where the opposing forces are and send back spot reports. When the first tank receives spot reports, it commands the second one to move to the waypoint, and also moves there itself. All commands and spot reports are sent through the radio, which is passed through 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.

9.4.1. 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.



Display Configurations

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

The Display Engine	10-2
Window Types	10-2
Adding a Window	10-3
Adding a Channel to a Window	10-5
Removing a Window	10-5
Removing a Channel	10-6
Saving a Display Configuration	10-6
Loading a Display Configuration	10-6
Changing a Window's Attributes	10-6
Changing a Channel's Attributes	10-8
Setting the Clipping Planes	10-10
Specifying the Projection Resize Policy Attribute	10-11
Changing a Channel's Frustum (Field of View)	10-13
Changing the Viewport	10-14
Configuring Multichannel Displays	10-15
Changing the Camera's Position and Orientation Offset	10-17
Stereoscopic Displays	10-18
Configuring Anaglyphic Stereo	10-19
Configuring Polarized Stereo	10-19

10.1. 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 configuration. You can save display configurations and can load them to quickly recreate a particular configuration.

10.1.1. Window Types

VR-Forces supports the following types of windows:

- ♦ Full screen. A full screen window uses the entire area of a monitor. It does not have any menus, panels, or other GUI controls.
- ♦ Resizable. A resizable window can be moved, resized, maximized, and minimized. It does not have any GUI controls. An inset view is an example of a resizable window. (For information about inset views, please see [“Inset Views,”](#) on page 5-21.)
- ♦ 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.

10.1.2. 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 Configuration Editor.

To add a window from the File menu:

1. Choose **File** → **Stealth Window** (or **Plan View Window**). A new window opens (Figure 10-1).

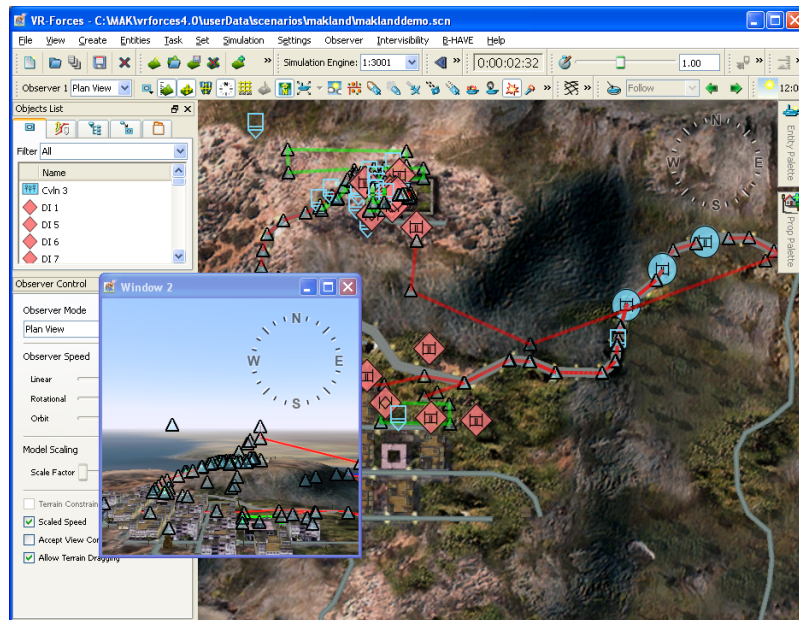


Figure 10-1. New window

To add a window from the Display Configuration Editor:

1. Choose **View** → **Display Configuration Panel**. The Display Configuration Panel is added to the VR-Forces window (Figure 10-2).

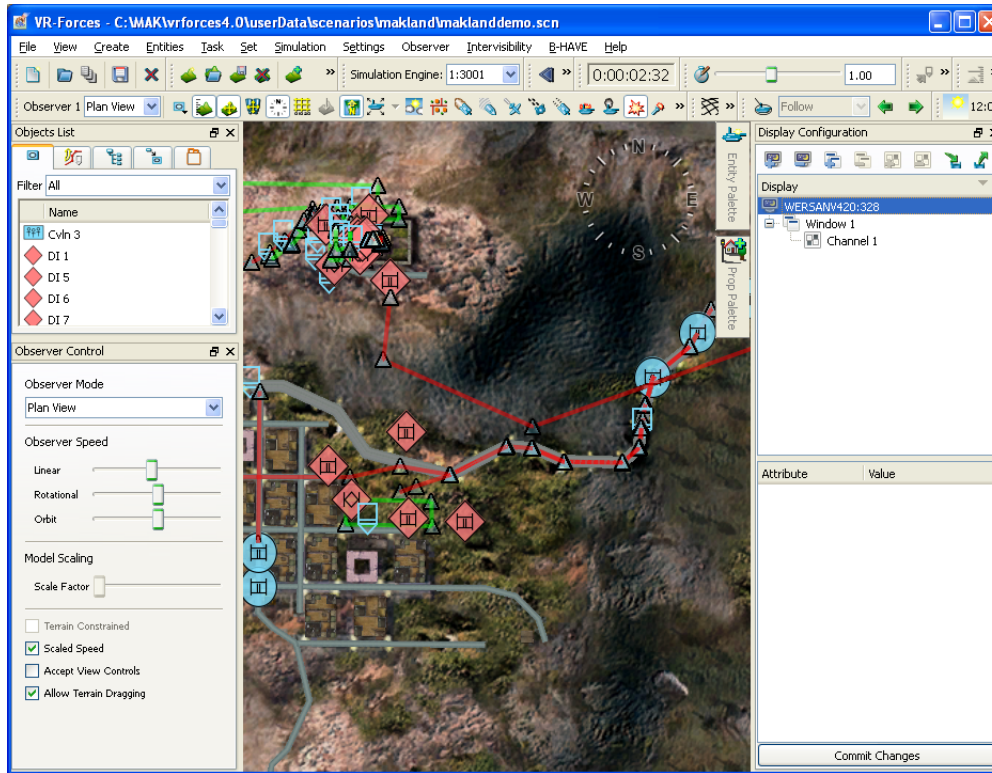


Figure 10-2. Display Configuration Panel

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 Configuration Panel (Figure 10-2).

The Display Configuration Panel

You can manage display engines, windows, and channels on the Display Configuration Panel (Figure 10-2).

- ♦ Connect (1) and disconnect (2) a display engine (not supported in VR-Forces)
- ♦ Add (3) and remove (4) a window
- ♦ Add (5) and remove (6) a channel
- ♦ Save (7) and load (8) a display configuration.



Figure 10-3. Display Configuration Panel icons

10.1.3. 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 Configuration 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 10-11). For details about editing channel attributes, please see “[Changing a Channel’s Attributes](#),” on page 10-8.

10.1.4. Removing a Window

To remove a window:

1. In the Display Configuration 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.

10.1.5. Removing a Channel

To remove a channel:

1. In the Display Configuration 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.

10.1.6. Saving a Display Configuration

You can save a display configuration so that you can easily replicate the configuration at a later time.

To save a display configuration:

1. Click the Save Display Configuration icon () , or right-click the name of the display engine and choose **Save Display Configuration** on the menu. The Save Display Configuration dialog box opens.
2. Type a name for the display configuration.
3. Click Save.

10.1.7. Loading a Display Configuration

To load a display configuration:

1. In the Display Configuration Panel, select the display engine at the top of the window.
2. Click the Load Display Configuration icon () , or right-click the name of the display engine and choose **Load Display Configuration** on the menu. The Load Display Configuration dialog box opens.
3. Select the display configuration you want to load.
4. Click Open.

10.2. 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 [“Window Types,”](#) on page 10-2)
- ♦ Screen Number
- ♦ Position (X, Y coordinates of upper left corner, in screen pixels)
- ♦ Width and height, in pixels
- ♦ Fixed size.

To change a window's attributes:

1. Select the window in the Display Configuration Panel. A list of attributes is displayed at the bottom of the editor (Figure 10-4).

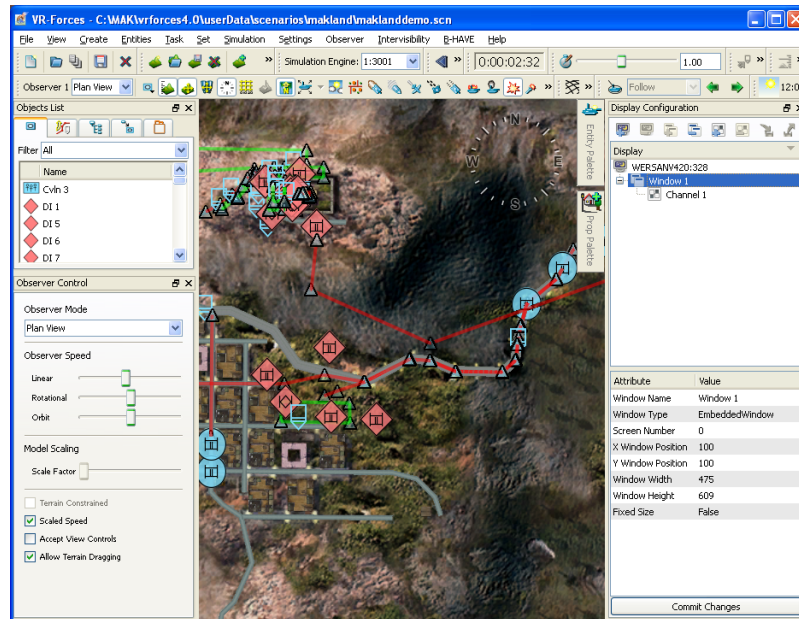


Figure 10-4. Window attributes

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 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 Configuration Panel.

10.3. 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 Configuration Panel. A list of attributes is displayed at the bottom of the editor ([Figure 10-5](#)).

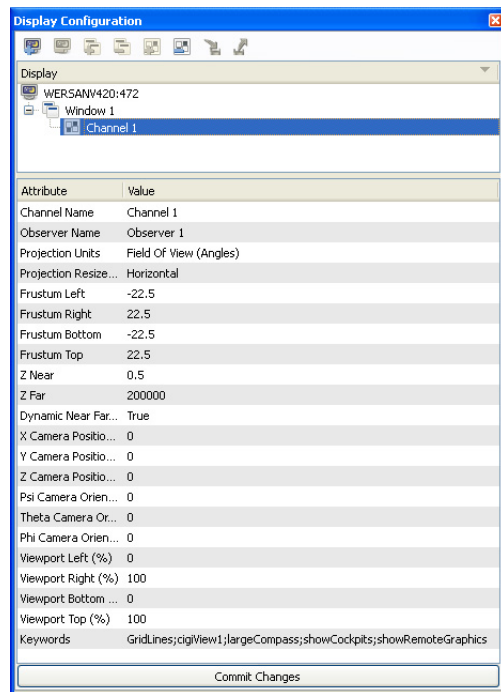


Figure 10-5. Channel attributes

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.

10.3.1. 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 10-6](#) illustrates the effect of changing the Z Near value. [Figure 10-7](#) illustrates the effect of changing the Z Far value.

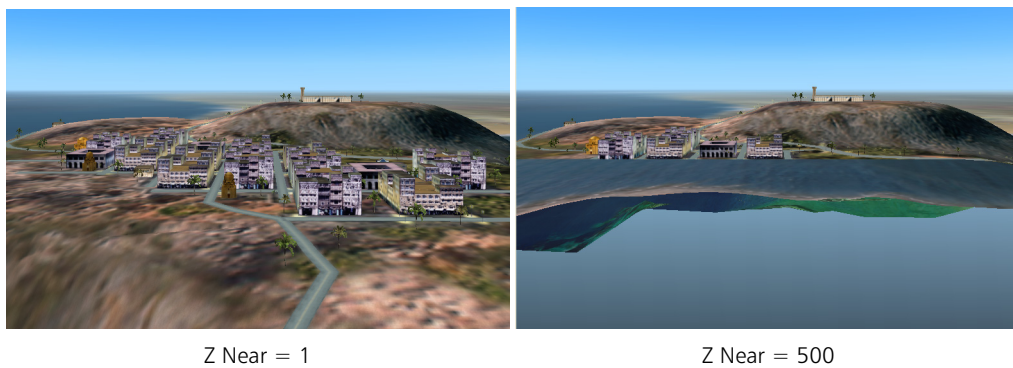


Figure 10-6. Near clipping

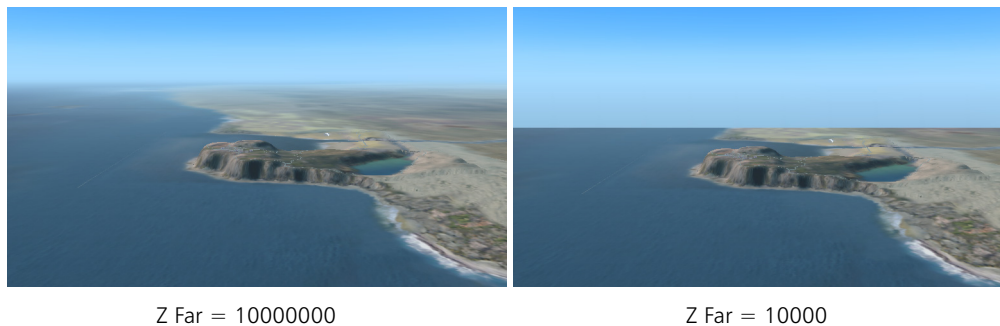


Figure 10-7. Far clipping

- 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.

10.3.2. 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 10-9 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.

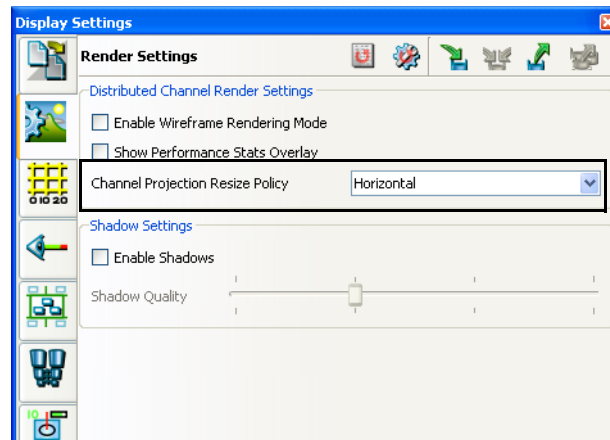


Figure 10-8. Display Settings, Render Settings page

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 10-9. Effect of different projection resize policies

10.3.3. 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 10-10 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.

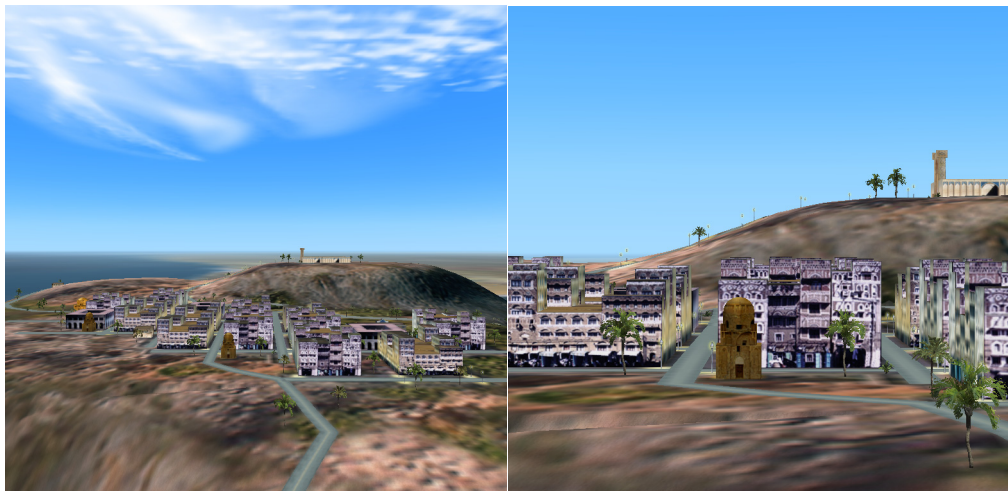


Figure 10-10. Field-of-view

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

10.3.4. 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 10-11](#) 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.

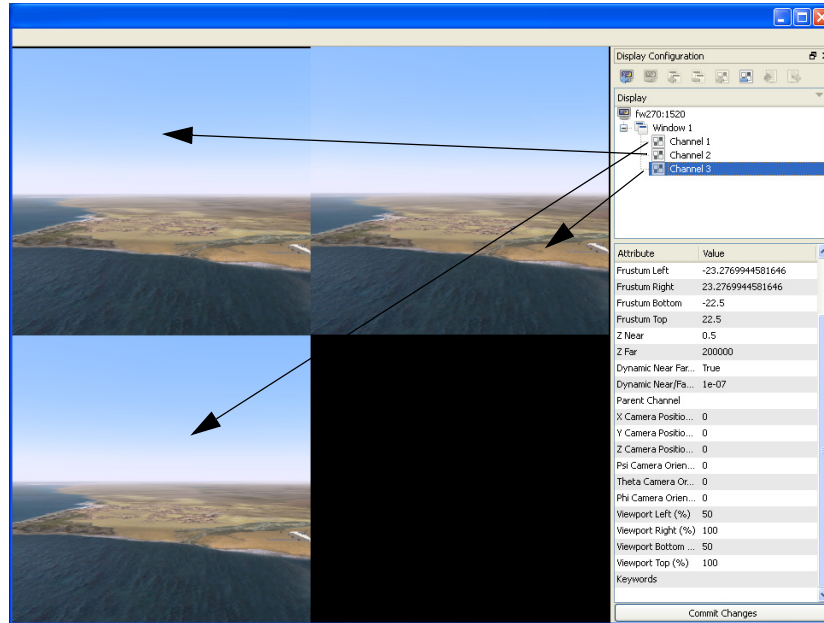


Figure 10-11. Window with three channels and three viewports

Viewports are specified as a percentage of the window from left to right and top to bottom. [Figure 10-12](#) shows the percentage allocation for the viewports in ([Figure 10-11](#)).

0		50%		100%	
Left:	0	Left:	50	50%	
Right:	50	Right:	100		
Top:	100	Top:	100		
Bottom:	50	Bottom:	50		
Left:	0	Left:	50	0	
Right:	50	Right:	100		
Top:	50	Top:	50		
Bottom:	0	Bottom:	0		

Figure 10-12. Viewport configuration



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

10.4. 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 10-13).

You could set up these display configurations by having one computer with several monitors attached, or you could have multiple computers, each running a VR-Forces front-end. You might also want to have a 2D view on one monitor and a 3D view on another.

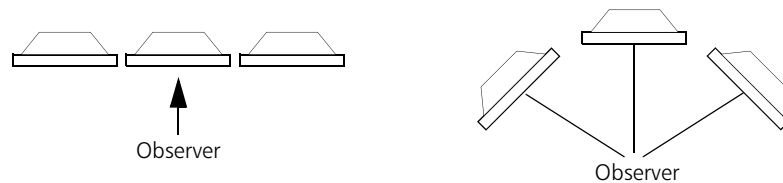


Figure 10-13. Multi-channel displays (top-down view)

By default, all windows and channels show the same view of the terrain (Figure 10-15, default window views). Therefore, if you want a setup in which each channel shows a different view of the terrain, you must configure the channels to calculate an offset from the master view.

The distribution of the view across multiple channels is controlled by the frustum. Figure 10-14 illustrates the default frustum for Channel 1. It has a 60° field of view, from -30° to +30°.

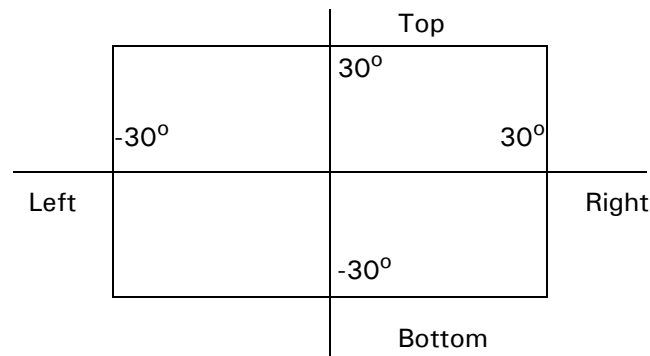


Figure 10-14. Default frustum for Channel 1

If you wanted to split the view in [Figure 10-14](#) into, for example, three horizontal channels, you would edit the left and right frustum values for each channel, as shown in [\(Figure 10-15\)](#). The field of view is the same: -30° to $+30^{\circ}$. However, now each channel shows a 20° segment of the total field of view.

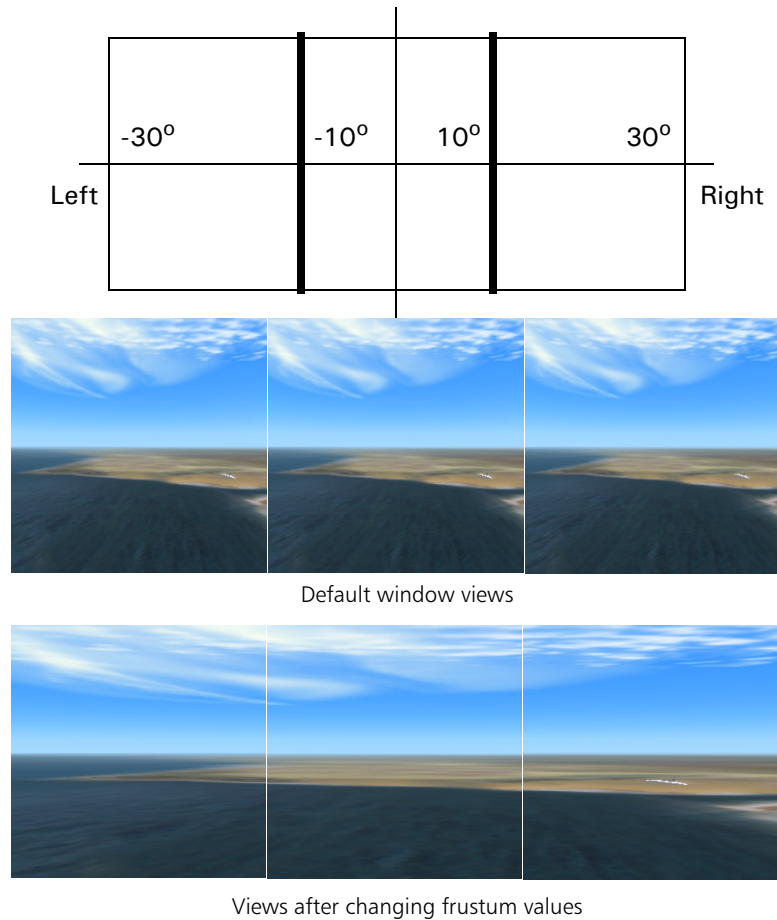


Figure 10-15. Frustum values for three channels

10.4.1. 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 10-16](#). (The camera location is represented by the binoculars.)

If you wanted to show the view as experienced by a gunner riding in the back of the vehicle, or in a turret, you would change the X, Y, and Z camera position offsets in the channel's attribute list (vehicle 2).

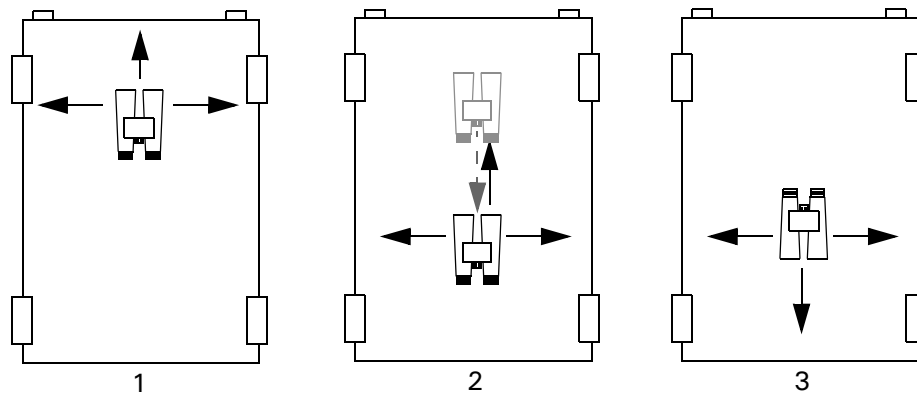


Figure 10-16. Position and orientation offsets

If you wanted the gunner to look out the rear of the vehicle, you would change its orientation, with the Psi (heading), Theta (pitch), and Phi (roll) Camera Orientation Offsets (vehicle 3).

10.5. 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. OpenScene-Graph can support all three of these methods, and since VR-Forces uses OpenScene-Graph 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 OpenScene-Graph 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.

10.5.1. 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 10-1](#).

Table 10-1: 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, please see the OpenSceneGraph documentation at <http://www.openscenegraph.org/projects/osg/wiki/Support/UserGuides/StereoSettings>.

10.5.2. 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 10-17](#) shows an example configuration.

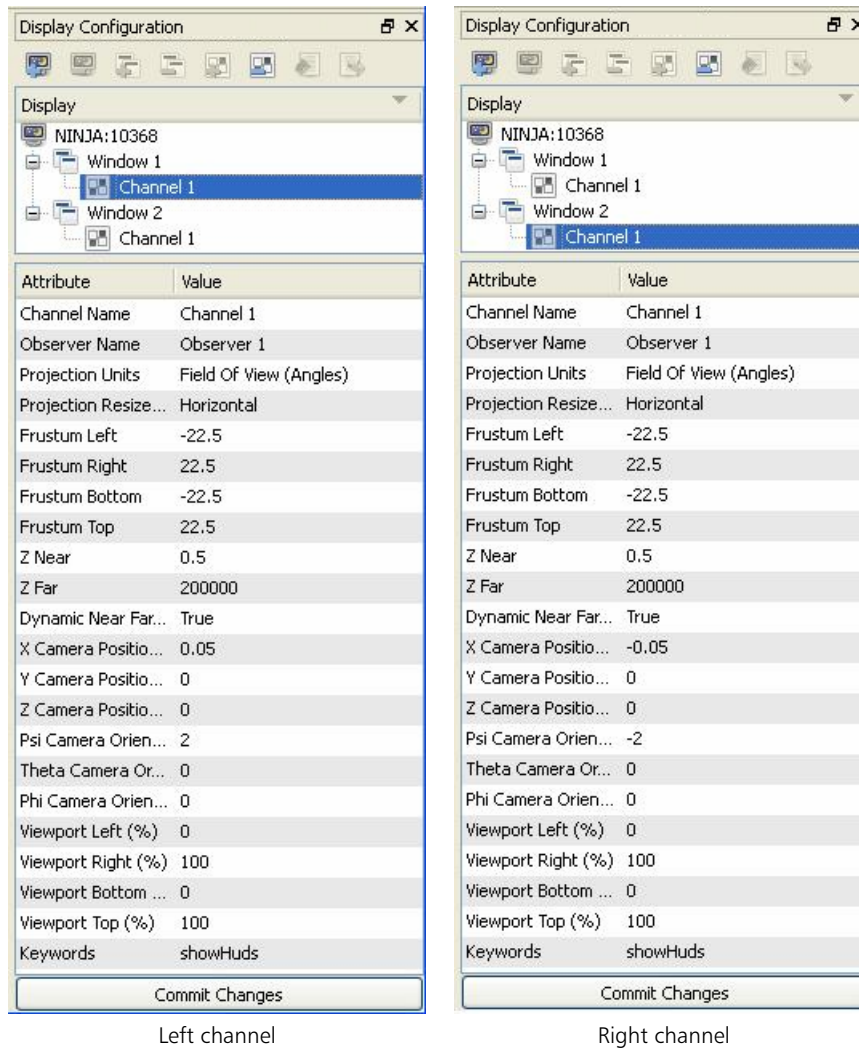
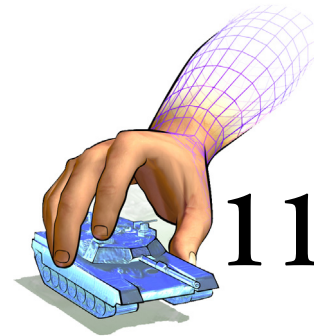


Figure 10-17. Polarized stereo channel configurations

To compute the correct channel configuration values:

1. Pick a focal length, L . Objects at this distance will appear to be in the plane of the display. Objects that are further away will appear behind the display, and objects closer will appear in front of it.
2. Pick an eye spacing, E , to represent the distance between the viewer's eyes. Human eyes are typically about 0.063m apart (<http://www.cl.cam.ac.uk/~nad10/pubs/EI5291A-05.pdf>).
3. Set the X Camera Position values to $-E/2$ meters for the left channel and $E/2$ meters for the right channel.
4. Set the Psi Camera Orientation values to $-\arctan(E/2)/L$ degrees for the left channel and $\arctan(E/2)/L$ degrees for the right channel.



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 3-28, 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	11-3
Considerations and Limitations for Building Terrains.....	11-3
Saving a Terrain.....	11-4
Adding Elevation Data (Terrain Patches) to a Terrain.....	11-4
Adding Images to a Terrain	11-6
Changing the Display Order of Raster Maps	11-9
Adding a Feature Layer	11-10
Connecting to Terrain Servers.....	11-11
Adding Terrain Server Connections.....	11-13
Editing a Terrain Server Configuration.....	11-14
Loading MetaFlight Terrains.....	11-15
Configuring MetaFlight File Loading.....	11-15
Manually Paging-In Terrain	11-18
Adding Props to a Terrain	11-19
Extracting Props from a Terrain Patch	11-19
Adding Props from a Feature Layer.....	11-21
Viewing a List of Props.....	11-23
Selecting Props	11-23
Setting the Opacity of Props.....	11-24
Changing a Prop’s Type	11-24
Changing a Prop’s Position or Orientation.....	11-25
Changing a Prop’s Model Definition	11-25

Configuring File Caching	11-26
Enabling Texture Compression.....	11-26
Clearing the File Cache	11-27
Displaying DDS Textures Correctly.....	11-27
Flipping DDS Textures Globally	11-27
Displaying Information About Paged and Streaming Terrains	11-28
Displaying Additional Terrain Paging Information	11-29
Configuring Terrain Paging for the Back-End	11-30
Building Efficient MetaFlight Terrains for VR-Forces	11-31

11.1. Creating a Composed Terrain

“[Composing a Terrain](#),” on page C-2 demonstrates how to create a composed terrain.

The general procedure for composing a terrain is as follows:

1. Choose **File** → **New Terrain**. 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 11-4.
3. Optionally, add imagery, as described in “[Adding Images to a Terrain](#),” on page 11-6.
4. Optionally, add feature data, as described in “[Adding a Feature Layer](#),” on page 11-10.
5. Optionally, add props or extract props, as described in “[Adding Props from a Feature Layer](#),” on page 11-21 and “[Extracting Props from a Terrain Patch](#),” on page 11-19.
6. Save the terrain, as described in “[Saving a Terrain](#),” on page 11-4.

11.1.1. 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.

11.1.2. 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 11-26.) 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**. The Save Terrain dialog box opens.
2. Type a name for the terrain.
3. Click Save.

11.2. 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 11-3.



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.



When you add a terrain patch, the settings on the Display Settings dialog box, Loader Settings page get applied to the terrain. The Flip DDS Textures setting is relevant to all terrains. The other settings only affect MetaFlight terrains. For details about these settings, please see “[Displaying DDS Textures Correctly](#),” on page 11-27 and “[Loading MetaFlight Terrains](#),” on page 11-15.

To add a terrain patch to a terrain database:

1. Choose **Settings** → **Terrain**. 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 11-1).

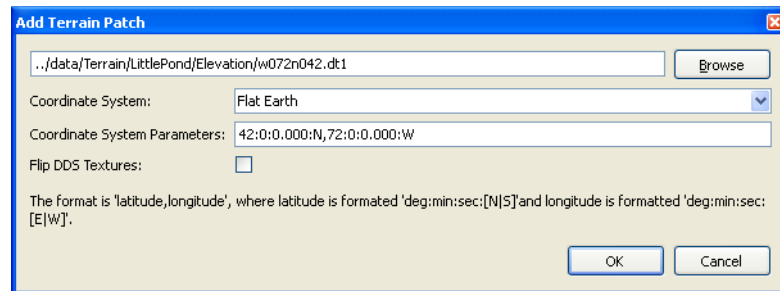


Figure 11-1. Add Terrain Patch dialog box

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.

11.3. 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 (Figure 11-2).

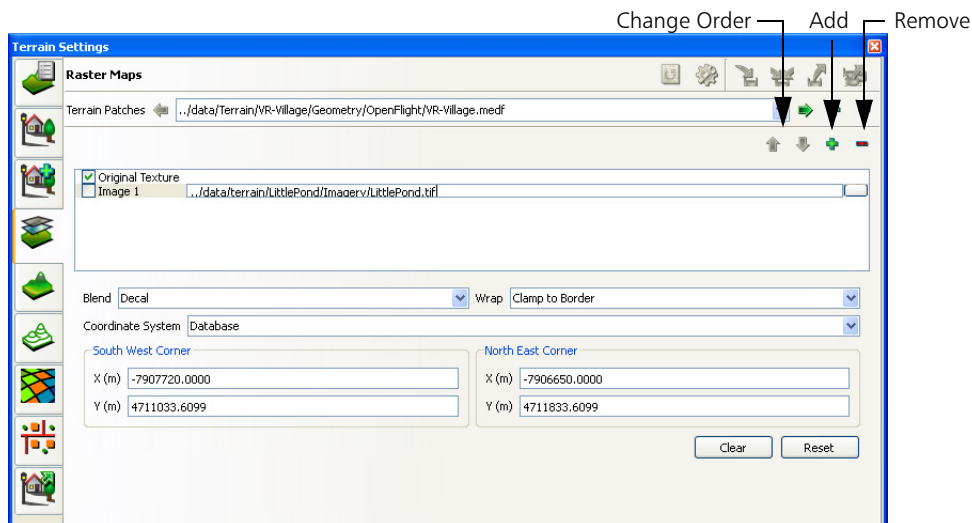


Figure 11-2. Raster Maps page

3. Click the Add icon (+) that is above the list of raster maps. A line is added to the Raster Maps list (Figure 11-2).
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 11-3).

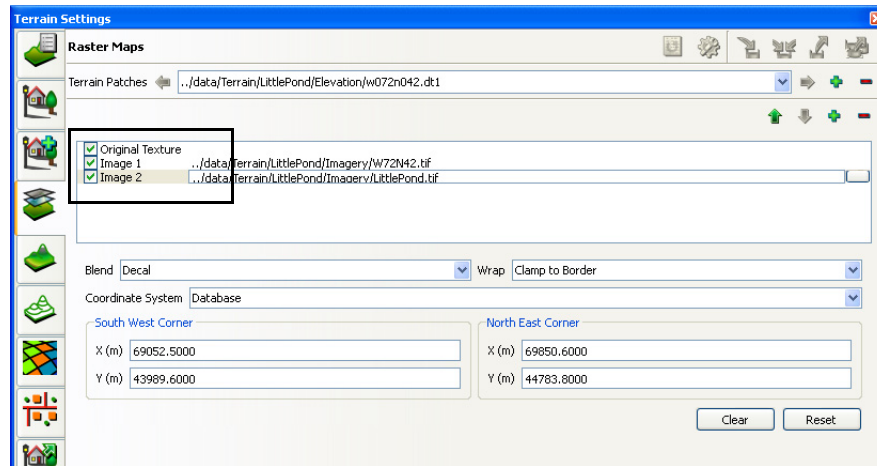


Figure 11-3. Making an image visible

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 11-4):
 - 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 Geo TIFF, 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.

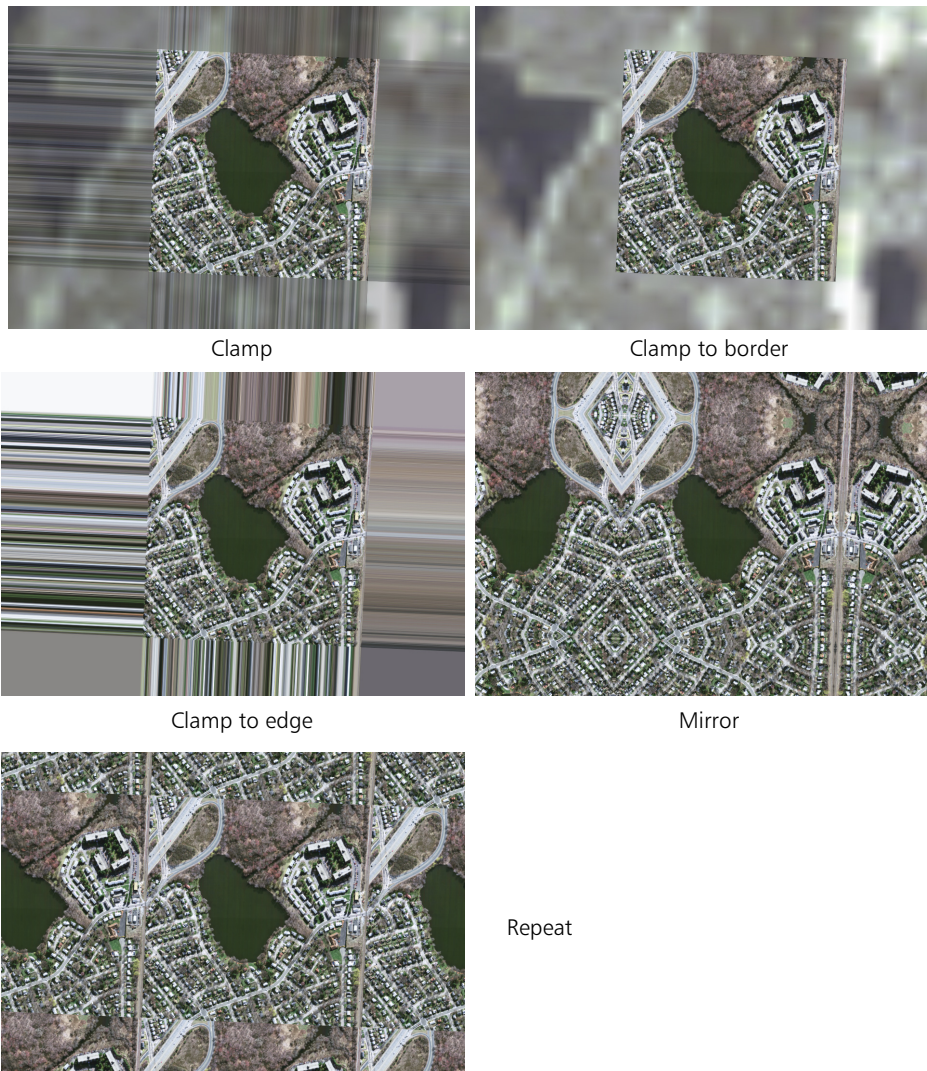


Figure 11-4. Raster image wrap effects

11.3.1. 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 11-5](#) 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.

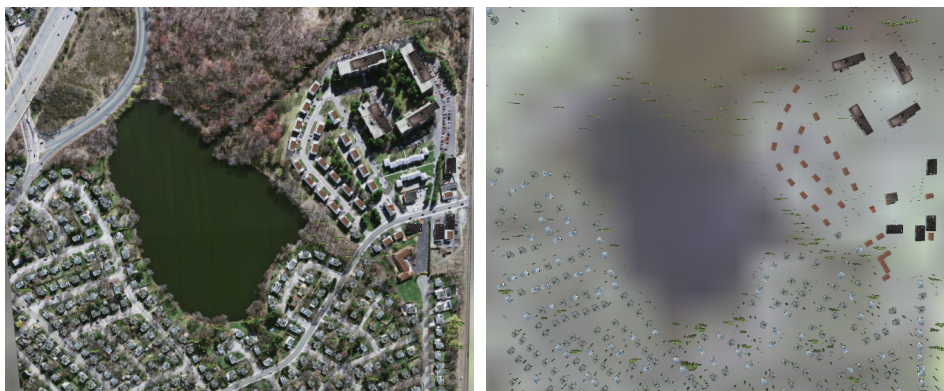


Figure 11-5. Changing the raster map order

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 11-2](#)).
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.

11.4. 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).

i

- 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 11-19.
- 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 11-6](#)).

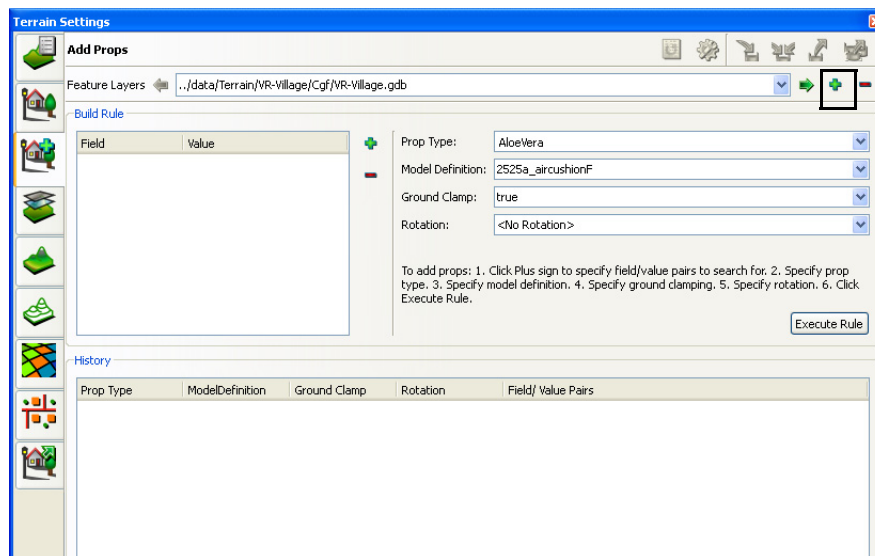


Figure 11-6. Feature Layer Settings dialog box

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.

11.5. Connecting to Terrain Servers

VR-Forces supports the osgEarth libraries for creating terrains from remote or local data sources (Figure 11-7). Terrain servers support streaming of elevation and image data and are, therefore, ideal for displaying terrains that have large amounts of data.

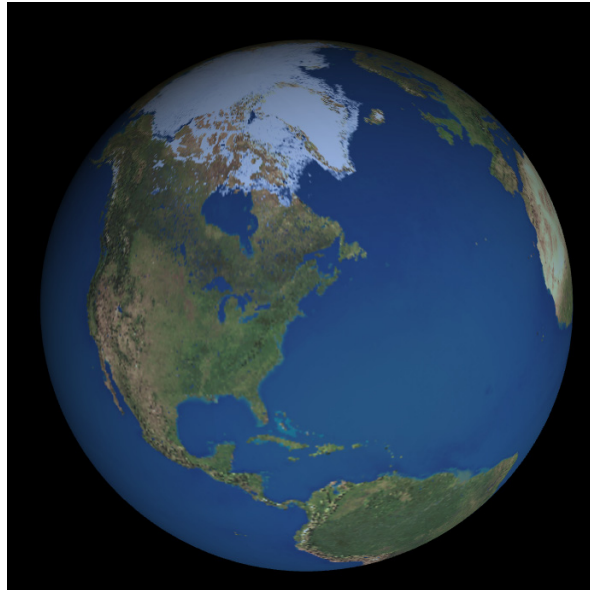


Figure 11-7. Geocentric terrain

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*. 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 that serves local data.

i

When you view a geocentric terrain and zoom out more than 6 million meters or so, 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 **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Terrain Contents page ([Figure 11-8](#)).

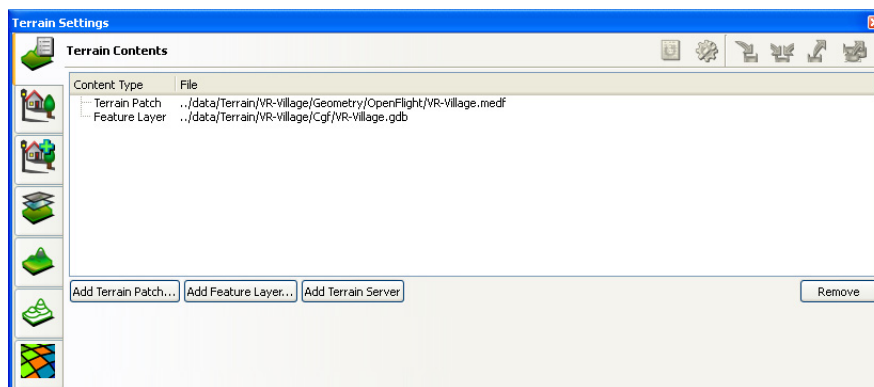


Figure 11-8. Terrain Contents

3. Click Add Terrain Server. The Add Terrain Server dialog box opens ([Figure 11-9](#)). 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 11-14.)

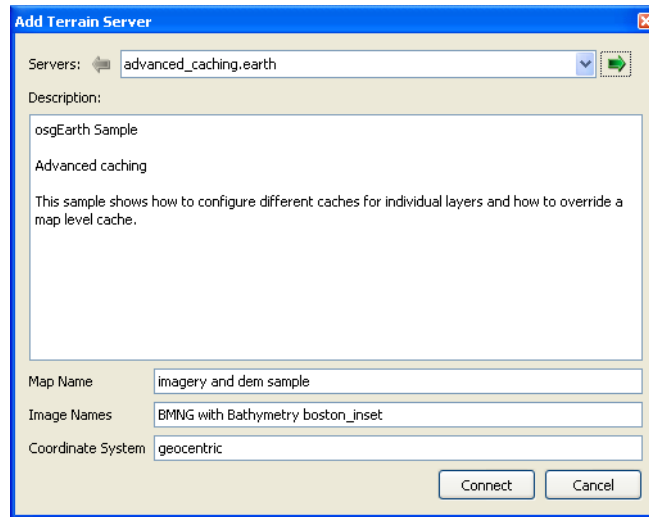


Figure 11-9. Add Terrain Server dialog box

4. Select a server from the Servers drop-down list.
5. 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 11-4.

11.5.1. Adding Terrain Server Connections

You can add additional terrain servers to the list provided with VR-Forces. Terrain servers 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.

[“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:

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

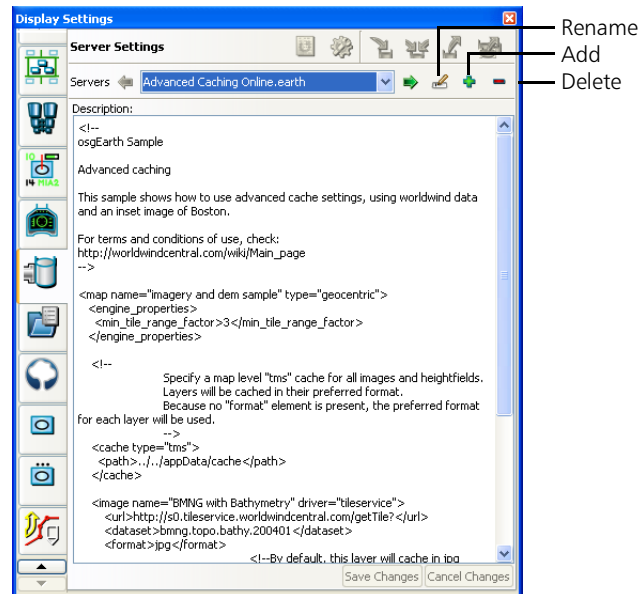


Figure 11-10. Server settings page

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*.

11.5.2. 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** → **Display**. The Display Settings dialog box opens.
2. Select the Server Settings page (Figure 11-10). 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.

11.6. Loading MetaFlight Terrains

VR-Forces supports MetaFlight terrain databases. To load a MetaFlight terrain, add a terrain patch, as you would for any other non-MTF terrain.



VR-Forces can only load MetaFlight terrains that conform to a particular specification. If you try to load a terrain that does not conform to the specification, VR-Forces sends an error message to the console and the log file. For details about this specification, please contact support@mak.com.

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 11-26.

11.6.1. Configuring MetaFlight File Loading

Loading MetaFlight terrains can affect performance. Therefore, VR-Forces lets you configure how you want to load these terrains. You can:

- ♦ Balance performance against visual quality
- ♦ Configure the space between tiles (geometry offset)
- ♦ Compress virtual textures.

Balancing Performance Against Visual Quality

VR-Forces can load a MetaFlight file so that it maximizes performance, maximizes visual quality, or achieves some balance of these goals. The factors that affect performance and visual quality are:

- ♦ Compositing. Compositing is the process of stitching N high-resolution image tiles, each of which covers a portion of the total terrain, into a single image tile that spans the combined coverage of those tiles. Compositing is used to try to achieve high-resolution imagery on coarse geometry.
- ♦ Tile-Matching. Tile matching is the process of finding or generating a single image tile whose extents match exactly the extents of a single geometry tile. If a tile with adequate resolution and coverage is not found, one can be generated by compositing multiple higher-resolution image tiles together.
- ♦ Highest 1x1. Highest 1x1 is the single highest-resolution tile that covers the entire coverage extent of a geometry dataset. Compositing can be used to generate a higher resolution image tile than that available in the virtual texture dataset.

You can choose the balance between visual quality and performance by adjusting a slider that has the following options:

- ♦ Tile-matching with max compositing
- ♦ Tile-matching with 4096 compositing
- ♦ Tile-matching with 2048 compositing
- ♦ Tile-matching with 1024 compositing
- ♦ Tile-matching with no compositing (Default)
- ♦ Highest 1x1 with max compositing
- ♦ Highest 1x1with 4096 compositing
- ♦ Highest 1x1with 2048 compositing
- ♦ Highest 1x1with 1024 compositing
- ♦ Highest 1x1with no compositing.

For the best performance, choose Highest 1x1 with no compositing. For the best visual quality, choose Tile-matching with max compositing.



High quality visual appearance is CPU intensive. Do not select this option unless you have very high-end hardware, such as a 64-bit operating system with multiple gigabytes of RAM.

To balance performance against visual quality:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Loader Settings page ([Figure 11-11](#)).

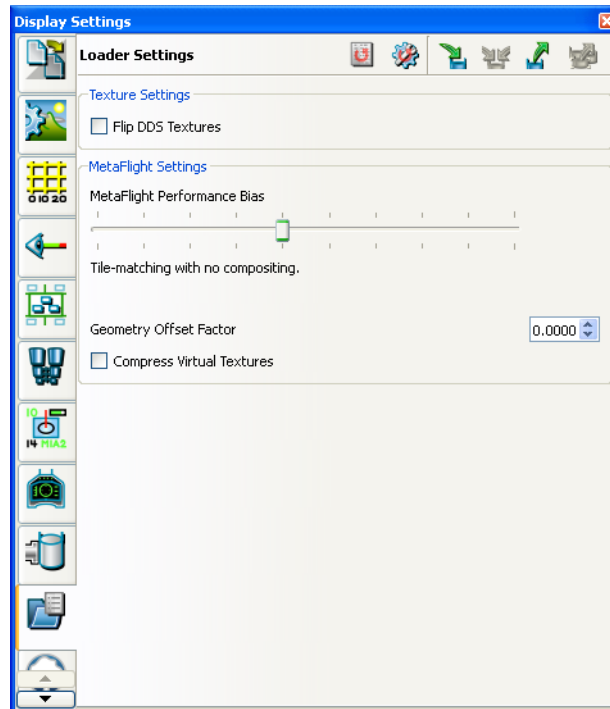


Figure 11-11. Loader Settings page

3. Adjust the MetaFlight Performance Bias slider. As you move the slider, the caption changes to tell you which option you have chosen.

Specifying the Geometry Offset Factor

The geometry offset factor scales the geometry tile offset by a specified percentage. It lets you tune the space between tiles. The factor is a multiplier on the tile width, which is subtracted from the tile offset. Typical values are in the 0.0005 to 0.001 range. The factor is applied in both X and Y dimensions.

To specify the geometry offset factor:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Loader Settings page (Figure 11-11).
3. Type a value in the Geometry Offset Factor box or use the spin control to set the value.

Compressing Virtual Textures

When you compress virtual textures, you apply DDS compression to virtual texture tiles.



Compressing virtual textures is CPU intensive. Do not select this option unless you have very high-end hardware, such as a 64-bit operating systems with multiple giga-bytes of RAM.

To compress virtual textures:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the Loader Settings page ([Figure 11-11](#)).
3. Select the Compress Virtual Textures check box.

Setting the MetaFlight Cache Using an Environment Variable

You can specify the location of the MetaFlight cache files with the MAK_OSG_MFT_CACHE_ROOT environment variable.

11.7. 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** → **Terrain Page-in Area**.
2. Create the area as described in Section 2.1, “[Creating Objects](#)”, in *VR-Forces Scenario Management Guide*.

11.8. Adding Props to a Terrain

Props are objects like buildings, lampposts, or trees that exist independently of the underlying terrain elevation data. You can extract props from a terrain patch or add them from a feature layer. You can view a list of the props in the terrain, specify their opacity, and enable or disable their visibility. You can change a prop's type and you can change the model definition used to render it. You can also create new prop types to meet your simulation needs.



In VR-Forces you can create buildings and cultural feature objects from the Create menu and Entity Palette that look like similarly names 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.

11.8.1. Extracting Props from a Terrain Patch

When you extract props from a terrain patch, you have to tell VR-Forces where to find prop data, what type of prop to use, and what model definitions to map the props to, as follows:

- ♦ Where to search: A terrain patch, such as an OpenFlight terrain, often references multiple files that define models that are incorporated into the geometry of the terrain. VR-Forces lets you specify the external files from which to extract props or you can specify a search expression that looks in all referenced files. If you select a referenced file, the process is quicker because VR-Forces searches for props only in the specified file.
- ♦ What prop type to use: You can select a prop type from a list, specify a type of your choice, or let VR-Forces auto-generate prop types. If you choose Auto Generate, VR-Forces makes the prop type match the name of the referenced file. For example, a prop extracted from the *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 11-12](#)).

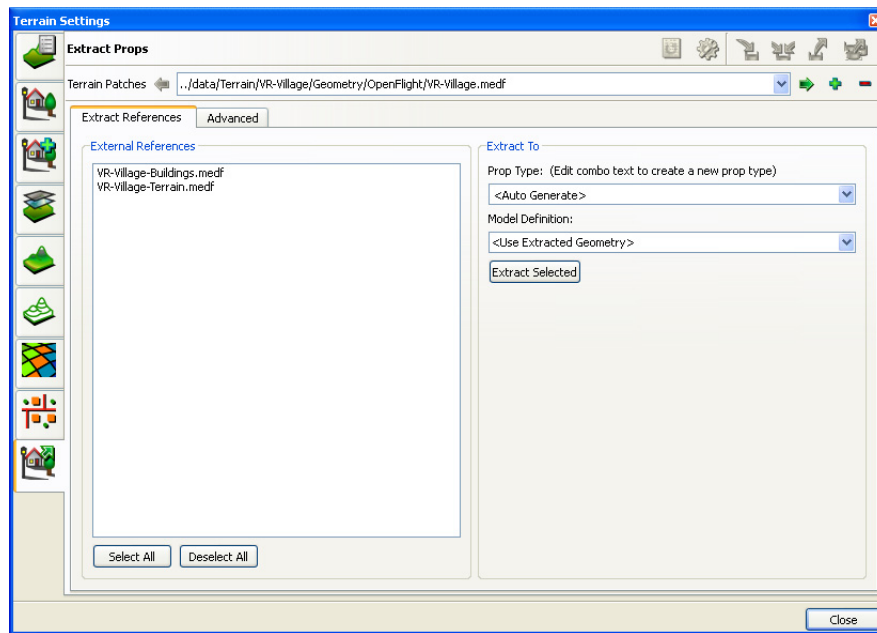


Figure 11-12. Extract Props page

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.

11.8.2. Adding Props from a Feature Layer

A feature layer source file contains information about point, line, or areal features. Feature information is stored in a table in which each row is a feature and each feature is defined by the values in an arbitrary number of columns. To add a prop from a feature layer, you must build a rule that specifies the column headings and values for the feature you want to import. Then you must associate a prop type and model definition with the prop. You also must specify whether or not to ground clamp the prop.

Please see “[Composing a Terrain](#),” on page A-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 11-13](#)).

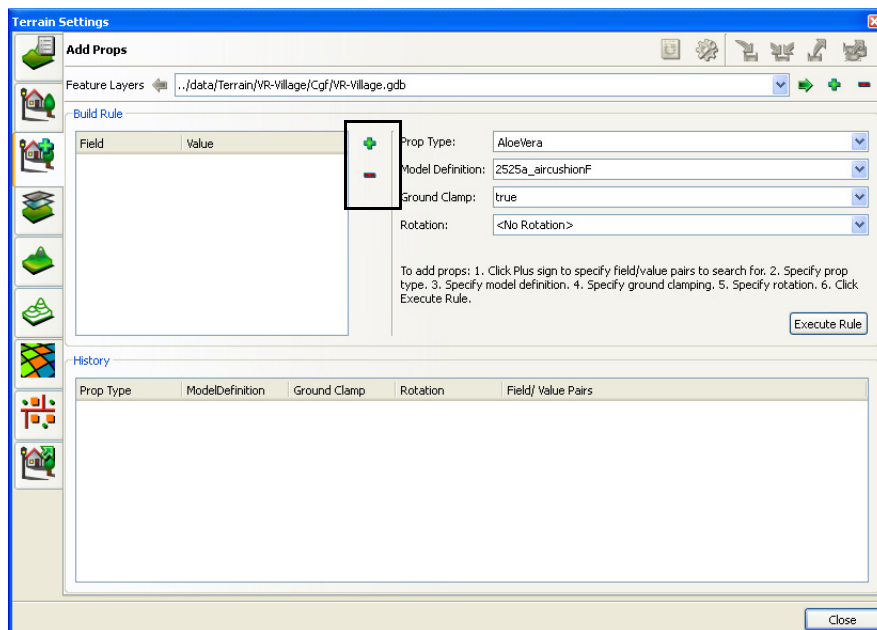


Figure 11-13. Terrain Settings, Add Props page

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 11-13). The Build Field/Value Pair dialog box opens (Figure 11-14).



Figure 11-14. 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 11-15).

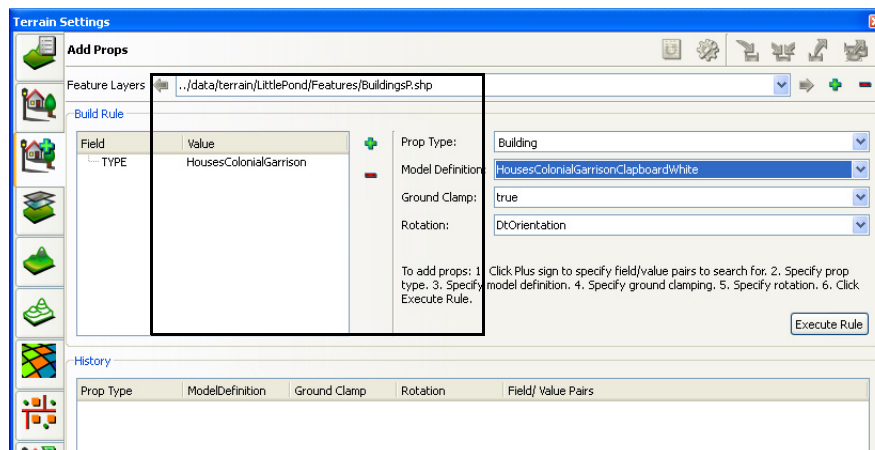


Figure 11-15. New Field/Value pair

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.

11.8.3. Viewing a List of Props

To view a list of the props in the terrain:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Props page (Figure 11-16).

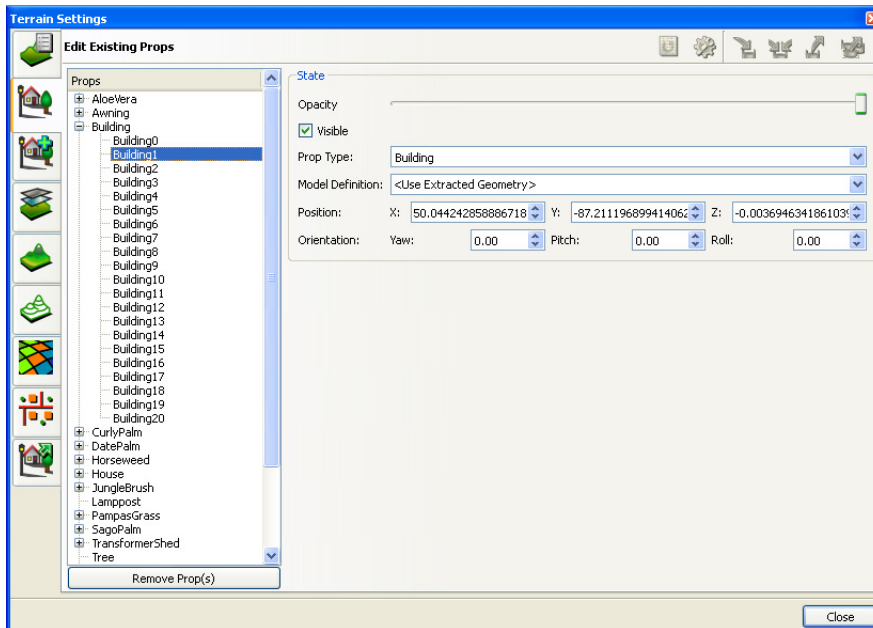


Figure 11-16. Props page

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

11.8.4. Selecting Props

- To select a prop, click it in the window or select it in the list of props on the Terrain Settings dialog box, Props page.

Press **Ctrl**+click to select multiple props.

11.8.5. Setting the Opacity of Props

To set a prop's opacity:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Props page ([Figure 11-16](#)).
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**.

11.8.6. Changing a Prop's Type

Every prop has a type. Props are organized by type in the list on the Props page.

To change a prop's type:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Props page ([Figure 11-16](#)).
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.

11.8.7. Changing a Prop's Position or Orientation

You can change a prop's position and orientation. You can change the orientation of multiple props at the same time without affecting their position. However, if you select multiple props and change position, they are all placed in the same position.

To change a prop's position or orientation:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Props page ([Figure 11-16](#)).
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

11.8.8. 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 13, [Model and Element Definitions](#). For an example of changing a prop's model definition, please see [“Test the Model Mapping,”](#) on page B-10.

To change a prop's model definition:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Props page ([Figure 11-16](#)).
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.

11.9. Configuring File Caching

When file caching is enabled, VR-Forces stores information about the terrain and models in a format that enables it to be loaded quickly. This improves loading time the next time the cached objects are loaded. If caching is disabled, terrains and models must be loaded from their native formats, which may take longer.

To enable or disable file caching:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the File Cache Settings page (Figure 11-17).

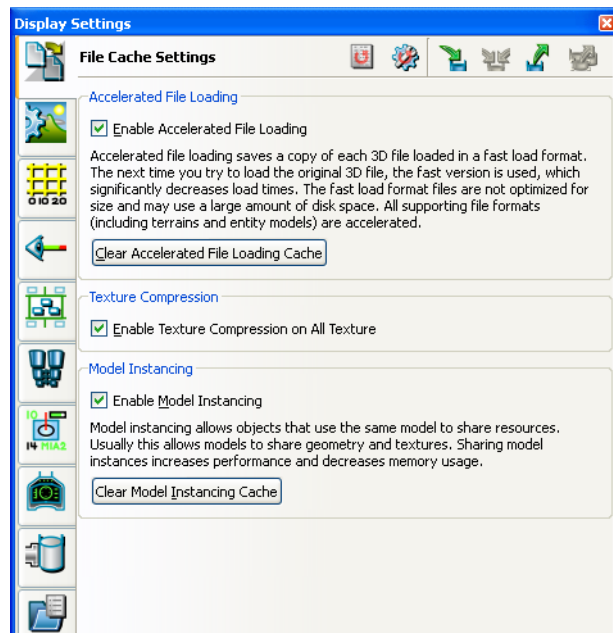


Figure 11-17. File Caching page

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

11.9.1. 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** → **Display**. The Display Settings dialog box opens.
2. Select the File Cache Settings page (Figure 11-17).
3. Select or clear the Enable Texture Compression on all Texture check box.

11.9.2. 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** → **Display**. The Display Settings dialog box opens.
2. Select the File Cache Settings page (Figure 11-17).
3. Click the Clear Accelerated File Loading Cache button.

11.10. Displaying DDS Textures Correctly

DDS is a DirectX bitmap format that is often used for textures. DirectX expects the origin of the image to be at the top left corner. OpenGL expects the image origin to be at the bottom left corner. By default, VR-Forces assumes that images have the OpenGL orientation. However, sometimes images are not oriented properly and therefore you may have to tell VR-Forces to flip them.

You can change the DDS textures setting as follows:

- ♦ On a per-model basis. (For details, please see “[Flipping DDS Textures for a Model](#),” on page 13-26.)
- ♦ When you add a terrain patch. (For details, please see “[Adding Elevation Data \(Terrain Patches\) to a Terrain](#),” on page 11-4.)
- ♦ As a global setting. (For details, please see “[Flipping DDS Textures Globally](#),” on page 11-27.)

11.10.1. 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 11-11).
3. Select the Flip DDS Textures check box.
4. Click Close.

11.11. 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 11-18 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.

i

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.

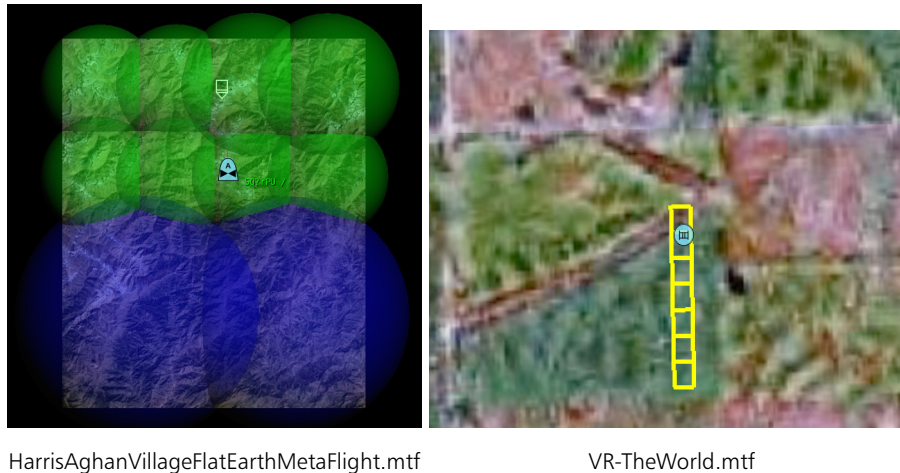


Figure 11-18. Paged-in terrain

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

11.11.1. Displaying Additional Terrain Paging Information

You can display the following terrain paging information by entering commands on the back-end console. [Table 11-1](#) lists the commands and their results.

Table 11-1: Terrain paging information commands

Command	Result
<code>terrain drawmode mode</code>	Specifies a bitmask that controls what information to display, as follows: ♦ 0 — Turn off information display ♦ 1 — Display polygons (Figure 11-19) ♦ 2 — Display collision object bounding boxes (paged terrains) (Figure 11-20) ♦ 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.
<code>terrain PSG</code>	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.
<code>pageout</code>	Pages out all tiles. Tiles that are needed are then paged back in.
<code>terrain save filename</code>	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 <code>simTerrain.mtf</code> .
<code>pbv</code>	Prints page bounding information to the console.
<code>terrain addlayer layer</code>	Displays information in the layer specified by <code>layer</code> . 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.

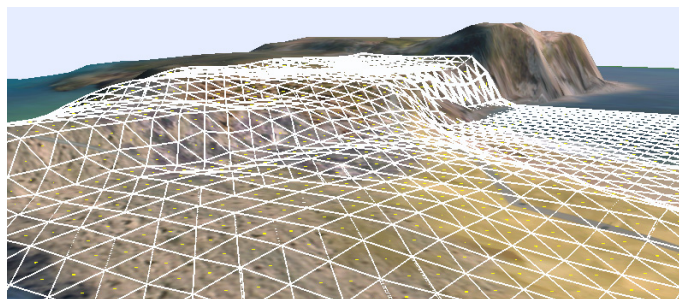


Figure 11-19. Polygon display

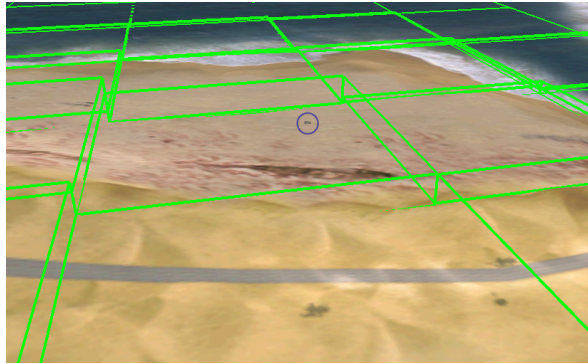


Figure 11-20. Collision objects display

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.

11.11.2. 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-Vantage 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 11-15.

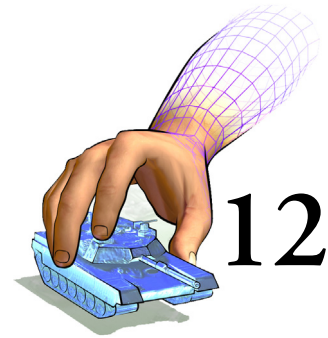
11.12. Building Efficient MetaFlight Terrains for VR-Forces

VR-Forces can load MetaFlight terrains built with Presagis Creator Terrain Studio (tested up to version 2.1). When constructing MetaFlight terrains for use with VR-Forces, note the following requirements:

- VR-Forces can load multiple MetaFlight files. However, they must share the same coordinate system, same projection and datum, have the same origin, and be all localized or all unlocalized.
- A MetaFlight file can contain one geometry grid dataset and a virtual texture dataset. However, if you load multiple MetaFlight files, only one of them can have a virtual texture dataset.
- The geometry grid dataset must be set up in a pyramid structure.

When you construct a MetaFlight terrain following these recommendations may improve performance:

- A geometry grid dataset should have relatively few top nodes and three to five levels of detail (LODs). OSG forces loading of all sibling nodes at the same time. Therefore, if you have many top nodes, they all get loaded, which defeats the purpose of terrain paging.
- Paging works on nodes flagged as FILE LOD GEOM. A FILE LOD GEOM node can contain LOD GEOM nodes internally. The terrain should be balanced so that you do not have too many FILE LOD GEOMS or too many LOD GEOM nodes in any one FILE LOD GEOM. Since the LOD GEOM nodes get loaded immediately, having too many of them may make a FILE LOD GEOM too large to be loaded without affecting performance, or conceivably too large to be loaded at all. At the same time, having too many FILE LOD GEOMS can also slow down terrain paging because of delays in accessing the file system.
- The geometry grid tile size (in meters) should match the virtual texture tile coverage (in meters.) For best performance there should be a one-to-one relationship between the highest LOD of the geometry tiles and the highest resolution of the virtual texture tile.



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	12-2
Adding a Model Mapping	12-3
Editing a Model Mapping	12-4
Filtering the Element Definition List	12-5
Deleting a Model Mapping	12-6
How VR-Forces Maps DI-Guy Models	12-6
Configuring Model Instancing	12-7
Clearing the Model Instancing Cache	12-7
Compressing Model Files	12-8

12.1. 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.

i

- ♦ 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.
-

12.1.1. Adding a Model Mapping

This procedure describes how to map an element definition to an entity type enumeration. The element definition must already exist.

To add a model mapping:

1. Choose **Settings** → **Entity Type Mappings**. The Entity Type Mappings dialog box opens.
2. Select the Entity Mappings page (Figure 12-1). 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.

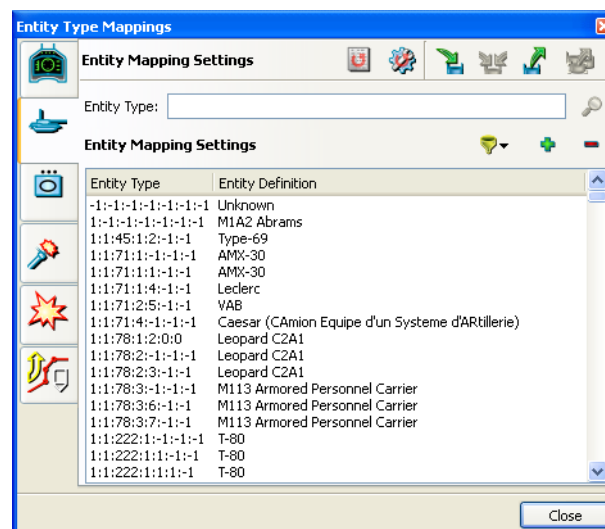


Figure 12-1. Entity Mappings page

3. Click the Add icon (+). The Create New Entity Type Model Mapping dialog box opens (Figure 12-2). It lists a default entity type enumeration.

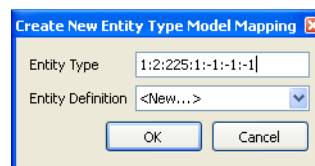


Figure 12-2. Create New Entity Type Model Mapping dialog box

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.

12.1.2. 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 Mappings page ([Figure 12-1](#)).
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 12-5.)
 - b. In the Entity Definition column, click the element definition to make it an editable list ([Figure 12-3](#))).

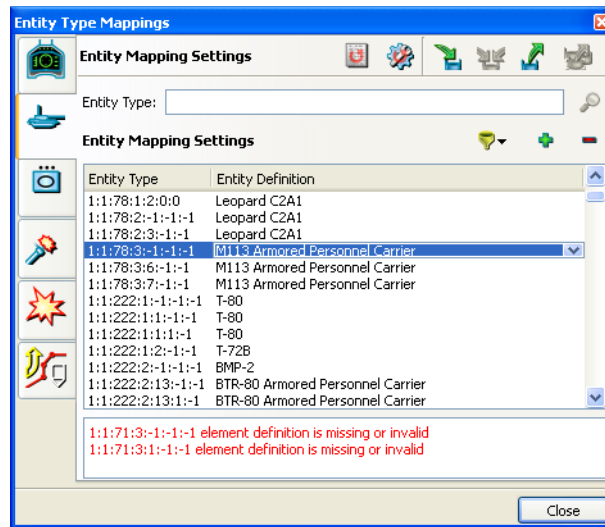


Figure 12-3. Editable entity definition

- c. Select the new element definition from the drop-down list.
- d. Click someplace else in the window.

12.1.3. 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 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.

12.1.4. 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 Mappings page ([Figure 12-1](#)).
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 (.

12.1.5. 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.

12.2. Configuring Model Instancing

When model instancing is enabled, models that use the same resources can share them. Usually, models can share geometry and textures. Use of model instance caching decreases memory use and improves performance.

To enable or disable model instancing:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the File Caching page (Figure 12-4).

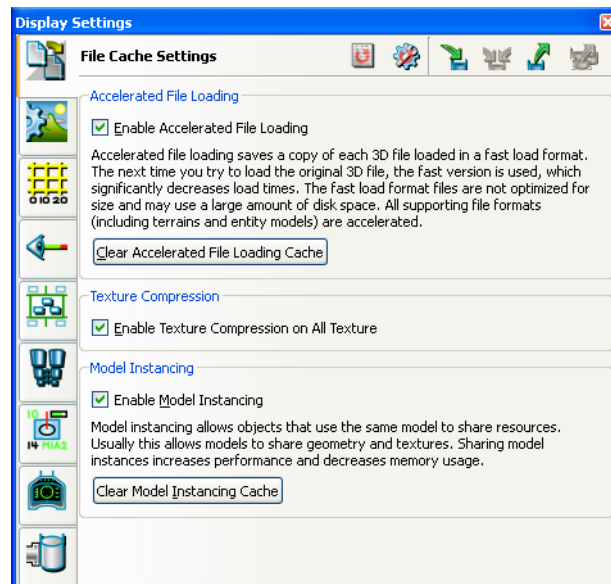


Figure 12-4. File Caching page

3. In the Model Instancing group box, select or clear the Enable Model Instancing check box.

12.2.1. Clearing the Model Instancing Cache

You can clear the model instancing cache if you need the disk space.

To clear the model instancing cache:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the File Caching page (Figure 12-4).
3. In the Model Instancing group box, Click the Clear Model Instancing Cache button.

12.3. 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/medf-tool.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 [-z optimization] [-i] [-o]
          [--image_extensions extensions] ...
          [--extensions extensions] ... [--norecurse]
          [--directory directory] [-f filename] [--] [-v] [-h]
```

Table 12-1 describes the arguments.

Table 12-1: medftool arguments

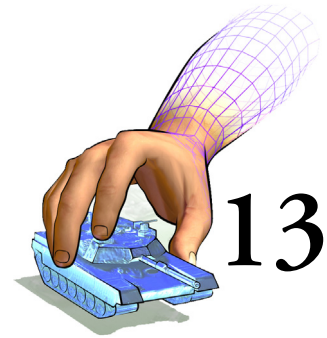
Argument	Description
<code>(-- ignore_rest)</code>	Ignore all arguments after this one.
<code>--directory directory</code>	The directory to recursively encrypt or decrypt.
<code>--extensions extension</code>	A file extension to encrypt. Use this argument to encrypt a directory. Can be used multiple times.
<code>(-f --file) filename</code>	The name of a file to encrypt.
<code>(-h --help)</code>	Displays usage information.
<code>(-i --ignore_errors)</code>	Ignore errors in the encryption process and continue without exiting.
<code>--image_extensions extension</code>	An image extension to encrypt. Use this argument to encrypt a directory. Can be used multiple times.
<code>--norecurse</code>	Do not recurse through subdirectories. Default: recurse through subdirectories.
<code>(-o --onlyIfNew)</code>	If a destination file for a source file already exists, encrypt or decrypt only if the source file is newer than the destination file.
<code>(-v --version)</code>	Display version information and exit.
<code>(-z --optimize) optimization</code>	Specifies the optimization level. Slow optimization may provide better optimization than other settings, but it is significantly slower. Possible values are: ♦ 0 = None ♦ 1 = Fast ♦ 2 = Slow. Default: Fast.

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
```

Model and Element Definitions

In “[Introduction to VR-Forces Object Modeling](#),” on page 4-4, 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	13-3
Creating Schemas	13-5
Deleting Schemas	13-6
Copying Schemas	13-6
Adding a Parameter to a Schema	13-7
Deleting a Parameter from a Schema	13-8
Editing a Schema Parameter	13-8
Creating and Editing Model Definitions	13-9
Creating a Model Definition	13-10
Deleting a Model Definition	13-11
Copying a Model Definition	13-11
Filtering the List of Model Definitions	13-12
Editing a Model Definition	13-12
Saving Model Definitions	13-15
Creating and Editing Element Definitions	13-15
Creating an Element Definition	13-16
Copying an Element Definition	13-17
Editing an Element Definition	13-18
Deleting an Element Definition	13-19
Reorganizing Element Definitions	13-20
Saving Element Definitions	13-20
Editing 2D Icons	13-21

Adding New Cockpit Display Models.....	13-23
Installing a Cockpit DLL	13-24
Creating a Model Definition for a Cockpit.....	13-24
Flipping DDS Textures for a Model.....	13-26
Configuring SpeedTree Trees	13-27
Specifying the SpeedTree Performance Profile.....	13-28
Best Practices for Creating Models for VR-Forces	13-29

13.1. Creating and Editing Schemas

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



Changes to schemas are saved automatically. For more information, please see [“Managing VR-Vantage Settings,”](#) on page 3-14.

VR-Forces includes model definition schemas and visual definition schemas. Model definition schemas define the elements that can be displayed in a scene. Visual definition schemas define the visual effects that can be applied to scene elements. 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.

The visual definition schemas provided with VR-Forces are:

-
♦ 2D Entity Heading Indicator Visualizer	♦ Intervisibility Entity Fan Visualizer
♦ 2D Entity Sensor Volume Visualizer	♦ Intervisibility Fan Visualizer
♦ 2D Environmental Area Visualizer	♦ Intervisibility Line Visualizer
♦ 2D Environmental Line Visualizer	♦ Intervisibility Vertex Visualizer
♦ 3D Environmental Area Visualizer	♦ Kinematic State Visualizer
♦ 3D Environmental Line Visualizer	♦ MilSTD Icon Visualizer
♦ Aggregate Extent Visualizer	♦ Model Visualizer
♦ Decal Effect Visualizer	♦ Remote Graphics Visualizer
♦ Detonation Line Visualizer	♦ Selection Visualizer
♦ Entity Sensor Volume Visualizer	♦ Shape Vertex Visualizer
♦ Flames Visualizer	♦ Smoke Plume Visualizer
♦ HAT Line Visualizer	♦ Track History Visualizer
♦ Indicator Visualizer	♦ Trail Effect Visualizer
♦ Interaction Visualizer	
-

You cannot add or delete schemas for visual definitions. The list of visual definitions is based on the visual definition classes in the application. You can only edit the visual definition schemas.

13.1.1. 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 13-1).

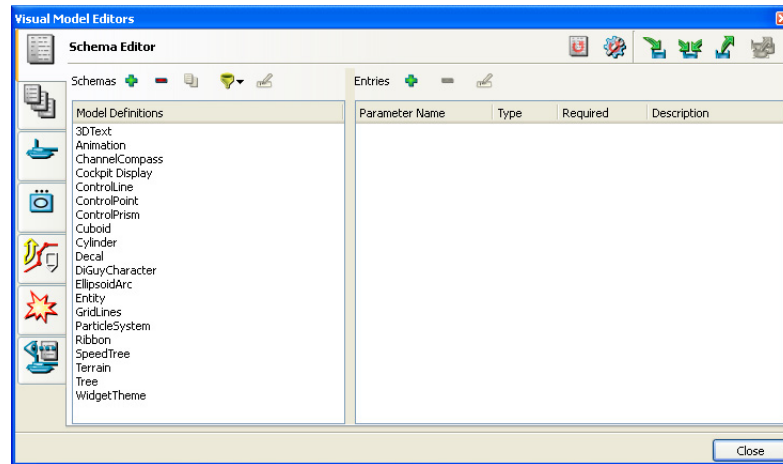


Figure 13-1. Schema Editor page

3. Click the Add icon (+). The Create Instance dialog box opens (Figure 13-2).

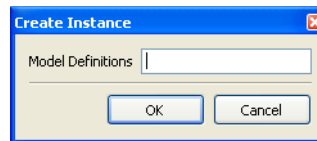


Figure 13-2. Create Instance dialog box

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 13-7.

13.1.2. Deleting Schemas



If a schema is used by a model definition, you cannot delete it. The Delete Schema icon will be unavailable.

You cannot delete Visual Definition schemas.

To delete a schema:

1. Choose **Settings** → **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page ([Figure 13-1](#)).
3. In the Schemas list, select the schema you want to delete.
4. Click the Delete icon (■). You are prompted to confirm the deletion.
5. Click Yes. The schema is removed from the list.

13.1.3. 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 13-1](#)).
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 13-8.

13.1.4. 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 13-1).
3. Optionally, filter the list by clicking the Filter icon (🔍) and selecting an option from the list.
4. In the Schemas list, select the schema to which you want to add a parameter.
5. Click the Add Parameter icon (+). A parameter called “key” is added to the parameter list.
6. Double-click “key” to enter edit mode.
7. Type a name for the parameter.
8. If the parameter has a specific type, edit the value in the Type column.
9. If the parameter is required, edit the value in the Required column.
10. Optionally, add a description. The description provides context sensitive help on the element definitions pages.

13.1.5. 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 (Figure 13-1).
3. Optionally, filter the list by clicking the Filter icon (🔍) and selecting an option from the list.
4. In the Schemas list, select the schema from which you want to delete a parameter.
5. Select the parameter that you want to delete.
6. Click the Delete icon (➖).

13.1.6. 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 13-1).
3. Optionally, filter the list by clicking the Filter icon (🔍) and selecting an option from the list.
4. In the Schemas list, select the schema for which you want to edit a parameter.
5. Click the parameter attribute that you want to edit. The attribute becomes editable.
6. Type the new attribute value.

13.2. Creating and Editing Model Definitions

Each model displayed in VR-Vantage 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 13-3.) 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 13-3](#)). The icons all have tooltips. For more information, please see [“Managing VR-Vantage Settings,”](#) on page 2-6 and [“Common VR-Vantage Icons,”](#) on page 2-10.

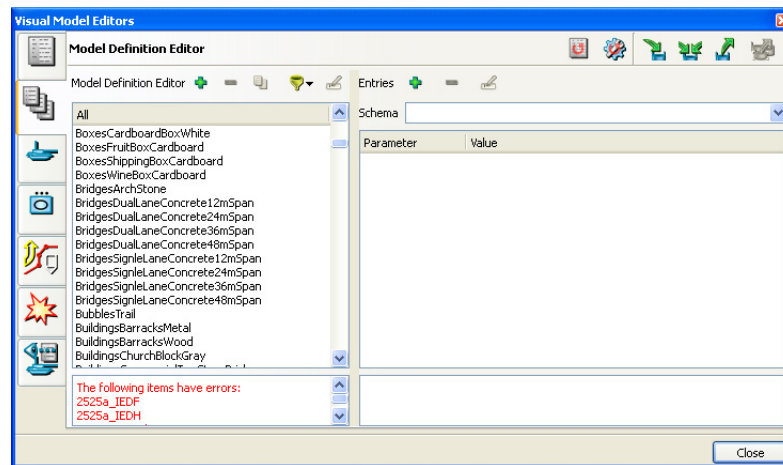


Figure 13-3. Model Definition Editor page

13.2.1. 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 13-3](#)).
3. Optionally, filter the model definitions in the list. (For details, please see “[Filtering the List of Model Definitions](#),” on page 13-12.)
4. Click the Add icon (+). The Add Model Definition dialog box ([Figure 13-4](#)).

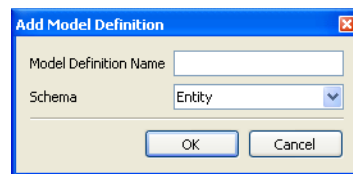


Figure 13-4. Add Model Definition dialog box

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 13-12.

13.2.2. 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 13-3).
3. Optionally, filter the model definitions in the list. (For details, please see “[Filtering the List of Model Definitions](#),” on page 13-12.)
4. Select the model that you want to delete.
5. Click the Delete icon (✖). You are prompted to confirm the deletion.
6. Click Yes. The model definition is removed from the list.

13.2.3. 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 13-3).
3. Optionally, filter the model definitions in the list. (For details, please see “[Filtering the List of Model Definitions](#),” on page 13-12.)
4. Select the model that you want to copy.
5. Click the Duplicate icon (📄). The Duplicate Model Definition dialog box opens (Figure 13-5).

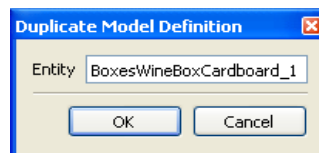


Figure 13-5. Copy model definition

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 13-12.

13.2.4. 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.

13.2.5. 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 13-3](#)).
3. Optionally, filter the model definitions in the list. (For details, please see [“Filtering the List of Model Definitions,”](#) on page 13-12.)
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 13-6](#)).

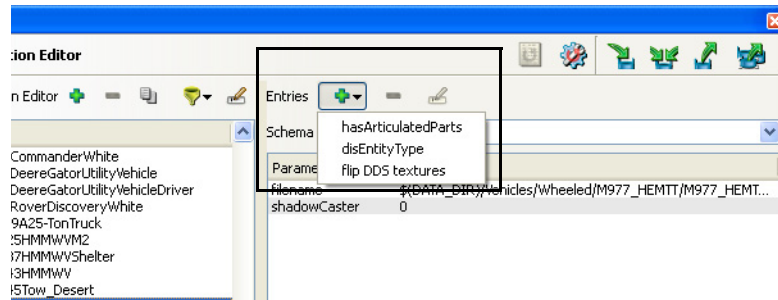


Figure 13-6. Add parameter list

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 13-7).

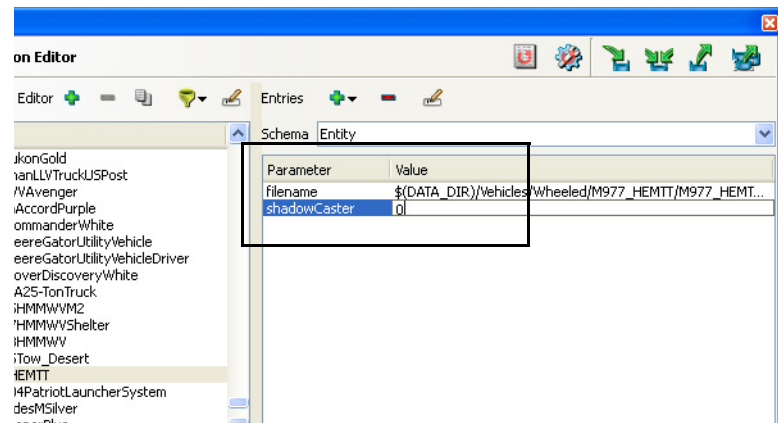


Figure 13-7. Editable parameter

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 13-3).
3. Optionally, filter the model definitions in the list. (For details, please see “[Filtering the List of Model Definitions](#),” on page 13-12.)
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 13-7).
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 13-3).
3. Optionally, filter the model definitions in the list. (For details, please see “[Filtering the List of Model Definitions](#),” on page 13-12.)
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 (■). The parameter is removed.

13.2.6. 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 [“Managing VR-Vantage Settings,”](#) on page 2-6.

13.3. Creating and Editing Element Definitions

Element definitions let you assign visualizers to scene elements (entities, tactical graphics, interactions, aggregates, tactical graphics, cultural features, and props), 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 for entities, tactical graphics, aggregates, spot reports, and interactions. The procedures are the same for each element type.

13.3.1. 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 13-8](#)).

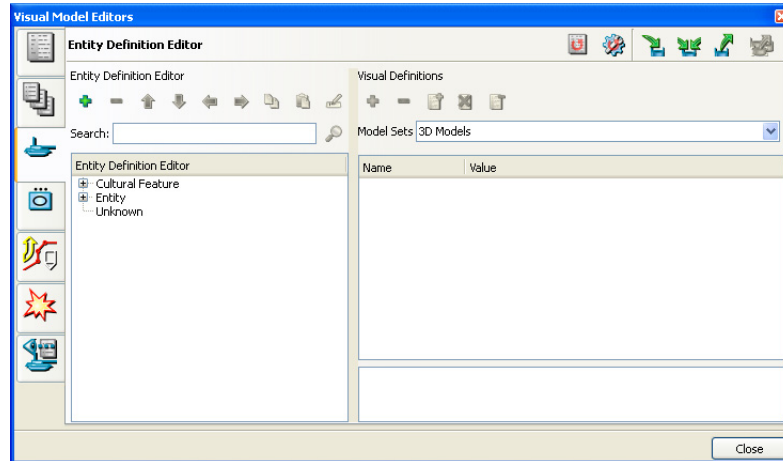


Figure 13-8. Entity Definition Editor page

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 (+). An element called “untitled” is added to the list of elements. It has all of the visualizers that its parent element has.
5. Select the new element.
6. Click the Rename icon (✎). The Rename Element Definition dialog box opens.
7. Type a name for the element.
8. Click OK.
9. Optionally, edit the visualizers assigned to the element. For details, please see [“Editing a Visualizer,”](#) on page 13-19.
10. Optionally, add a visualizer to the element. For details, please see [“Adding a Visualizer,”](#) on page 13-18.
11. Optionally, save the element. For details, please see [“Saving Element Definitions,”](#) on page 13-20.

13.3.2. Copying an Element Definition

If you want to create a new element definition that will be similar to an existing definition, it may be easier to copy an element definition and edit it than to create a new one.

When you copy an element definition, you can only paste in the same level of the element hierarchy from which you copied it. In other words, you cannot copy an element definition of an element in the Land/Tank group and paste the copy into the Land/Armored Fighting Vehicle group.

To copy 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 copy ([Figure 13-8](#)).
3. Select the element definition that you want to copy.
4. Click the Copy icon (.
5. Click the Paste icon (). A new element definition is added to the list next to the one you copied. It has a default name. To rename the element definition:
 - a. Click the Rename icon (). The Rename Element Definition dialog box opens.
 - b. Type the new name.
 - c. Click OK.
6. Edit the parameters as described in [“Editing an Element Definition,”](#) on page 13-18.

13.3.3. 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 13-8](#)).
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 (+). The Add Visualizer 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 13-19.

Editing a Visualizer

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

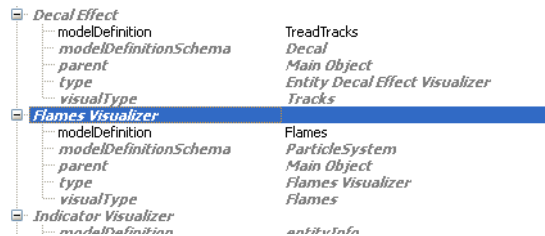


Figure 13-9. Visualizer entry

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 13-8).
3. Select the element definition that you want to edit.
4. In the visualizer pane, double-click the value of the parameter that you want to edit. Depending on the parameter, a drop-down list of available values is displayed, a dialog box with a list of possible values is displayed, or the field becomes editable.
5. Select or type the new value.

13.3.4. 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 13-8).
3. Select the element definition that you want to delete.
4. Click the Delete icon (■).

13.3.5. Reorganizing Element Definitions

Element definitions are organized hierarchically, when it is appropriate or useful to do so. You can move them around in the hierarchy. You cannot create new hierarchical units.

To reorganize 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 move ([Figure 13-8](#)).
3. Expand the element definition hierarchy.
4. Select the element definition that you want to move.
5. To move an element definition up or down within its list of siblings, click the Up (↑) or Down (↓) icon.

To move an element definition out of its sibling group and up in the hierarchy, move it to the top of its sibling group and click the Left Arrow icon.

To move an element definition into a hierarchical group (down the hierarchy), use the arrow icons to move it below the hierarchical level into which you want to move it and click the Right Arrow icon.

13.3.6. Saving Element Definitions

Changes to element definitions get saved automatically. You can also save element definitions using the standard VR-Forces options for saving settings. For details, please see [“Managing VR-Vantage Settings,”](#) on page 2-6.

13.4. Editing 2D Icons

2D icons are configured in the Military Symbol Icon Visualizer, which is part of an entity's entity definition. 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 13-8).
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 Symbol Icon Visualizer, select the fillGlyph attribute (Figure 13-10).

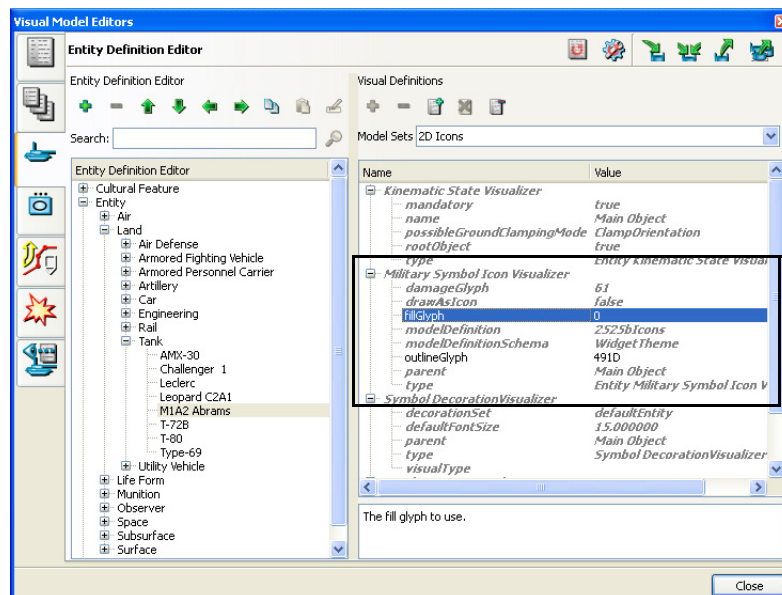


Figure 13-10. Military Symbol Icon Visualizer

- b. Click the value in the Value column. The Choose Font Glyph Index dialog box opens (Figure 13-11).

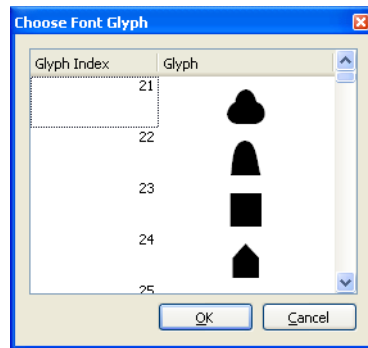


Figure 13-11. Choose Font Glyph dialog box

- 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.

13.5. 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 13-12](#) shows the model definition for a cockpit.

Cockpit updaters are associated with an attribute name that is known to the RSO. The value you provide for a cockpit-specific attribute is the name of an updater for dynamically changing values or a constant value. Constant values are commonly used to configure a cockpit when it is loaded. VR-Forces provides a set of commonly used updaters. You can add additional updaters or override existing updaters through a plug-in.



The GL Studio license included with VR-Forces only covers the cockpits delivered with the product. If you add your own cockpits, you will need GL Studio run-time licenses.

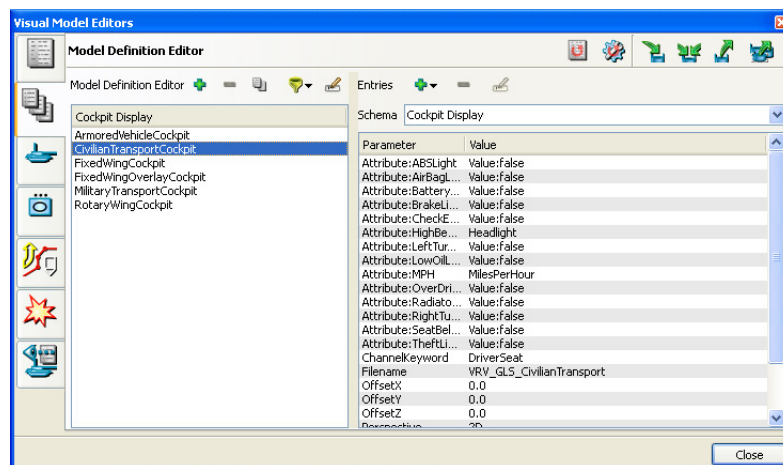


Figure 13-12. Cockpit model definition

The general procedure for adding a new cockpit model is:

1. Install the cockpit's DLL.
2. Create a model definition for the cockpit.
3. Map entity types to the model definition, as described in [“Adding a Model Mapping,”](#) on page 12-3.

13.5.1. 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.

13.5.2. 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 13-10. Be sure to select Cockpit Display as the schema. A new model definition is added to the display ([Figure 13-13](#)). The message window lists the parameters that you must add. These are the parameters defined by the schema.

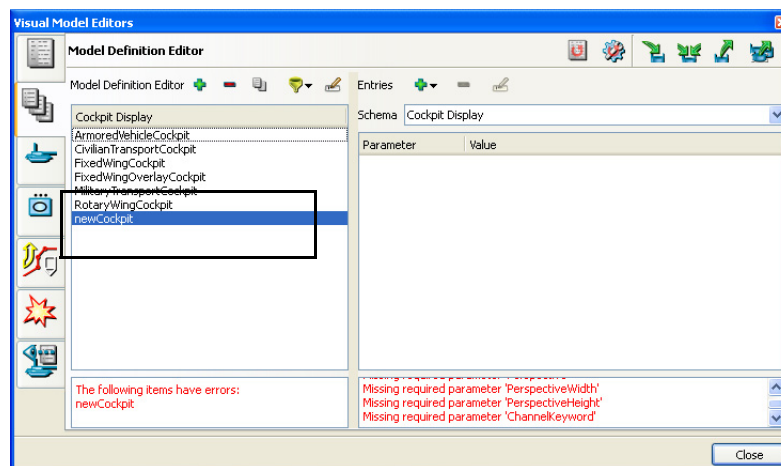


Figure 13-13. New cockpit model definition

2. Add the required parameters, as described in “[Adding Parameters to a Model Definition](#),” on page 13-12. [Table 13-1](#) describes the required parameters.

Table 13-1: 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: <code>filenameClass</code> .
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.

Table 13-1: Required parameters for cockpit models

Parameter	Description
PerspectiveWidth	Adjusts the orthogonal projection for a 2D cockpit. Adjusts the perspective projection for a true 3D cockpit.
PerspectiveHeight	
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	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.
Offset Y	
Offset Z	

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 13-12.
 - b. Double-click the parameter to make its name editable.
 - c. Add a colon and the name of an RSO attribute that you need to map. For example, Attribute:Heading.
 - d. In the Value column, click the value to make it editable. Type the name of an updater or specify a constant value in the format Value:*value*. You can specify an updater that you have added through a plug-in or one of the updaters provided with VR-Forces, as described in [Table 13-2](#).
4. Map an entity type to your definition, as described in [“Adding a Model Mapping,”](#) on page 12-3.

Table 13-2: Updaters provided

Updater	Description
AOA	Angle of Attack. Returns the pitch of the entity clamped (5 to 30 degrees).
FeetAGL	Feet Above Ground Level. Returns the feet above the terrain (or buildings below).
FeetMSL	Feet Above Mean Sea Level. Returns the altitude above the sea level.
FeetPerMinute	Feet Per Minute. Returns the climb rate.
KilometersPerHour	Speed of the entity in kilometers per hour.
MilesPerHour	Speed of the entity in miles per hour.
KnotsPerHour	Speed of the entity in nautical miles per hour.
Pitch	Returns the pitch of the entity clamped (-180 to 180 degrees).

Table 13-2: Updaters provided

Updater	Description
Roll	Returns the roll of the entity clamped (-180 to 180 degrees).
Yaw	Returns the yaw of the entity clamped (0 to 360 degrees).
Headlight	Returns the headlight state of the entity as true or false.
Value	This updater sends any value to the cockpit once and then never again. It is used to configure initial setup and to turn display components on or off.

13.6. 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 11-27.)

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 13-3](#)).
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.

13.7. Configuring SpeedTree Trees

SpeedTree trees are high resolution models of trees. To improve performance, you can set clipping plane and levels-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 10-10.

To configure SpeedTree trees:

1. Choose **Settings** → **Display**. The Display Settings dialog box opens.
2. Select the SpeedTree Settings page ([Figure 13-14](#)).

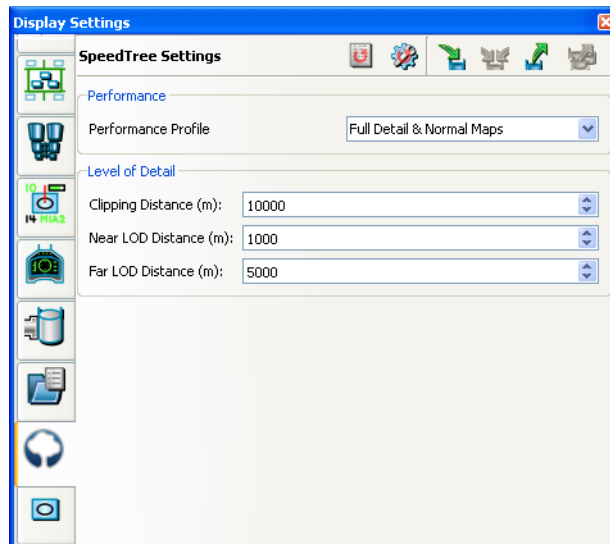


Figure 13-14. SpeedTree Settings page

3. In the Performance group box, select a performance profile. (For details, please see [“Specifying the SpeedTree Performance Profile,”](#) on page 13-28.)
4. In the Level of Detail group box, specify the clipping distance, and the near and far LOD distance.
5. Click Close.

13.7.1. 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 13-3](#) describes the profiles.

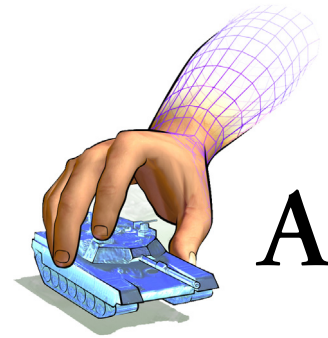
Table 13-3: SpeedTree Performance Profiles

Profile	Branch Texture	Fronde 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

13.8. 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, 128x128, 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.



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 5-29. 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.

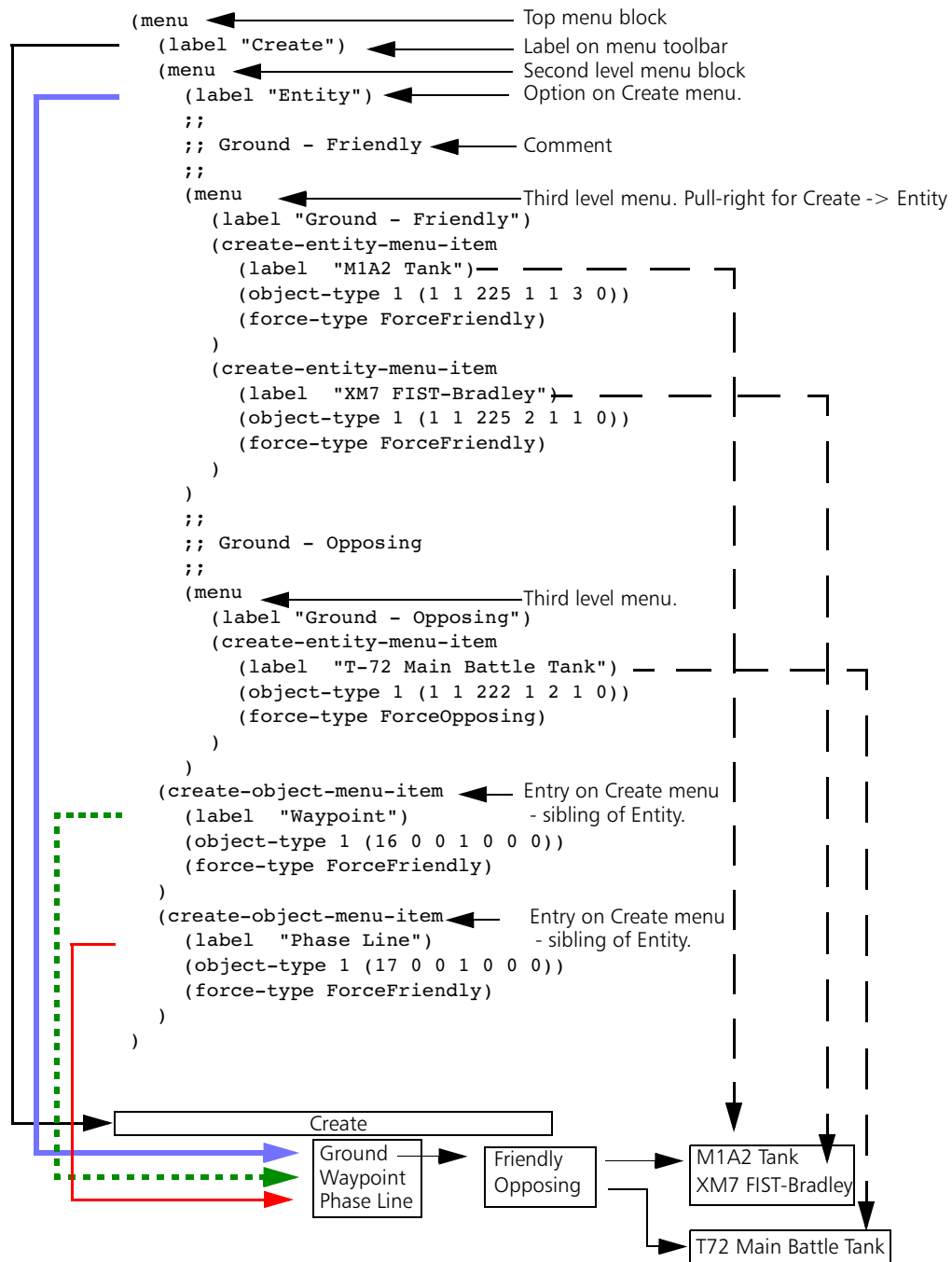


Figure A-1. Abbreviated *vrf.ent* file and menus created

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*

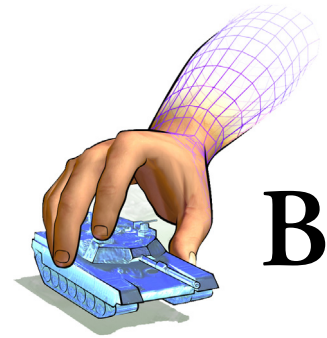


Before making changes to a configuration file, be sure to make a backup copy of the original file.

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.



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 7-3, in “Parameter Types,” on page 7-9, 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 7-6.
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
<code>state-repository</code>	(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: ♦ <code>"vrf-object-base-state-repository"</code>: Used for the base object in the object parameter database. ♦ <code>"vrf-object-state-repository"</code>: Derived from <code>vrf-object-base-state-repository</code>, 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. ♦ <code>"vrf-overlay-object-state-repository"</code>: For tactical graphics. ♦ <code>"vrf-moving-object-state-repository"</code>: 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. ♦ <code>"ground-entity-state-repository"</code>: For ground vehicles, such as tanks, trucks, dismounted infantry. ♦ <code>"rotary-wing-entity-state-repository"</code>: For rotary-wing aircraft. ♦ <code>"fixed-wing-entity-state-repository"</code>: For fixed-wing aircraft. ♦ <code>"missile-entity-state-repository"</code>: For missile and rocket entities. ♦ <code>"surface-entity-state-repository"</code>: For surface entities. ♦ <code>"human-state-repository"</code>: For lifeform entities. ♦ <code>"cultural-feature-state-repository"</code>: 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 $\text{rand}(\text{net-interface-min-tick-period}/2)$
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
<code>state-repository</code>	(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 <code>"vrf-moving-object-state-repository"</code> 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: ♦ <code>"vrf-object-state-repository"</code>: Derived from <code>vrf-object-base-state-repository</code>, 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. ♦ <code>"vrf-overlay-object-state-repository"</code>: For tactical graphics. ♦ <code>"vrf-moving-object-state-repository"</code>: For any object that you expect to simulate movement, including individual and aggregate entities. ♦ <code>"ground-entity-state-repository"</code>: For ground vehicles, such as tanks, trucks, dismounted infantry. ♦ <code>"rotary-wing-entity-state-repository"</code>: For rotary-wing aircraft. ♦ <code>"fixed-wing-entity-state-repository"</code>: For fixed-wing aircraft. ♦ <code>"missile-entity-state-repository"</code>: For missile and rocket entities. ♦ <code>"human-state-repository"</code>: For lifeform entities. ♦ <code>"cultural-feature-state-repository"</code>: For cultural features that are created as entities. ♦ <code>"surface-entity-state-repository"</code>: 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 *objTypeEnums.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*.

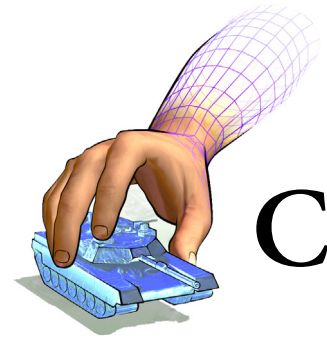
Other	0
ControlArea	1
Obstacle	2

ArealObjectCategory

B.4.9. Interaction

Kinds for Supertype interaction.

Other	0
Fire	1
Detonate	2



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 a Terrain Patch	C-3
Add Imagery	C-5
Finding the High Resolution Image	C-7
Add a Feature Layer	C-8
Add Props	C-8
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-24
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 a Terrain Patch.](#)
2. [Add Imagery.](#)
3. [Finding the High Resolution Image.](#)
4. [Add a Feature Layer.](#)
5. [Add Props.](#)

C.1.1. Add a Terrain Patch

First add a terrain patch. The terrain patch 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](#)).

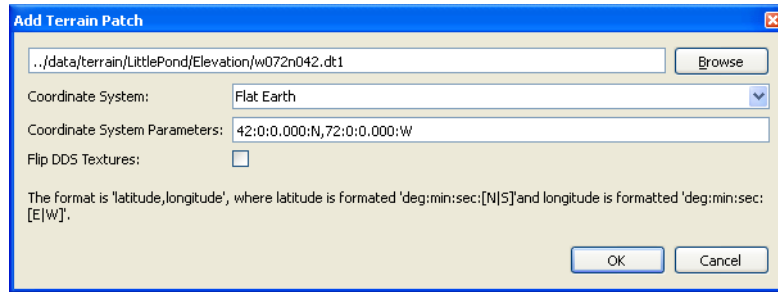


Figure C-2. Add Terrain Patch

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.

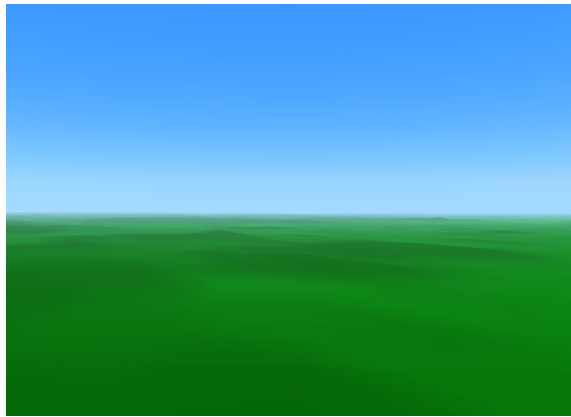


Figure C-4. Elevation coloring

To add image data:

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

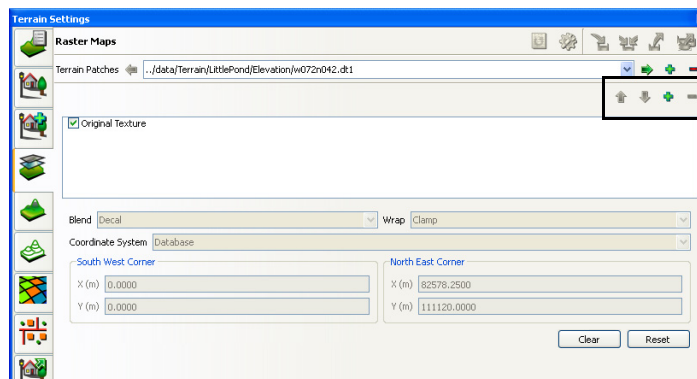


Figure C-5. Raster Maps page

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).

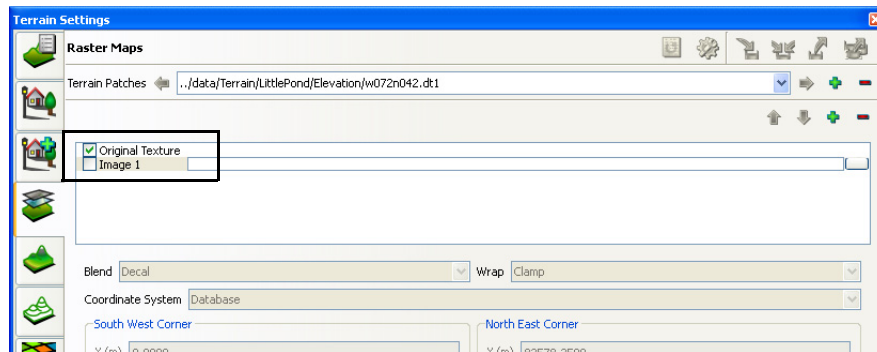


Figure C-6. New image

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).

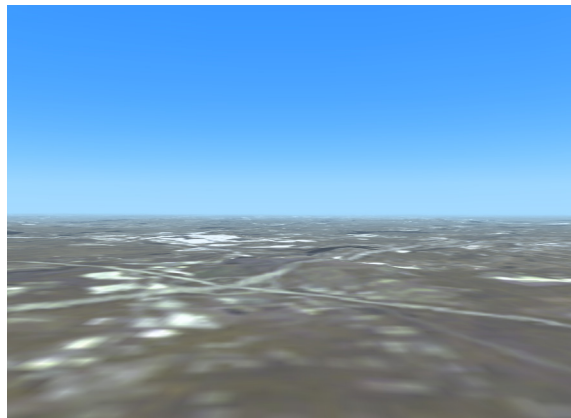


Figure C-7. Low resolution image

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).

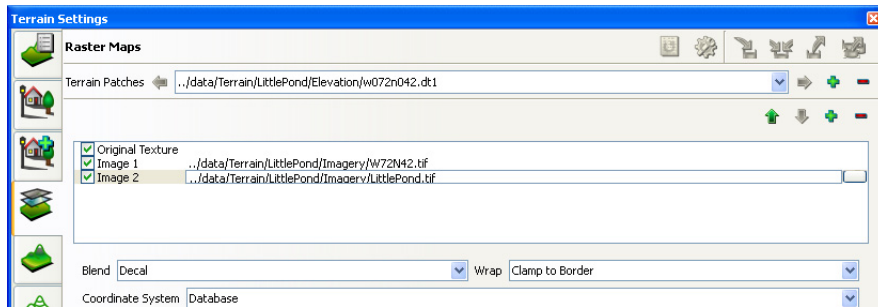


Figure C-8. Raster Maps page with both images added

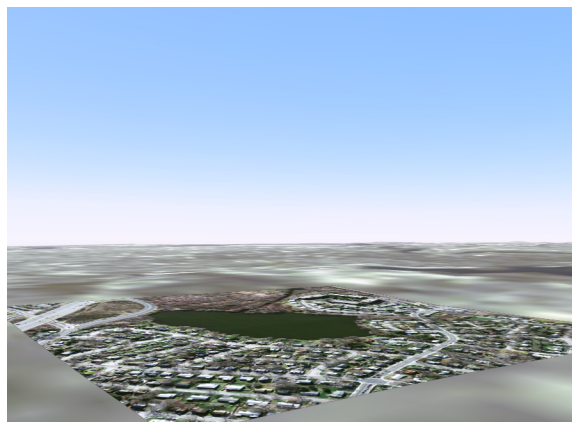


Figure C-9. Little Pond image

C.1.3. 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.1.4. 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 and build rules to add props.

To add the feature layer:

1. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Add Props page.
3. On the Feature Layers list at the top of the dialog box, click the Add icon (+). 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 (Figure C-10).

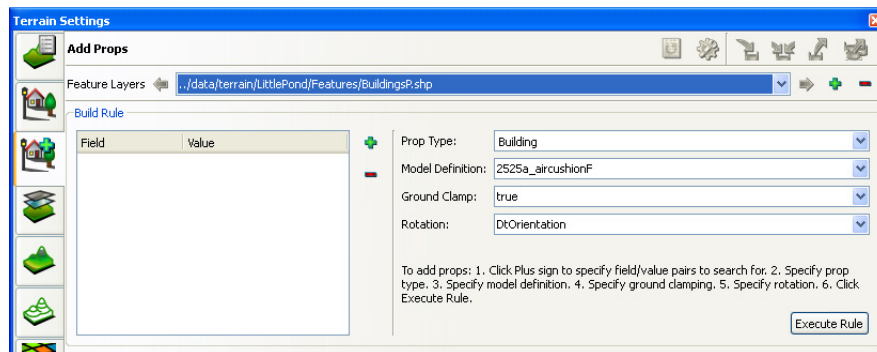


Figure C-10. Add Props page

C.1.5. 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 (+). The Build Field/Value Pair dialog box opens.
2. In the Field drop-down list, select TYPE.

3. In the Value drop-down list, select HousesColonialGarrison (Figure C-11).

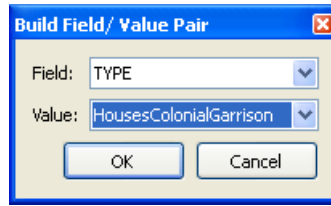


Figure C-11. Build Field/Value Pair dialog box

4. Click OK.
5. 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-12.

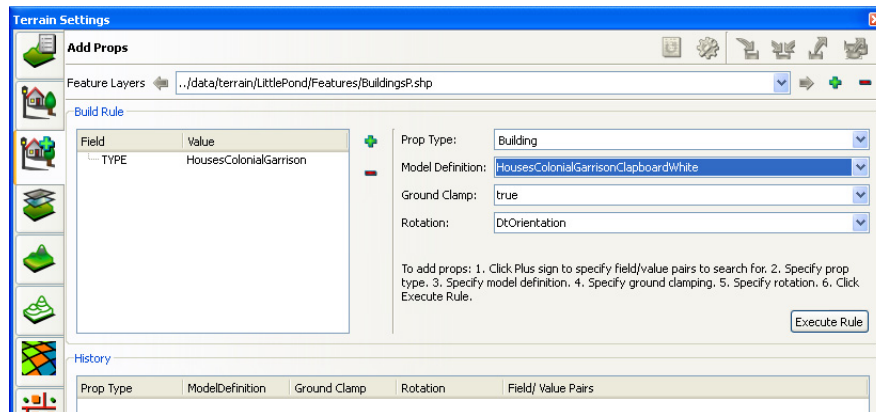


Figure C-12. Completed rule

9. Click Execute Rule. The rule is executed. The Add Props page is updated (Figure C-13) and houses are added to the terrain. Zoom in for a better look at them (Figure C-14). (You might need to close the Feature Layer Settings dialog box or move it out of the way to see the terrain.)

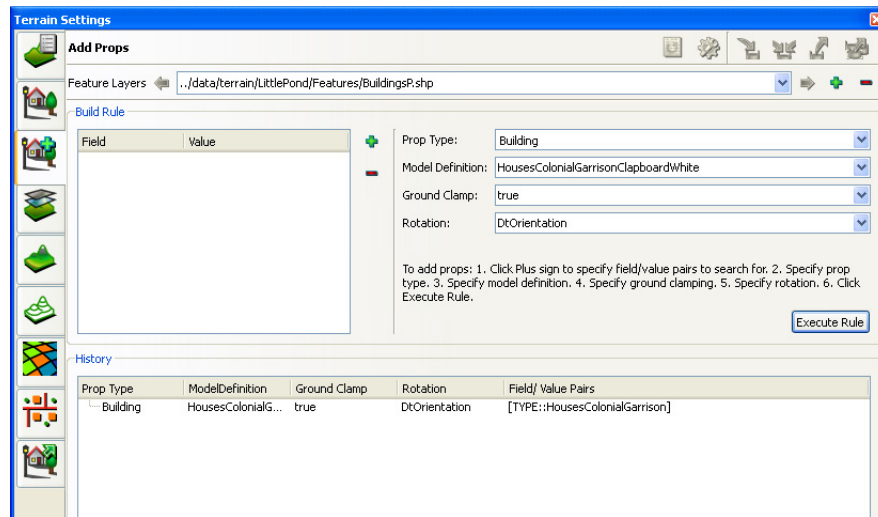


Figure C-13. Build rule implemented

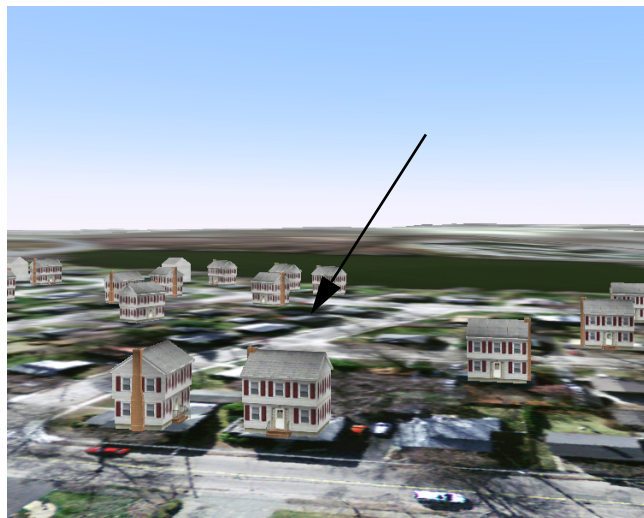


Figure C-14. Props added to terrain

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. On the Feature Layers list at the top of the dialog box, click the Add icon (+). 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-15).

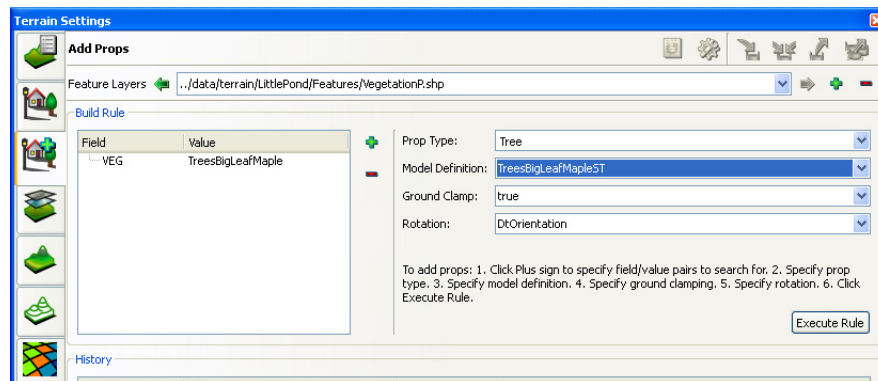


Figure C-15. Rule for big leaf maples

12. Click Execute Rule. The page is updated and trees are added to the terrain (Figure C-16).

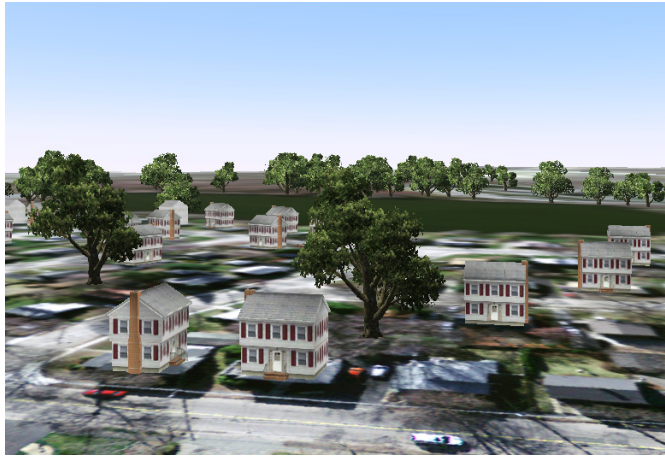


Figure C-16. Little Pond with maples

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** → **Display**. The Display Settings dialog box opens.
2. Select the Server Settings page (Figure C-17).

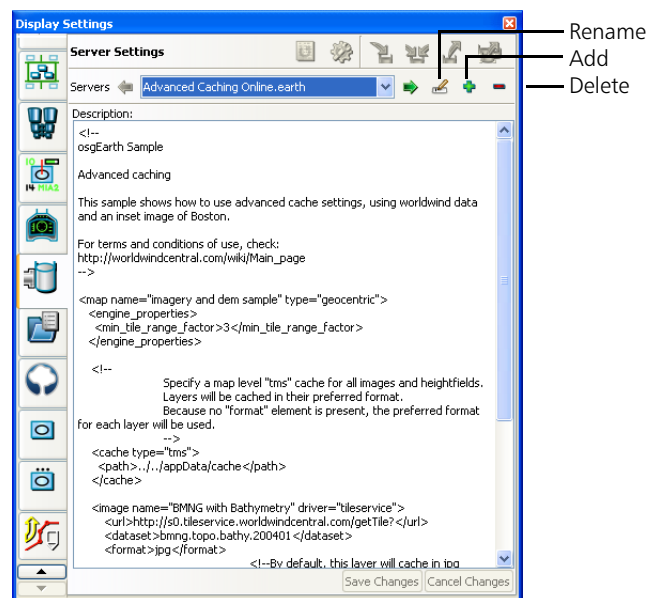


Figure C-17. Server settings page

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 imagery section of the file reads as follows:

```
<image name="world" 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. VR-Forces will use all of the files as needed.

Change the imagery URL line to:

```
<url>../../data/Terrain/SanLuisObispo/Imagery</url>
```

- 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>
```

- Click the Rename icon (🔗). (If you are prompted to save changes, click Save.) The Rename Server dialog box opens.
- Change the name to *mySanLuisObispo.earth*. (If you are prompted to save changes, click Save.) The completed server should look like [Figure C-18](#).

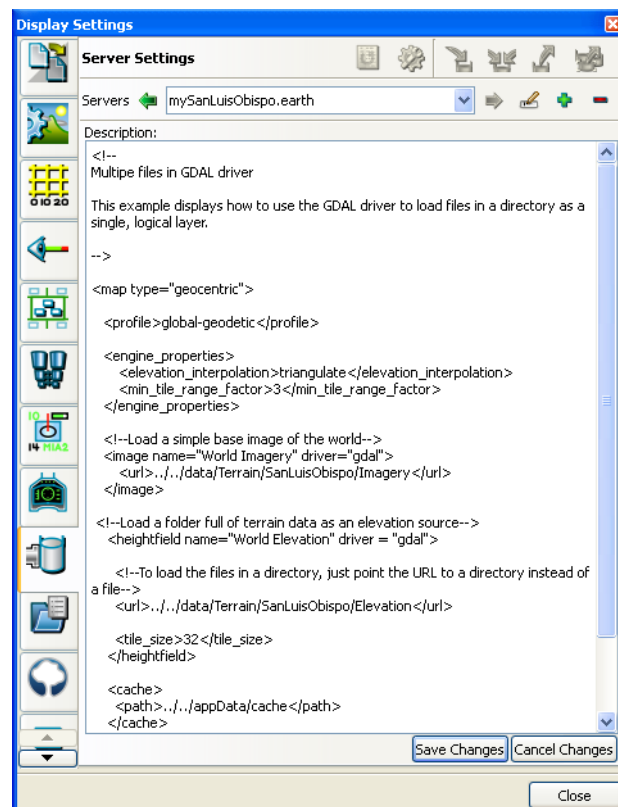


Figure C-18. SanLuisObispo.earth server

- Click Save Changes. The new server file is saved to *./userData/servers*.
- Close the Display 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 **Settings** → **Terrain**. The Terrain Settings dialog box opens.
2. Select the Terrain Contents page.
3. Click Add Terrain Server. The Add Terrain Server dialog box opens.
4. In the Servers list, select *mySanLuisObispo.earth*.
5. Click Connect. The server is loaded. It will be visible as a small speck in the window ([Figure C-19](#)), so you will have zoom in on it ([Figure C-20](#)). Notice how as you zoom in, the terrain imagery gets paged in at greater levels of detail.

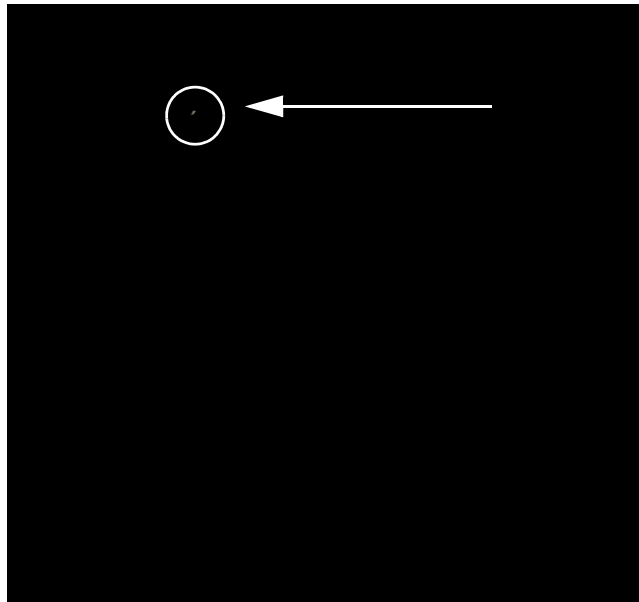


Figure C-19. SanLuisObispo.earth initial view

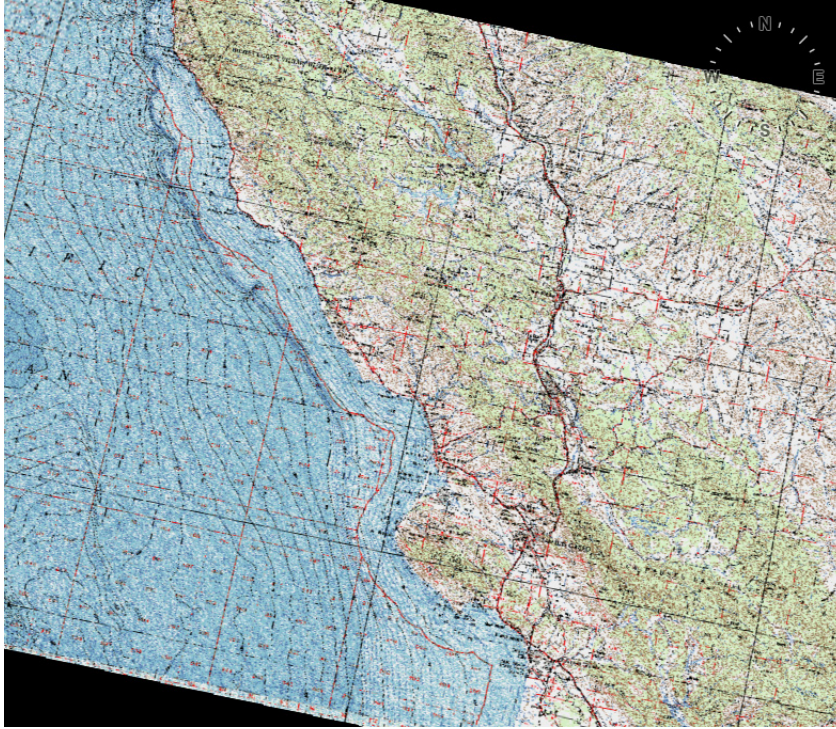


Figure C-20. SanLuisObispo.earth zoomed in

6. Choose **File** → **Save As** → **Terrain**. The Save Terrain dialog box opens.
7. Type a name, such as mySanLuisObispoEarth. (Do not name it SanLuisObispo or you will overwrite the terrain shipped with VR-Forces.)
8. Click Save.
9. Choose **File** → **Close** → **Terrain**.
10. 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 other terrain tutorials, 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-21).



Figure C-21. Makland

2. Click on the trees or buildings in the scene. Notice that you cannot select them. They are part of the terrain; they do not exist as props.
3. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
4. Select the Extract Props page (Figure C-22). On the Extract References tab, the External References window lists all of the objects in the terrain that are incorporated into the terrain by reference. These are the objects that you can extract as props. You can extract them individually. For this tutorial, we will extract all of them.

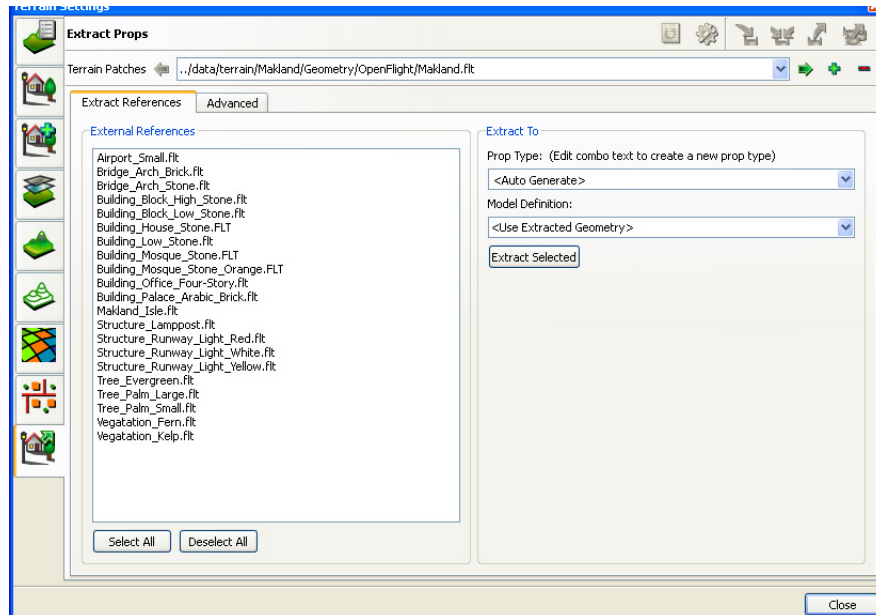


Figure C-22. Extract props page

5. Click Select All. All of the references are selected.
6. In the Extract To group box, click Extract Selected. The objects are extracted as props. The list of external references is cleared.
7. Click the trees and buildings on the plateau. You can now select them because they are props, not part of the terrain (Figure C-23). They are displayed using the geometry in the OpenFlight files from which they were referenced. You can change the models that the props use if you want to. For example, you might want to change the trees to SpeedTrees.



Figure C-23. 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 Props page. The selected prop is highlighted in the list of props.

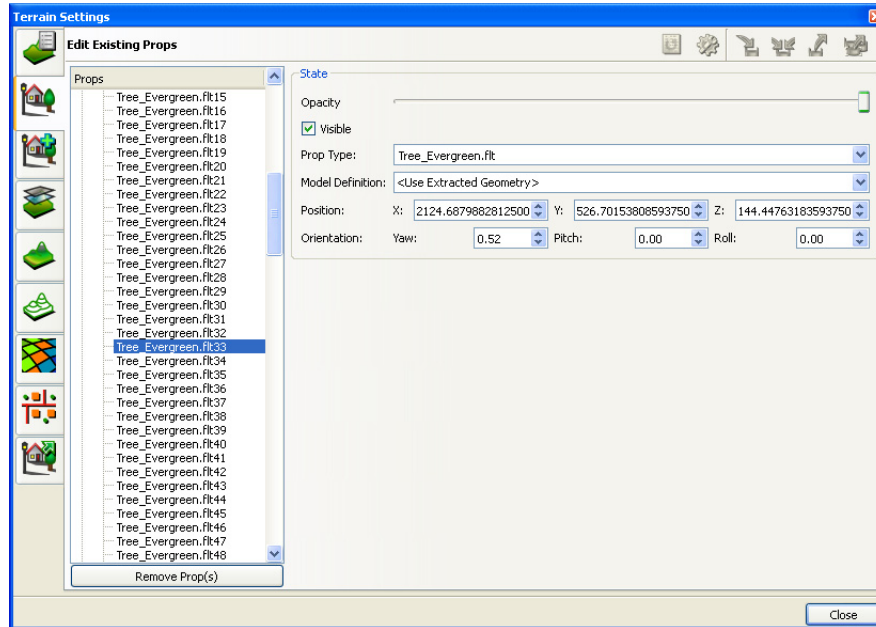


Figure C-24. Props page

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-25).



Figure C-25. New model definition for tree

5. To restore the tree to its original model, on the 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-12.) 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.](#)
2. [Add a Point Feature to the Terrain.](#)
3. Save the GDB file.
4. [Add the GDB Terrain to VR-Forces as a Feature Layer.](#)

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 → **Tools** → **TDB Tool**. The TDB Tool opens ([Figure C-26](#)).)

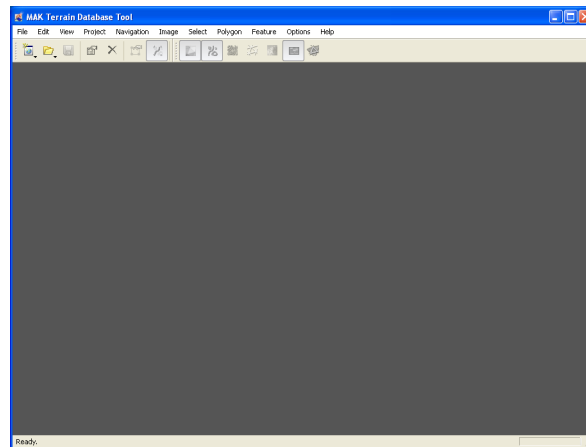


Figure C-26. TDB Tool window

2. Choose **File** → **New** → **Terrain Project**. The Utility Bar is added to the window ([Figure C-27](#)).

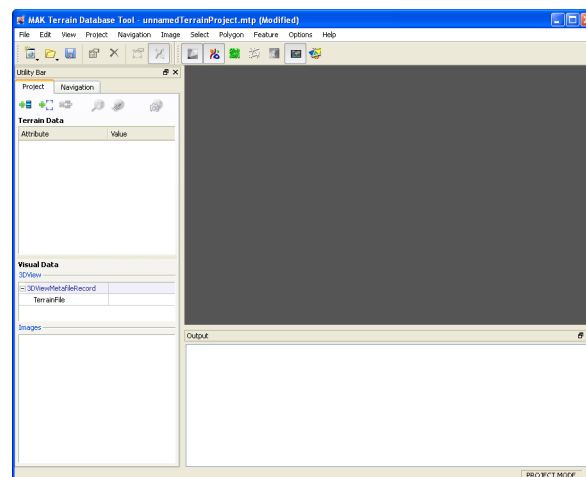


Figure C-27. New project

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-28).

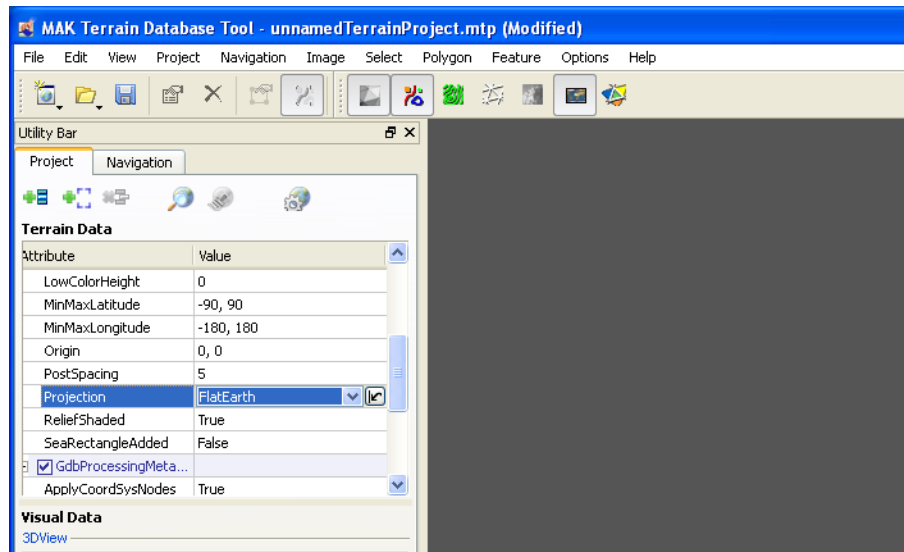


Figure C-28. Change the projection to FlatEarth

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 `myLittlePondFeature.mtd` and `myLittlePondFeatures.gdb` files.
12. Click Yes.
13. Optionally, save the project when prompted. The terrain is displayed (Figure C-29).

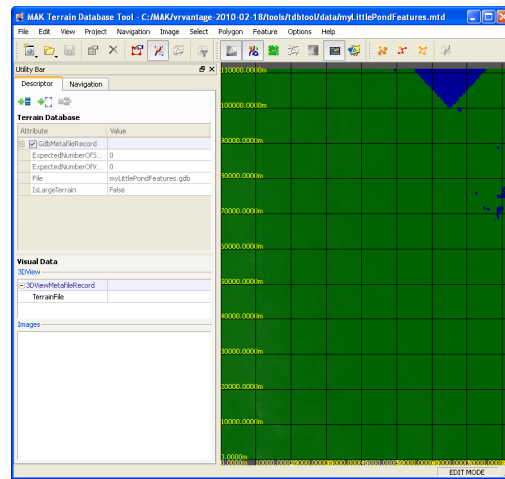


Figure C-29. New terrain

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. This tutorial uses the Little Pond terrain.

To add feature data 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-30). Its purpose is to give us an easily identified location on which to place our point features.

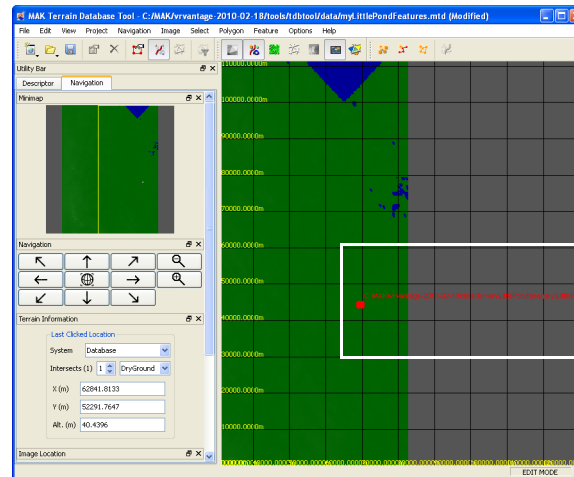


Figure C-30. Terrain with image added

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-31).



Figure C-31. 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-32).

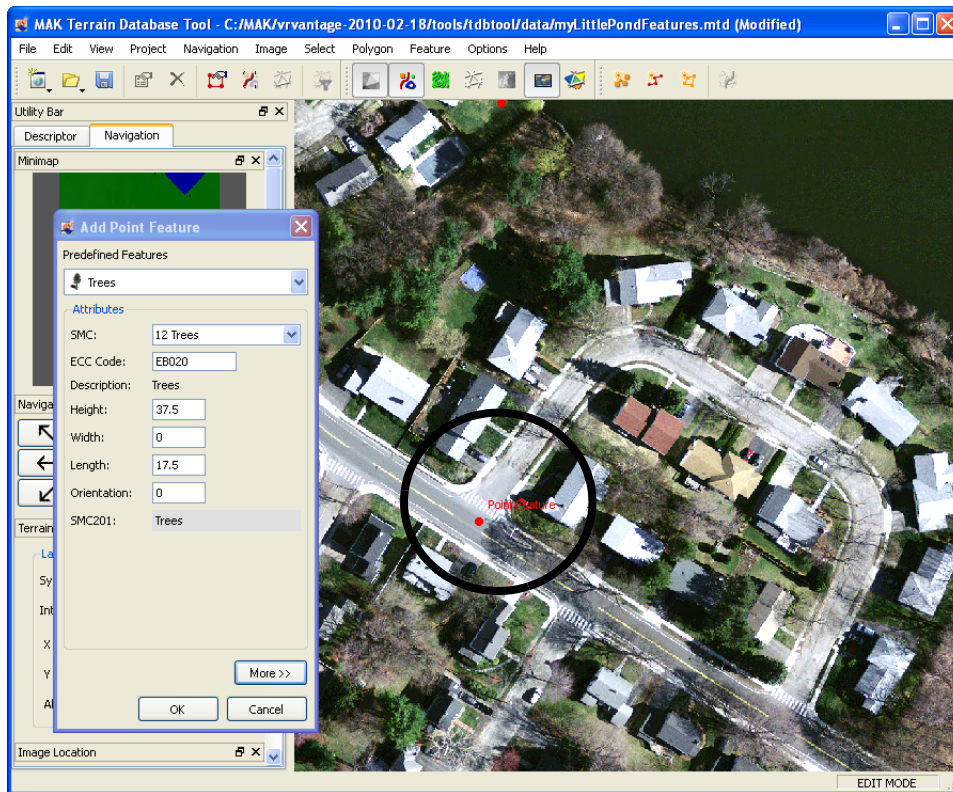


Figure C-32. Add Point Feature dialog box and new point

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-32).
8. Click OK. The feature is added to the terrain (Figure C-33).

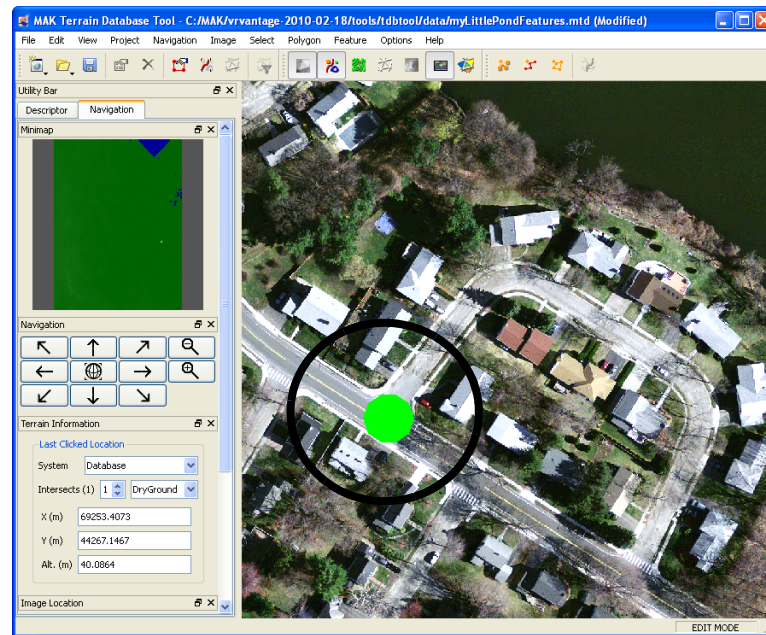


Figure C-33. Point feature added

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:

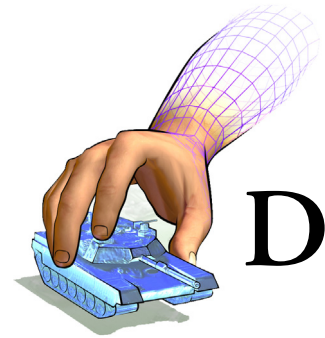
1. Start VR-Forces.
2. Change the observer mode to Stealth mode.
3. Open the *LittlePond.mtf* terrain.
4. Orient your view of the terrain so that you are viewing the area in which you created the point feature in the GDB file ([Figure C-34](#)). (If you open the Observer Views panel and load the *LittlePond.osrx* saved views file, you can select the saved view and jump right to the inset.)



Figure C-34. 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-12.



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-12

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.](#)
2. [Add an Element Definition.](#)
3. [Add a Model Mapping.](#)
4. [Test the Model Mapping.](#)

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](#)).

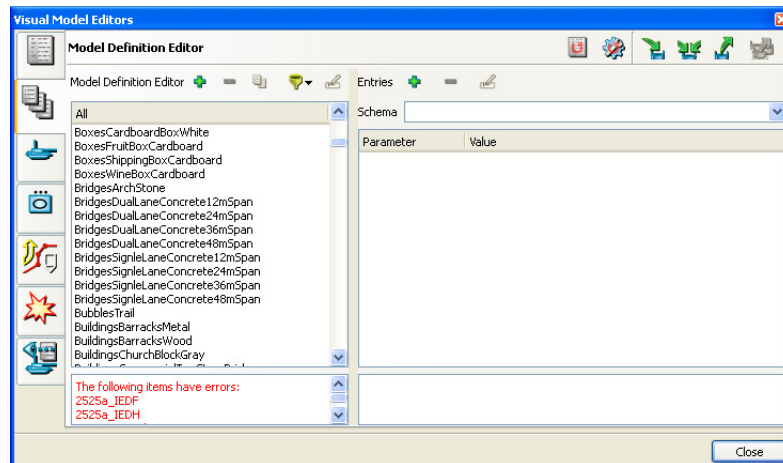


Figure D-1. Model Definition Editor page

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 13-12.)
4. Click the Add icon (+). The Add Model Definition dialog box opens ([Figure D-2](#)).

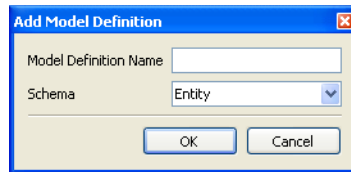


Figure D-2. Add Model Definition dialog box

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).

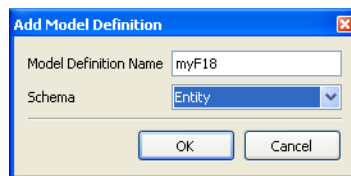


Figure D-3. New model definition

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.

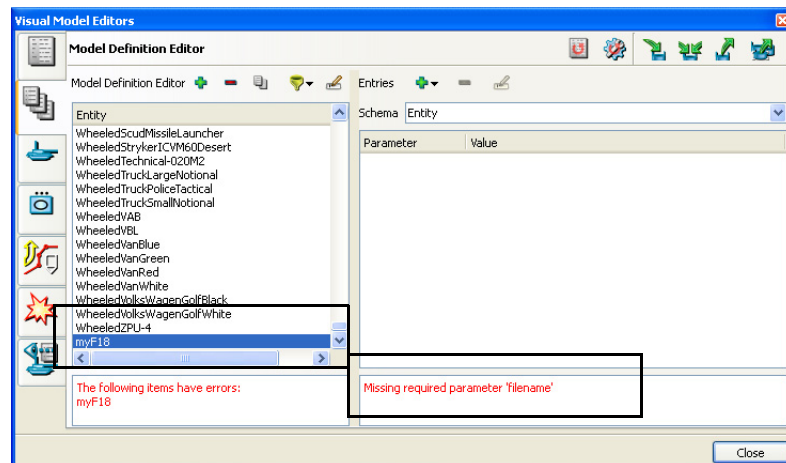


Figure D-4. New model definition in list

8. Click the Add Parameter icon (+). A list of parameters is displayed (Figure D-5).

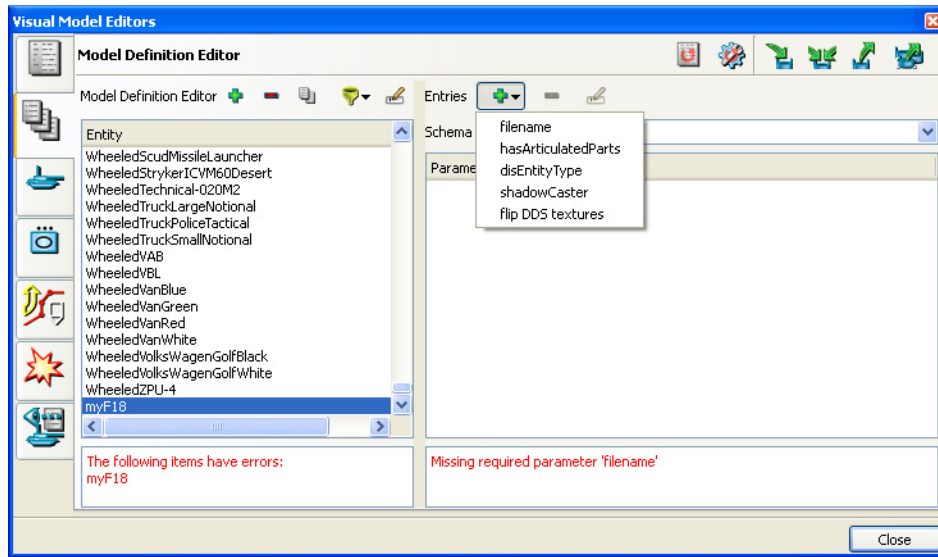


Figure D-5. List of parameters

9. Select filename. It is added to the parameter list.
10. Click the value for the parameter. A file selection dialog box opens.
11. Navigate to `./data/Vehicles/FixedWing/FA-18_Hornet` and select `FA-18_Hornet.flt`.
12. Click OK. The value is updated (Figure D-6).

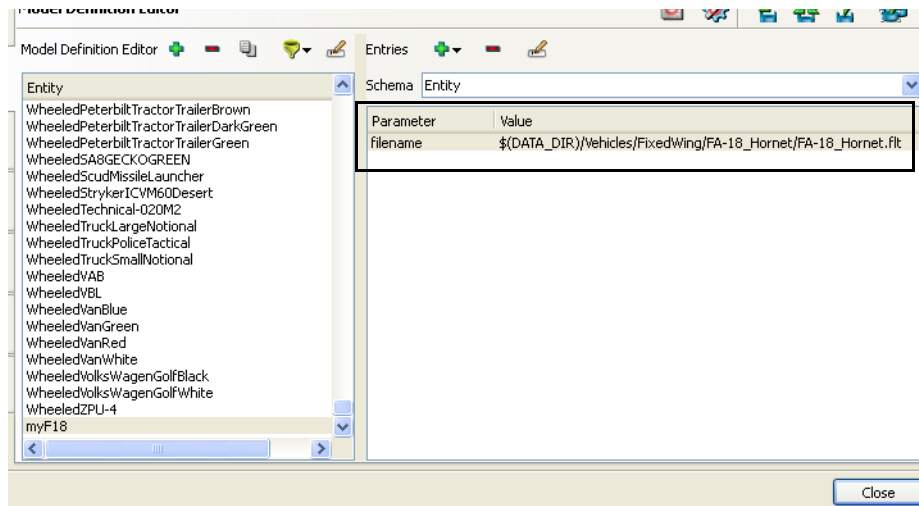


Figure D-6. Model definition parameter

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).

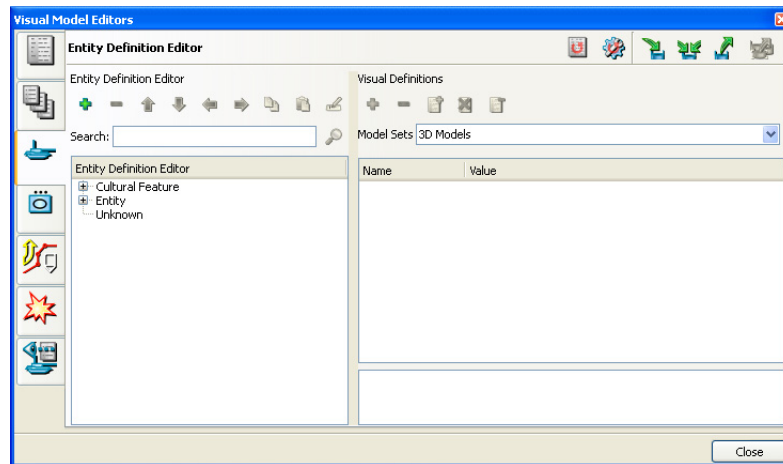


Figure D-7. Entity Definition Editor page

3. Expand the Entity section of the hierarchy and then expand the Air section.
4. Select Fixed Wing.
5. Click the Add (+). An entry called untitled is added to the list of fixed wing entities (Figure D-8). The untitled entity has the same list of visualizers as the fixed wing parent level.

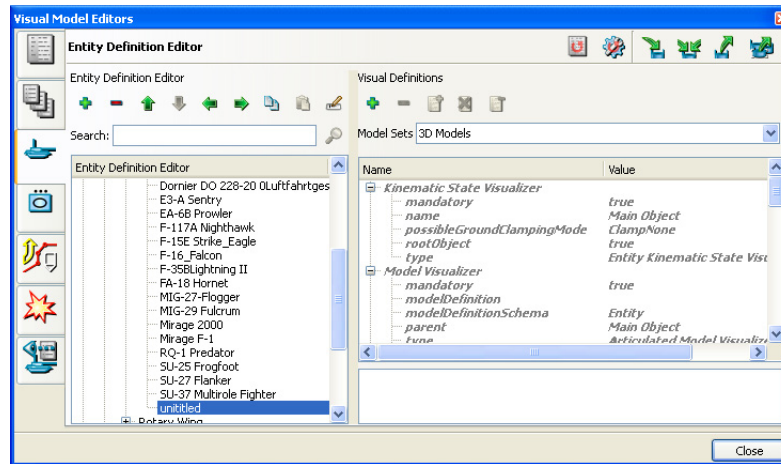


Figure D-8. New entity definition

6. Click the Rename icon ().
7. Type a name for the new entity definition, for example, myF18.
8. Click OK.
9. Scroll through the list of visualizers until you find the Model Visualizer. This visualizer specifies the model definition that this element should use.
10. Click the Value column for the Model Definition attribute to make it editable. The Model Definition for Visualizer dialog box opens (Figure D-9).

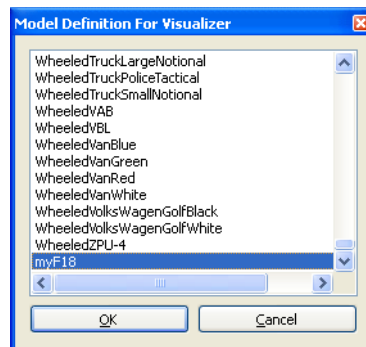


Figure D-9. Model Definition for Visualizer dialog box

11. Select the new model definition (myF18) and click OK. The model definition is now specified (Figure D-10).

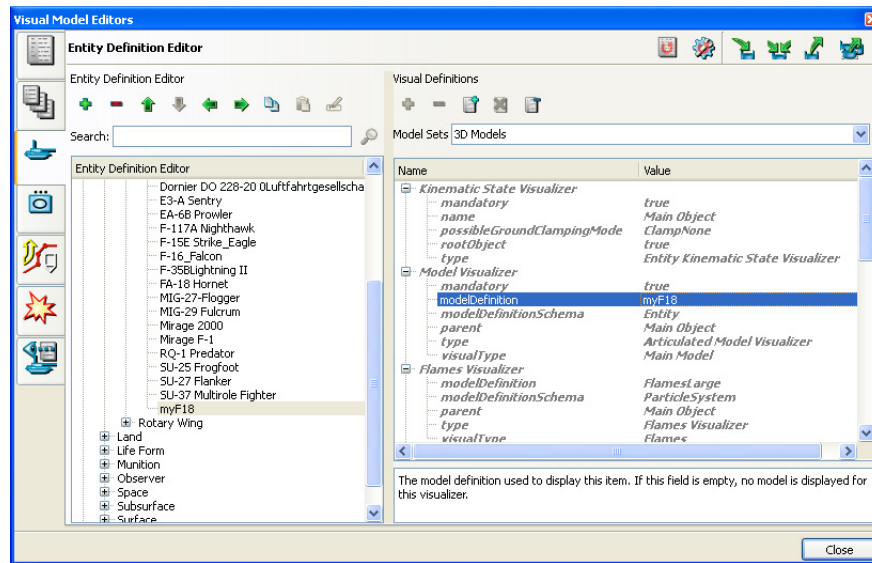


Figure D-10. Model Definition for Model Visualizer

12. 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 Mappings page (Figure D-11).

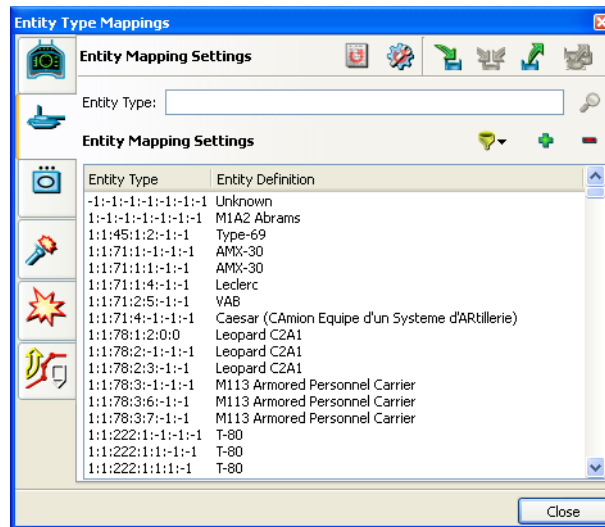


Figure D-11. Entity Mappings page

- Click the Filter icon (🔍).
- 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.)
- Click the Add icon (✚). The Create New Entity Type Model Mapping dialog box opens (Figure D-12). It lists a default entity type enumeration.

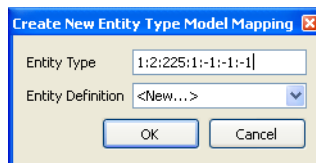


Figure D-12. Create New Entity Type Model Mapping dialog box

6. In the Entity Type text box, type the enumeration you want to map to the new model mapping. (The enumeration for an F/A18 is 1:2:225:1:9:1:-1.)



VR-Forces already has an F/A18 mapped to this enumeration. However you can map multiple model definitions to an enumeration. When an 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 Element Definition drop-down list, select the model definition that you added in the previous procedure (myF18) (Figure D-13).

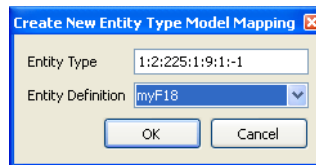


Figure D-13. Create new model mapping

8. Click OK.
9. The new model mapping is added to the list.

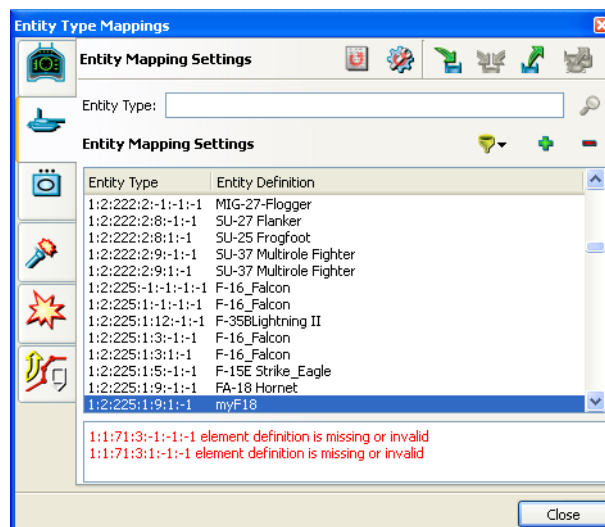


Figure D-14. New model mapping

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-15](#)).

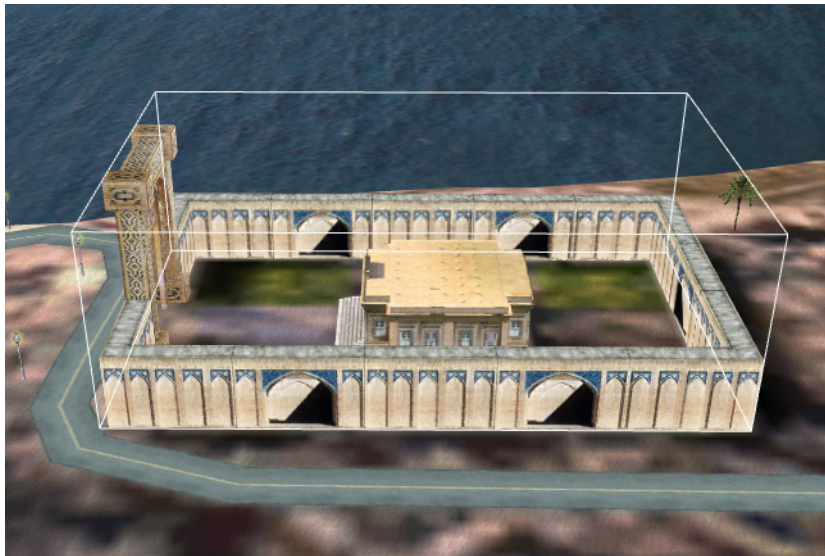


Figure D-15. Palace

4. Choose **Settings** → **Terrain**. The Terrain Settings dialog box opens.
5. Select the Edit Existing Props page. The list of props shows that Building26, the palace, is selected ([Figure D-16](#)).

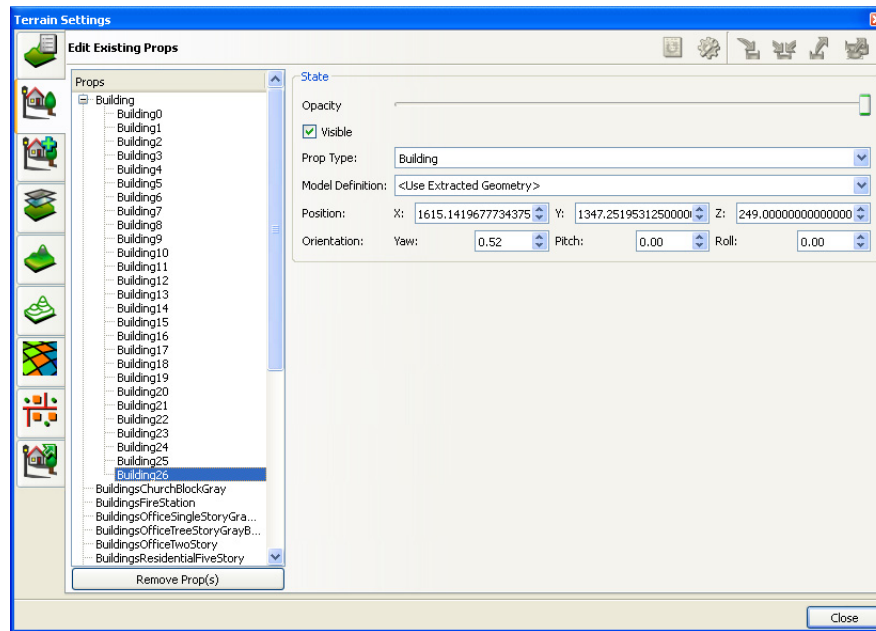


Figure D-16. Edit Existing Props page

6. In the Model Definition drop-down list, select the model definition that you added earlier in this tutorial (Figure D-17). The model of the palace is replaced with the model for the F/A-18 (Figure D-18). If this is the model that you wanted to add, then you know that you completed the procedure correctly. If this is not the model, you expected, check the filename you specified for the model definition.

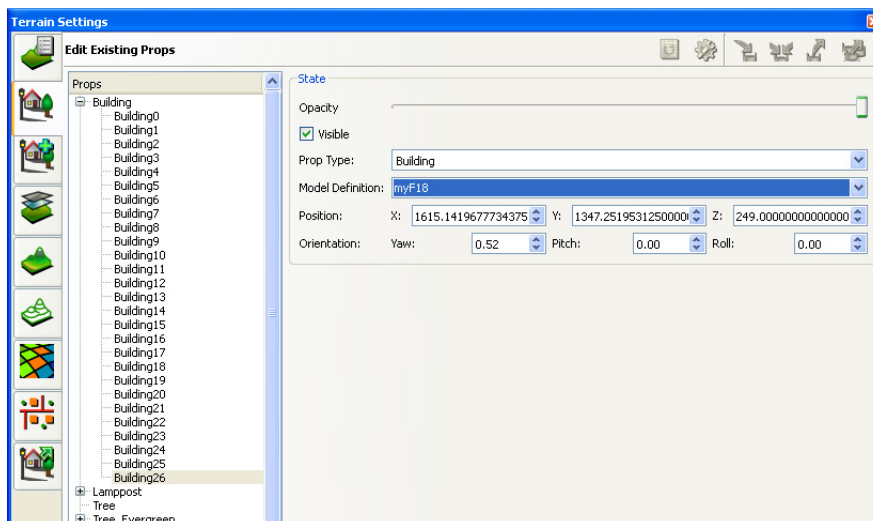


Figure D-17. Changed model definition



Figure D-18. F/A-18 replaces the palace

7. If you are going to use this terrain for simulations, change the model definition back to Use Extended Geometry.

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-19](#)).

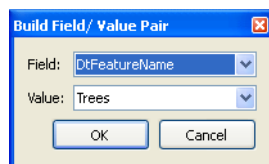


Figure D-19. Build Field/Value Pair dialog box

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-20).

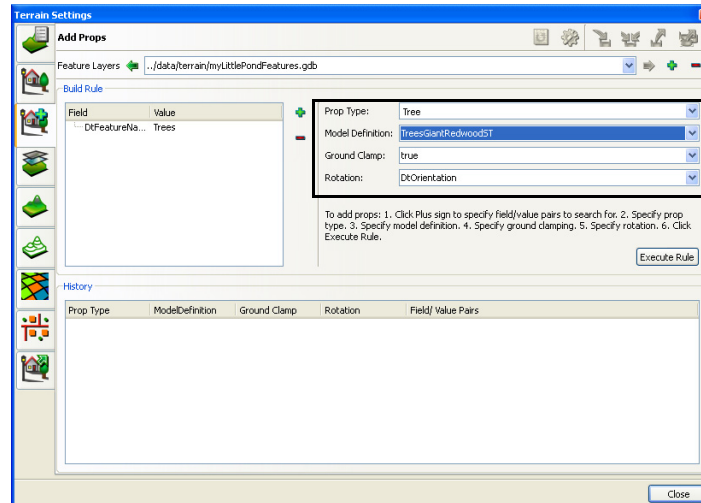
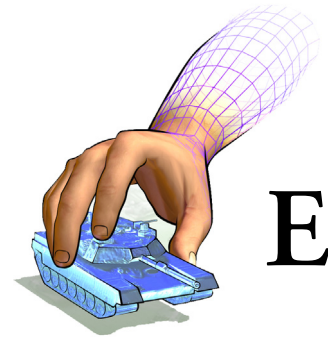


Figure D-20. Full build rule specified

11. Click Execute Rule. The tree is added to the scene where the point feature was located in the GDB file (Figure D-21).



Figure D-21. New tree in Little Pond



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-5
Using Combined Key Mappings	E-6
Filtering the Function List	E-6
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 maps that we think will meet the needs of most VR-Forces users. However, you can edit the default keyboard maps and create new mappings of your own with the Key Mapping Editor.

To see a list of all keyboard and mouse button mappings, please see [“VR-Vantage Quick Reference Card”](#).

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.

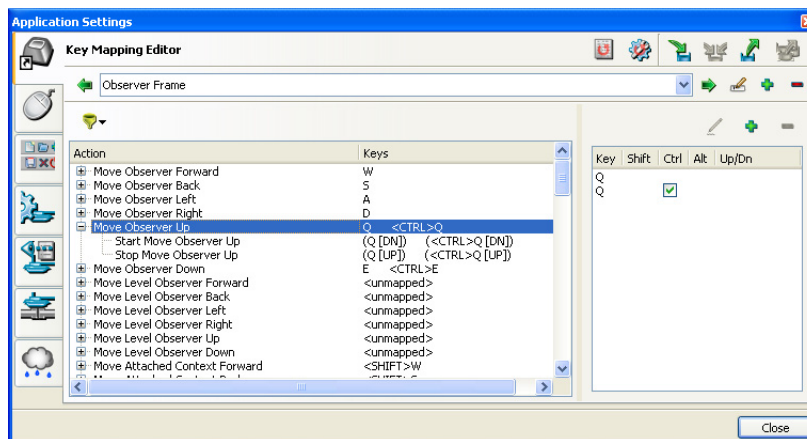


Figure E-1. Key Mapping Editor

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 Keyboard Mappings 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-5.
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. Select the function to which you want to add a key mapping.
2. Click the Add icon (+) (Figure E-2). The Key Entry dialog box opens (Figure E-3).

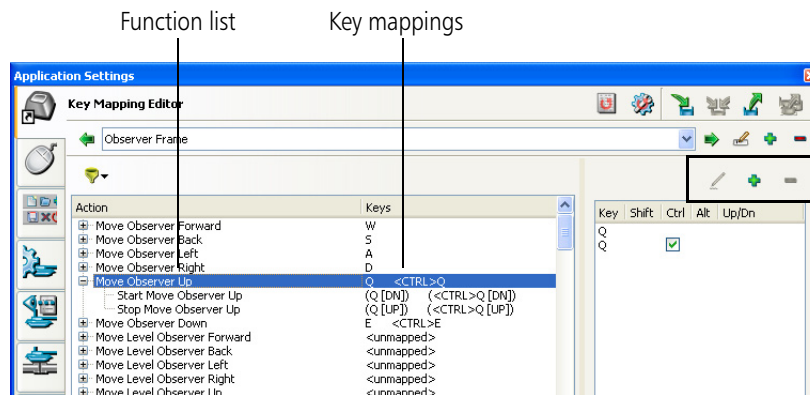


Figure E-2. Key Mapping Editor

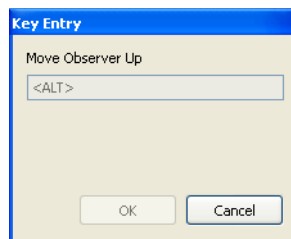


Figure E-3. Key Entry dialog box

3. On the keyboard, press the key or key combination (such as **Ctrl**+key or **Shift**+key) that you want to map to this function.

4. 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 Select 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 Select Key Map window (Figure E-2).
2. In the Select 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.

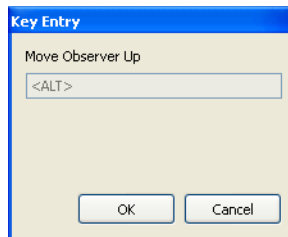


Figure E-4. Key Entry dialog box

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 Select Key Map window (Figure E-2).
2. In the Select Key Map window, select the key mapping that you want to delete.
3. Click the Delete icon (✖). 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.

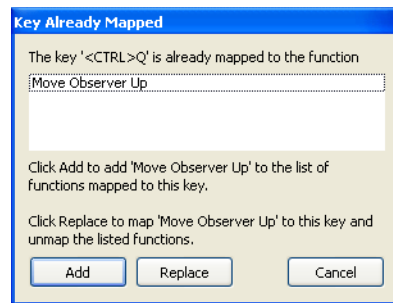


Figure E-5. Key Already Mapped dialog box

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

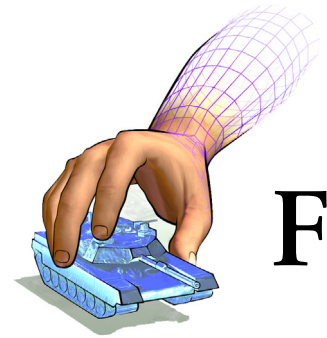
To create a key map:

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.



The WRM Specification

Some OpenFlight files contain certain 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 Simulations 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.

Background.....	F-2
General Information	F-3
Articulated Parts.....	F-4
Entity Appearances	F-5
Appearance Attributes Supported by VR-Forces Models.....	F-6
Animation.....	F-8

F.1. Background

In order for VR-Forces to change the state of a model based on incoming information, certain information about the model must be available to it. 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 within an OpenFlight format model so that VR-Forces can make these associations. For the most part, information is associated with a MultiGen bead by including it as a comment associated with the bead. Comments that contain information that is meant to be interpreted by VR-Forces 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 about any of the sections of this chapter, 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 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 stealths from other vendors.

F.2. General Information

This section lists a series of requirements for meeting the WRM specification:

- ♦ 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.
- ♦ The model's orientation should be such that the positive X axis points through the right side of the vehicle, the positive Y axis points through the front of the model, and the positive Z axis points through the top of the model. This is the normal orientation of OpenFlight models.
- ♦ The top-level node of the whole model should have the following two-line comment:

```
@dis version_number spec_version  
@dis entity_type entity_type
```

for example,

```
@dis version_number 1  
@dis entity_type 1:1:225:1:1:1:0
```

The first line indicates that the model conforms with version 1 of the model specification. The second line indicates that this model represents an M1A1 Abrams tank, since 1:1:225:1:1:1:0 is the enumeration for that entity type.

i

VR-Forces ignores the entity type line in the comment, because it uses its model mapping configuration to associate models with entity types.

- ♦ Remember that all comments will be parsed by VR-Forces, so spelling counts.

F.3. 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 bead for each articulated part. (This means that if there are several LODs, they should be children of the DOF, and not vice-versa.) Each articulation DOF should be a descendant of the DOF for the part to which it is attached (unless of course 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 bead must have all of the geometry and other beads 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.4. 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 beads are used to represent a choice between several different children, based on a particular appearance attribute.

We define a switch bead as a group bead 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.) [“Appearance Attributes Supported by VR-Forces Models,”](#) on page F-6, lists the attributes you can use with VR-Forces.

The children of the switch bead represent the different portions of a model among which VR-Forces will select, based on the appearance attribute. If a switch bead 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 bead 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 bead 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 beads 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 beads. 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 bead method described previously.

F.4.1. Appearance Attributes Supported by VR-Forces Models

VR-Forces models support the following appearance attributes:

- ♦ All entity types:
 - mobility
 - fire_power
 - damage
 - trailing_effects
 - hatch
 - moving
 - paint_scheme (HLA only)
 - density (HLA only)
- ♦ Platform entities support:
 - frozen_status
 - power_plant_status
- ♦ Land platform entities:
 - launcher
 - camouflage_type
 - concealed
 - head lights, tail lights, brake lights, blackout lights, blackout brake lights, spot lights, interior lights
- ♦ Air platform entities:
 - afterburner
 - landing lights, navigation lights, anticollision lights, formation lights, spot lights, interior lights
- ♦ Munition entities:
 - launch_flash

- ♦ Non-lifeform entities:
 - smoke
 - flaming
- ♦ Surface
 - running lights, spot lights, interior lights
- ♦ Subsurface
 - running lights
- ♦ Sensor
 - lights, blackout lights, interior lights
- ♦ Cultural features
 - exterior lights, interior lights
- ♦ Lifeform entities:
 - life_form_state
 - weapon_1
 - weapon_2
 - flashlight
 - compliance
 - camouflage_type
 - health.

F.5. Animation

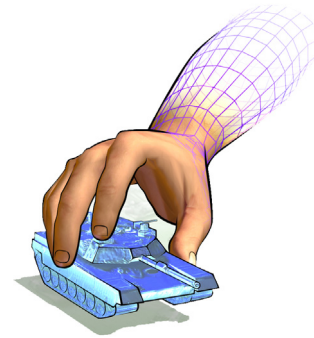
By default, animation sequences are displayed as quickly as possible. However, you can configure the animation sequence rate using a WRM animation comment in the animation node's comment field.

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.



Index

Numerics

-1 (wildcard) 7-8
 2525B icon 13-21
 2D icon, editing 13-21

A

--targetFrameRate command-line option 1-12
 absolute, pathname 3-6
 active stereo 10-18
 add image attribute 11-7
 Add Model Definition dialog box 13-10
 Add Soil Type dialog box 6-12
 Add Terrain Patch dialog box 11-5, C-3, C-17
 Add Terrain Server dialog box C-15
 adding
 category 5-34
 channel 10-5
 cockpit model 13-23
 component to object in OPD Editor 6-8
 entities to Create menu A-5
 entity to object parameter database 7-17
 feature layer 11-10
 formation 5-20
 ingress or egress points 8-30
 key mapping E-4
 model mapping 12-3
 new preconfigured aggregate 7-25
 object in OPD Editor 6-7
 parameter
 model definition 13-12
 schema 13-7
 prop 11-19
 from feature layer 11-21
 scenarios to Scenario Merge project 4-3
 sensor signature 8-5
 sensor type 8-5
 soil type parameter 6-12
 subordinate object 6-11
 subordinate to aggregate 5-17
 system to entity 5-26
 terrain patch to terrain 11-4
 terrain server 11-13, C-13
 visualizer 13-18
 window 10-3
 additional-ops parameter 3-6
 addMulticastAddr parameter 2-6
 advisory message, enabling and disabling 2-4, 2-6
 aggregate B-8
 adding new preconfigured type 7-25
 adding preconfigured to object parameter
 database 7-23
 bounding box, Entity Editor 5-21
 connection mode 2-9
 creating 5-15
 editing 5-15
 editing preconfigured 7-24
 formation
 adding 5-20
 automatic layout 5-23
 copying 5-22
 default 5-21
 editing 5-15
 manual layout 5-23
 removing 5-20
 renaming 5-20
 snap to grid 5-24
 formations 8-15

- aggregate B-8 (continued)
 - parameter-type 7-23
 - subordinate
 - adding 5-17
 - changing order 5-18
 - editing 5-19
 - removing 5-18
 - replacing 5-18
- aggregate-formation-controller** parameter 8-15
- 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 8-25
- alpha 13-29
- altitude
 - observer, filtering entities by 1-8
- ammo** parameter 8-10
- ammo select table 8-10
- anaglyphic stereo 10-18
 - configuring 10-19
- angle, damage 8-13
- Angle of Attack 13-25
- angle-of-incidence** parameter 8-13
- angleSegmentMax** parameter 8-22
- animation
 - DI-Guy, choosing 12-6
 - model F-8
- AOA, updater 13-25
- API, scenario merge 4-2
- appearance
 - DI-Guy, random 5-14
 - specifying for DI-Guy model 5-13
- appearance attribute, list of supported F-6
- appldRange** parameter 2-6
- application number 3-11
 - configuring 2-2
- Application Settings dialog box 1-11
- applying updated model definitions 13-12
- appNumber** parameter 2-2
- arc, segment 8-22
- arrow, forward and back 5-19, 5-29
- articulated part F-2, F-4
- assertOnBlockingTerrainCalls** parameter 2-3
- asynchronous I/O 2-7
 - using for performance 1-8

- attribute
 - appearance F-6
 - cockpit updater 13-23
 - window 10-6
- attributes, image 11-7
- Auto Generate 11-19
- auto-reorganize** parameter 3-5
- avoiding, objects 8-27

B

- back arrow 5-19, 5-29
- back-end
 - console 2-3
 - heartbeat 2-5
 - multiple 3-11
 - reloading OPD 2-4
- ballistic gun, controller descriptor 8-10
- batch file, configuring 2-3
- batchScenarioFileName** parameter 2-3
- bead
 - DOF F-4
 - switch F-5
- binary key mapping E-3
- binding, system definition file variable 7-22
- bitmap, displaying 11-6
- blend image attribute 11-7
- bounding box, aggregate 5-21
- building
 - adding to terrain C-21
 - MetaFlight 11-31
 - prop type 11-24
 - terrain 11-4

C

- cache
 - clearing 11-27
 - clearing model instancing 12-7
 - model instance 12-7
- caching, file 11-26
- camera, position and orientation 10-17
- Camera Orientation Offset 10-8
- Camera Position Offset 10-8, 10-17
- catastrophic-kill** parameter 8-13
- category
 - adding 5-34
 - changing 5-9
 - changing echelon 7-26
 - removing 5-34, 5-35

- category (continued)
 - renaming 5-34, 5-35
 - undoing changes 5-35
- category** parameter 7-17, 7-26
- cgfDispatcherReceivePort** parameter 2-3
- changing
 - channel attributes 10-8
 - component priority 6-9
 - echelon ID category and echelon ID level 7-26
- entity
 - category 5-9
 - enumeration 5-11
 - force 5-10
 - name 5-11
 - parameter 5-12
 - view in OPD Editor 6-4
- field-of-view 10-13
- frame rate mode 3-6
- grid scale factor 5-24
- image display order 11-9
- key mapping E-5
- model definition, prop C-20
- prop
 - model definition 11-25
 - position or orientation 11-25
 - type 11-24
- site ID 3-11
- subordinate order 5-18
- terrain database for scenario 3-8
- viewport 10-14
- window attributes 10-6
- channel
 - adding 10-5
 - changing attributes 10-8
 - keyword 10-8
 - multiple 10-15
 - projection resize policy 10-11
 - property 10-8
 - removing 10-6
- Channel Name 10-8
- ChannelKeyword** parameter 13-25
- character, specifying for DI-Guy model 5-13
- character_animations_table.mtl* 2-8
- character_digest.xml* 2-8
- child, element definition 13-15
- clamp image attribute 11-7
- clamp to border image attribute 11-7
- clamp to edge image attribute 11-7
- class
 - DtSimComponentManager* 7-10
 - class* (continued)
 - DtSimObjectNetInterface* 7-10
 - DtSimObjectStateRepository* 7-10
 - DtSimTaskManager* 7-10
 - DtVrfObject* 7-6, 7-10
 - DtVrfObjectBaseStateRepository* 7-6
- clearing
 - file cache 11-27
 - model instancing cache 12-7
- clipping plane 13-27
 - dynamic 10-10
- clock mode 3-6
- closing, scenario file in Scenario Merge 4-7
- cockpit display 12-2, 13-23
 - adding new 13-23
 - element definition 13-15
 - installing 13-24
 - model definition, creating 13-24
 - updater 13-23
 - attribute 13-23
- coefficient probability 8-12
- collateral damage, probability table 8-13
- collision avoidance, configuring 8-27
- color
 - represents sensor volume frequency 8-20
 - sensor volume, pulse rate 8-20
- combining, key mapping E-6
- command line, running Scenario Merge from 4-8
- command-line option, --targetFrameRate 1-12
- comment, MTL parameter 2-10
- comm-model-name** parameter 8-19
- commModelParams.mtl* file 2-9, 9-3, 9-4
- communication model 2-9, 8-19
 - custom 2-10
 - setting 9-3
 - simple-radio-comm-model 2-9
- component 7-9
 - adding to object in OPD Editor 6-8
 - attaching to remote entity 8-32
 - creating connection in OPD Editor 6-10
 - deleting in OPD Editor 6-6
 - descriptor, elements of 7-14
 - editing in OPD Editor 6-9
 - entity 7-13
 - occupancy-director-controller 8-30
 - priority 6-9
 - systems 7-18
 - tuning 1-9
 - performance 1-9
- component attachment table 8-32

- Component Manager B-5
- Component-Attachment** parameter 3-5
- Component-Attachment-Table** parameter 3-5
- `componentAttachmentTable.mtl` 8-32
- component-descriptor-type 7-14
- component-manager** parameter B-5, B-7
- component-type 7-14
- compositing 11-16
- compressing
 - model and image files 12-8
 - virtual texture 11-18
- compression, texture 11-26
- conditional expression 3-9
- configuration
 - display, saving 10-6
 - display engine, loading 10-6
- configuration file
 - character_animations_table.mtl* 2-8
 - character_digest.xml* 2-8
 - commModelParams.mtl* 2-9, 9-3, 9-4
 - for object parameter database 5-4
 - `sysdef` 7-18
 - vrf.ent* A-2, A-5
 - vrfSim.mtl* 2-2
 - vrfSimSettings.xml* 1-11
- configuring
 - 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, 5-13
 - DIS 2-6
 - `drainInput` 2-4
 - `drainInput` for DIS 2-6
 - `drainInput` for HLA 2-6
 - emitter 8-17
 - exercise ID 2-6
 - External Communications Model 9-4
 - feature avoidance 8-28
 - FED file name 2-6
 - federation execution 2-5
 - name 2-5
 - FOM Mapper 2-5
 - formations 8-15
 - frequency of data sends 2-4
 - heartbeat threshold 2-6
 - HLA 2-5
 - configuring (continued)
 - interest management 1-7
 - joysticks 8-26
 - map datum 2-5
 - MetaFlight 11-15
 - model definition 13-9
 - multicast address 2-6
 - multichannel display 10-15
 - mutually exclusive tasks 8-25
 - object avoidance 8-27
 - obstruction avoidance 8-27
 - path to OPD file 2-3
 - port number 2-6
 - projection resize policy 10-11
 - pulse rate 8-20
 - radio message model 2-9
 - radios 8-19
 - receive port 2-3
 - scenario file to use 2-4
 - schema 13-3
 - sensor volume, segments 8-22
 - sensor volumes 8-20
 - sensors 8-3
 - session ID 2-4
 - simulation management message 2-5
 - site ID 2-5
 - start mode 2-5
 - stereo
 - anaglyphic 10-19
 - polarized 10-19
 - terrain database
 - path 2-3
 - to load 2-5
 - terrain server 11-14
 - time stamp order 2-6
 - visual quality slider 1-12
 - VR-Forces for external communications effects
 - server 9-2
- configuring slider 1-12
- connection
 - component 6-10
 - configuring in system definition file 7-19
 - sensor 8-3
- connection mode 2-9
 - aggregate 2-9
 - all 2-9
 - force 2-9
- connection model, external 2-10
- console, configuring for back-end 2-3
- constructing, terrain 11-4

- control object 3-11, B-7
 - parameter-type 7-27
- coordinates, UV 11-8
- copying
 - element definition 13-17
 - formation 5-22
 - model definition 13-11
 - schema 13-6
- CPU time, giving up 1-12
- Create Connection dialog box 6-10
- Create Entry dialog box 6-7
- Create menu A-2
 - adding entity to A-5
 - assigning or removing entity 5-12
 - configuring A-2
 - editing 5-29, A-4
 - manually 5-31
 - reorganizing A-4
 - reverting 5-34
- Create Menu Tree, OPD Editor 6-4
- Create New Component dialog box 6-8
- Create New Entity Type Model Mapping dialog box 12-3
- Create Resource dialog box 6-11
- create-entity-menu-item A-5
- creating
 - aggregate 5-15
 - component connection 6-10
 - element definition 13-16
 - entity A-5
 - from existing 5-28
 - new 5-28
 - key map E-7
 - MEDF and MEIF files 12-8
 - model, guidelines for 13-29
 - model definition 13-9, 13-10, D-2
 - cockpit display 13-24
 - Scenario Merge project 4-3
 - schema 13-5
 - simulation model set 5-5, 5-6
 - system definition file 7-19
 - terrain 11-3
 - user-defined formation 8-16
- cultural-feature-param** parameter 7-12, B-8
- cultural-feature-state-repository 7-12
- custom, communication model 2-10
- customizing, object parameter database 7-28

D

- daemon, running vrfSim as 2-3
- daemon** parameter 2-3
- damage
 - configuring entity to take 8-14
 - probability table 8-13
- damage actuator 8-13
- damage file 5-4, 7-27
- damage model 5-24, 5-25
 - specifying 5-26
- data
 - configuring frequency of sends 2-4
 - input 7-13
- data distribution management 1-6, 1-7
- database, object parameter 7-2
- dataset, geometry grid and virtual texture 11-31
- DDM 1-6, 1-7, 1-8
 - region size 1-7
- DDS compression 11-18
- DDS texture 11-27
 - flipping 11-5, 13-26
- decal effects 12-2
- decal image attribute 11-7
- default
 - formation 5-21
 - value, parameter 3-12
- DefaultEllipsoidArc, model definition 8-20
- defaultHeartbeatThreshold** parameter 2-6
- defaultObjectTimeout** parameter 2-6
- defaultParameterDatabasePath** parameter 2-3, 3-7
- default.sms* file 5-4, 5-6
- defaultTerrainDatabasePath** parameter 2-3, 3-7
- definition
 - element 13-15
 - parameter 6-4
- degrees of freedom F-4
- deleting
 - category 5-34, 5-35
 - channel 10-6
 - element definition 13-19
 - entity from simulation model set 5-29
 - key map E-7
 - key mapping E-5
 - model definition 13-11
 - model mapping 12-6
 - parameter
 - model definition 13-14
 - schema 13-8
 - parameters and components in OPD Editor 6-6

- deleting (continued)
 - schema 13-6
 - terrain, cached 11-27
 - window 10-5
- description, parameter 6-4
- destAddrString** parameter 2-6
- destination address 2-6
- destroyFedExec** parameter 2-5
- destroying, federation execution 2-5
- detonation, effects 12-2
- dialog box
 - Add Model Definition 13-10
 - Add Soil Type 6-12
 - Add Terrain Patch 11-5, C-3, C-17
 - Add Terrain Server C-15
 - Application Settings 1-11
 - Create Connection 6-10
 - Create Entry 6-7
 - Create New Component 6-8
 - Create New Entity Type Model Mapping 12-3
 - Create Resource 6-11
 - Display Settings
 - Interest Management Settings page 1-7
 - Render Settings page 1-5
 - Duplicate Model Definition 13-11
 - Entity Type Mappings 12-2
 - New Scenario 3-3
 - Open Model Set 5-7
 - Performance Options, configuring slider 1-12
 - Performance options 1-11
 - Visual Model Editors 13-9
- DI-Guy
 - choosing animation 12-6
 - configuring 2-3
 - configuring character and appearance 5-13
 - configuring use of 2-5
 - model definition 13-10
- diGuyAnimationsFile** parameter 2-3
- diGuyCharacterDataFile** parameter 2-3
- DirectX 11-27
- DIS
 - configuring 2-6
 - encoding information in OpenFlight format F-2
 - enumerations document 12-2
 - heartbeat, threshold 2-6
 - WRM specification F-1
- DIS Enumeration Document 7-6
- disabling
 - advisory messages 2-4, 2-6
- disabling (continued)
 - creation of an entity type 5-12
 - frame rate statistics 2-4
 - texture compression 11-26
- display
 - configuration, saving 10-6
 - field-of-view 10-13
 - perspective 10-13
- Display Configuration Panel 10-4, 10-5, 10-6, 10-7, 10-8
- display engine 10-2
 - configuration, loading 10-6
 - window, adding 10-3
- Display Settings dialog box
 - Interest Management Settings page 1-7
 - Render Settings page 1-5
- displaying
 - performance statistics 1-4
 - raster maps 11-6
 - scenario file in Scenario Merge 4-6
 - segment outlines 8-23
- disPort** parameter 2-6
- DLL
 - cockpit display 13-23
 - installing cockpit display, installing 13-24
- DOF bead F-4
- doNotUseConsole** parameter 2-3
- drainInput
 - configuring 2-4
 - configuring for DIS 2-6
 - configuring for HLA 2-6
- dr-allow-gui-overrides** parameter 1-12
- DTED, support for 11-3
- DtNotifyLevel** parameter 2-3
- DtSimComponentManager* class 7-10
- DtSimObjectNetInterface* class 7-10
- DtSimObjectStateRepository* class 7-10
- DtSimTaskManager* class 7-10
- DtVrfObject* class 7-6, 7-10
- DtVrfObjectBaseStateRepository* class 7-6
- duplicate, key mapping E-6
- Duplicate Model Definition dialog box 13-11
- dynamic clipping plane 10-10
- Dynamic Near Far Planes attribute 10-8, 10-10

E

- echelon, changing level 7-26
- echelon ID, category and echelon ID level 7-26
- echelon-level** parameter 7-23, 7-26

- editing
 - aggregate 5-15
 - aggregate subordinate 5-19
 - channel attributes 10-8
 - components in OPD Editor 6-9
 - Create menu 5-29, 5-31, A-4
 - element definition 13-18
 - embarkation ingress and egress points 8-29
 - embarkation parameters 8-30
 - entity model 5-8
 - formation 5-15
 - frustum 10-13
 - icon, 2D 13-21
 - key maps E-2, E-3
 - model definition 13-9, 13-12
 - model mapping 12-4
 - object map file 3-11
 - parameter
 - model definition 13-14
 - schema 13-8
 - parameters 5-12, 6-4
 - parameters for multiple objects 6-6
 - plan file 3-8
 - preconfigured aggregate 7-24
 - scenario file 3-3
 - in Scenario Merge 4-7
 - sensor 8-4
 - sensor signature 8-5
 - system 5-27
 - system properties 5-27
 - systems 6-13
 - terrain server configuration 11-14
 - visualizer 13-19
 - window attributes 10-6
- effects, mapping 12-2
- egress point 8-29, 8-30
- egress-points** parameter 8-29
- electromagnetic emissions, segment angle 8-22
- element, parent 13-19
- element definition 13-15
 - adding visualizer 13-18
 - copying 13-17
 - creating 13-16
 - deleting 13-19
 - editing 13-18
 - filtering list in Entity Type Mappings dialog box 12-5
 - rename 13-17
 - reorganizing 13-20
 - saving 13-20
- embarkation
 - editing ingress and egress points 8-29
 - occupancy-director-controller 8-29
 - slots 8-29
- embarkation-slot-entry** parameter 8-29
- embarkation-slots** parameter 8-29
- embedded window 10-2
- EM-Emission update message 8-21
- emitter, configuring 8-17
- Emitter Power 8-21
- enabling
 - advisory messages 2-4, 2-6
 - creation of an entity type 5-12
 - External Communications Model 9-4
 - frame rate statistics 2-4
 - radio transmitter 9-3
 - texture compression 11-26
- entity
 - adding new to parameter database 7-17
 - appearance F-2
 - category, changing 5-9
 - changing
 - force 5-10
 - name 5-11
 - undoing changes 5-14
 - component, attaching to remote 8-32
 - components 7-13
 - configuring
 - damage by munitions 8-14
 - emitter 8-17
 - obstruction avoidance 8-27
 - creating from existing 5-28
 - creating in Entity Editor 5-28
 - creation, enabling and disabling 5-12
 - deleting from simulation model set 5-29
 - editing 5-3
 - element definition 13-15
 - filtering 1-6, 1-7
 - model, editing 5-8
 - models 5-4
 - systems 5-24
 - adding 5-26
 - tick rate 1-9
 - type 12-2
 - enumeration 12-3
 - vrf.ent* file A-2, A-5
- entity definition, renaming D-6
- Entity Editor 5-3
 - configuring DI-Guy character and appearance 5-13

Entity Editor 5-3 (continued)

- force
 - adding 5-36
 - editing 5-37
 - removing 5-38
- formation, bounding box 5-21
- managing forces 5-36
- navigating edited entities 5-29
- opening OPD Editor from 5-27
- starting 5-4
- entity enumeration, changing 5-11
- Entity List, OPD Editor 6-4
- Entity Mappings page 12-3, 12-4, 12-6
- Entity Palette, assigning or removing entity 5-12
- Entity Resource Manager 8-10
- Entity Tree, OPD Editor 6-4
- entity type, changing 5-11
- Entity Type Mappings dialog box 12-2
 - filtering element definition list 12-5
- entity-type** parameter 8-11, A-5
- entVrfDataTimeThreshold** parameter 2-4
- enumeration 7-6, 12-3
 - document 12-2
 - entity, changing 5-11
 - object type B-9
- Enumeration and Bit Encoded Values for Use with Protocols for DIS Applications 7-6
- environment variable
 - MetaFlight 11-18
 - OpenSceneGraph, anaglyphic stereo 10-19
- execName** parameter 2-5
- exercise ID 2-6
- exerciseld** parameter 2-6
- exercise-start-time** parameter 3-6
- external communication model 2-10
 - enabling 9-3
- external communications effects server
 - configuring VR-Forces for 9-2
 - QualNet 9-5
- External Communications Model
 - configuring 9-4
 - enabling 9-4
- external communications model 9-3
- external reference 13-29
- extracting
 - prop C-17
 - props from terrain patch 11-19

F

- far clipping plane 10-10
- feature
 - adding to terrain C-21
 - avoiding 8-27
 - configuring avoidance of 8-28
 - extracting as prop 11-20
- feature layer
 - adding 11-10
 - adding prop from 11-21
- FED file, name 2-6
- federate, name 2-6
- federateName** parameter 2-6
- federation execution
 - destroying 2-5
 - name 2-5
- fedFileName** parameter 2-6
- Feet Above Ground Level 13-25
- Feet Above Mean Sea Level 13-25
- FeetAGL, updater 13-25
- FeetMSL, updater 13-25
- FeetPerMinute, updater 13-25
- field of view 8-21
- field-of-view 10-13, 10-14
 - changing 10-13
- file
 - cache, clearing 11-27
 - caching 11-26
 - compressed 12-8
 - default.sms* 5-4
 - formation 8-15
 - joyParam.dat* 8-26
 - MTL 2-10
 - object map 3-11
 - order of battle 3-11
 - plan 3-8
 - scenario 3-3, 3-6
 - simulation model set 5-4
 - vrf.ent* A-2, A-5
 - vrfSim.mtl* 2-2
 - vrfSim.opd* 7-2
 - vrfSimSettings.xml* 1-11
- File Cache Settings page 11-26, 11-27
- File Caching page 12-7
- FILE LOD GEOM 11-31
- Filename** parameter 13-24
- filtering
 - element definition list 12-5

- filtering (continued)
 - entities
 - by observer altitude 1-8
 - in scene 1-6
 - function list E-6
 - model definition list 13-12
 - firing effects 12-2
 - 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 7-12
 - fixed-wing-entity-param** parameter B-8
 - fixed-wing-entity-state-repository 7-12
 - flaming effects 12-2
 - flipping, DDS textures 11-5, 13-26
 - FLT file, mapping to models 12-2
 - FOM Mapper
 - configuring 2-5
 - initialization string 2-5
 - fomMapperInitData** parameter 2-5
 - fomMapperLib** parameter 2-5
 - force
 - adding 5-36
 - color, changing 5-37
 - connection mode 2-9
 - editing 5-37
 - entity, changing 5-10
 - hostility, changing 5-37
 - name, changing 5-37
 - removing 5-38
 - Force-Hostility** parameter 3-5
 - forceParameterDbReload** parameter 2-4
 - force-type** parameter A-4, A-5
 - format, OpenFlight F-2
 - formation
 - adding 5-20
 - bounding box 5-21
 - configuring 8-15
 - copying 5-22
 - default 5-21
 - editing 5-15
 - expanding and collapsing in Formation Editor 5-21
 - file 5-4, 8-15
 - layout
 - automatic 5-23
 - manual 5-23
 - removing 5-20
 - renaming 5-20
 - formation (continued)
 - snap to grid 5-24
 - user-defined 8-16
 - Formation Editor 5-20, 5-21, 5-22, 5-23
 - forward arrow 5-19, 5-29
 - frame, length of 3-6
 - frame rate 1-3, 1-12
 - 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 8-20
 - frustum 10-8, 10-17
 - editing 10-13
 - full screen window 10-2
 - function, movement E-2
 - function list, filtering E-6
- ## G
- GDB terrain, shapefile 11-10
 - Geo TIFF image 11-8
 - geocentric terrain 11-11
 - geographic DDM 1-6, 1-7
 - geometry 13-29
 - sharing 12-7
 - geometry grid dataset 11-31
 - geometry offset factor, MetaFlight 11-17
 - GL Studio 13-23
 - Global-Plan** parameter 3-5
 - grid scale factor, changing 5-24
 - ground vehicle, configuring obstruction avoidance 8-27
 - ground-entity-state-repository 7-12
 - ground-vehicle-param 7-12
 - ground-vehicle-param** parameter 7-6, B-8
 - guidelines 13-29
 - creating models 13-29
 - Gui-Terrain-Database** parameter 3-5
- ## H
- header file, *objTypeEnums.h* B-9
 - Headlight, updater 13-26
 - heartbeat
 - back-end, configuring 2-5
 - DIS 2-6
 - relationship to thresholds 1-11

- hierarchy
 - element definition 13-15
 - changing 13-20
- hit probability file 7-27
- hit probability table 8-11
- HLA 1-6, 1-7
 - parameters 2-5
- horizontal navigation E-2
- HUD. See cockpit display
- human-param** parameter 7-12, B-8
- human-state-repository 7-12

I

- icon
 - 2525B 13-21
 - 2D, editing 13-21
- ID, site 2-5
- ignore list 4-4
- ignoreAdvisories** parameter 2-4
- image
 - attributes 11-7
 - compressing 12-8
 - display order 11-9
 - displaying 11-6
 - GeoTIFF 11-8
 - texture coordinate system 11-8
- individual-lifeform-param 7-12
- individual-lifeform-state-repository 7-12
- information message 2-3
- ingress point 8-29, 8-30
- ingress-points** parameter 8-29
- inheritance, element definition 13-15
- initialization string, FOM Mapper 2-5
- input
 - of data 7-13
 - port and port group 7-14
- interest management 1-6, 1-8
 - configuring 1-7
- Interest Management Settings page 1-7
- intersection, intervisibility 1-8
- intervisibility, intersection 1-8

J

- joy-device-type** parameter 8-26
- joyParam.dat* file 8-26
- joy-rot-gain** parameter 8-26
- joy-rot-invert** parameter 8-26
- joy-rot-sensitivity** parameter 8-26

- joystick, configuring 8-26
- joy-trans-gain** parameter 8-26
- joy-trans-invert** parameter 8-26
- joy-trans-sensitivity** parameter 8-26

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-5
- Key Mapping Editor E-2
 - adding mapping E-4
 - binary mapping E-3
 - changing mapping E-5
 - deleting mapping E-5
 - filtering function list E-6
- keyboard E-2
 - mappings E-2
 - navigation E-2
- key-names, location of configuration files 8-33
- keyword, channel 10-8
- KilometersPerHour, updater 13-25
- KnotsPerHour, updater 13-25

L

- label** parameter A-5
- lamppost, prop type 11-24
- lateral-clearance** parameter 8-27
- layer
 - feature
 - adding 11-10
 - adding prop from 11-21
- layout
 - formation
 - automatic 5-23
 - manual 5-23
- leader-promotion-id** parameter 8-15
- level-of-detail 13-27
- lifeform
 - appearance, random 5-14
 - configuring DI-Guy appearance and character 2-8

- limits, terrain 11-3
 - lisp B-2
 - list, props 11-23
 - listing, components in parameters database 7-16
 - load-entry** parameter 8-29
 - load-entry-point** parameter 8-29
 - loading
 - display engine configurationdisplay configurations 10-6
 - MetaFlight 11-15
 - models 11-26
 - Scenario Merge project 4-5
 - terrain 11-26
 - loadPluginIfVersionMismatch** parameter 2-4
 - local
 - object 7-10
 - network interface 7-13
 - local-objects** parameter 7-10, 7-11, B-4
 - location, window 10-6
 - location image attribute 11-8
 - LOD 13-27, 13-29
 - MetaFlight 11-31
 - LOD GEOM 11-31
 - logFrameRateStatistics** parameter 2-4
- ## M
- MÄK Technologies LISP 2-10
 - MÄK Terrain Database Tool C-21
 - map datum, configuring 2-5
 - mapping
 - keys E-2
 - model, to entity type 12-2, 12-3
 - maxDrainInputReads** parameter 2-6
 - maxDrainInputTime** parameter 2-4
 - mcastTtl** parameter 2-6
 - MEDF, creating 12-8
 - MEIF, creating 12-8
 - memory 1-3
 - memory use, improving 12-7
 - menu, adding to Create A-5
 - merging, scenarios 4-2, 4-6
 - message
 - frequency of 2-4
 - information and warning 2-3
 - notification for Scenario Merge 4-7
 - sending and receiving 8-19
 - message-timeout** parameter 9-4
 - metadata, in system definition file 7-18
 - MetaFlight
 - building 11-31
 - configuring 11-15
 - environment variables 11-18
 - geometry offset factor 11-17
 - loading 11-15
 - LOD 11-31
 - performance 11-15, 11-17, 11-18
 - tile 11-15
 - tile-matching 11-15
 - virtual texture 11-31
 - visual quality 11-15
 - migrating, object parameter database to new release 7-28
 - MilesPerHour, updater 13-25
 - Military Symbol Icon Visualizer 13-21
 - minDrainInputTime** parameter 2-6
 - minimum tick period 1-9
 - minimum tick period variance 1-9
 - min-tick-period** parameter 1-9
 - min-tick-period-variance** parameter 1-9
 - mirror image attribute 11-7
 - missile-entity-state-repository 7-12
 - missile-param 7-12
 - missile-param** parameter B-8
 - model
 - adding, tutorial D-2
 - animation F-8
 - communication 2-9
 - compressing 12-8
 - creating 13-29
 - DI-Guy, choosing 12-6
 - editing model mapping 12-4
 - entity 5-4
 - editing 5-3, 5-8
 - file caching 11-26
 - flipping DDS textures 13-26
 - instance, caching 12-7
 - mapping
 - adding 12-3
 - deleting 12-6
 - editing 12-4
 - to FLT files 12-2
 - object 5-4
 - schema 13-9
 - set 13-18
 - sharing resources 12-7
 - simulation 5-4
 - model definition 12-2
 - applying updated 13-12

- model definition 12-2 (continued)
 - copying 13-11
 - creating 13-9, 13-10, D-2
 - cockpit display 13-24
 - DefaultEllipsoidArc 8-20
 - deleting 13-11
 - DI-Guy character 13-10
 - editing 13-12
 - filtering list of 13-12
 - parameter
 - adding 13-12
 - deleting 13-14
 - editing 13-14
 - prop, changing 11-25, C-20
 - saving 13-15
 - schema 13-3
 - SpeedTree 13-10
- Model Definition Editor page 13-9, 13-10, 13-11, 13-12, 13-14, D-2
- model instancing 12-7
 - cache, clearing 12-7
- Model Specification for DIS Interoperability F-2
- monitor, multiple 10-15
- movement model 5-24
 - specifying 5-26
- movement system 5-25, 7-18
- moving object parameter B-8
- moving-object-param 7-12
- moving-object-param** parameter 7-23, 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 10-15
- MultiGen-Paradigm, OpenFlight format F-2
- multiple, back-ends 3-11
- munition, configuring entity for damage by 8-14
- munition-type parameter 8-10
- mutually exclusive task, configuring 8-25
- mutually-exclusive-task-groups** parameter 8-25

N

- name
 - changing, entity 5-11
 - component 7-14
 - FED file 2-6
 - federate 2-6
- navigating, aggregate hierarchy 5-19

- navigation
 - function, editing key map E-3
 - key map E-2
 - keyboard E-2
- near clipping plane 10-10
- net-interface-min-tick-period** parameter 1-10, B-5, B-7
- net-interface-min-tick-period-variance** parameter 1-10, B-5, B-7
- network interface B-5
 - hierarchy 7-13
- network-interface** parameter B-5, B-7
- new, terrain 11-3
- New Scenario dialog box 3-3
- node, LOD GEOM and FILE LOD GEOM 11-31
- notification level 2-3
 - Scenario Merge 4-7

O

- object
 - adding, resources in OPD Editor 6-11
 - adding to terrain C-21
 - configuring, avoidance of 8-27
 - count 1-3
 - editing, multiple 6-6
 - models 5-4
 - parameter B-3
 - default value 3-12
 - subcomponents 7-10
 - super-type 7-7
 - timeout threshold 2-6
 - type 7-6
 - hierarchy 7-8
- object map file 3-11
- object parameter database 5-4, 7-2
 - adding
 - component in OPD Editor 6-8
 - new entity 7-17
 - new object in OPD Editor 6-7
 - preconfigured aggregate 7-23, 7-25
 - soil type parameter 6-12
 - subordinate object 6-11
 - configuring path to 2-3
 - configuring sensors 8-3
 - loading in OPD Editor 6-3
 - migrating to new release 7-28
 - pathname interpretation 7-27
 - reloading 2-4
 - saving 6-14

- object parameter database 5-4, 7-2 (continued)
 - system definition file 7-18
 - Object Parameter Database Editor. *See* OPD Editor
 - object type enumeration B-9
 - objectConsoleNotifyLevel** parameter 2-4
 - Object-Map** parameter 3-5
 - object-type** parameter 7-8, B-3
 - objTypeEnums.h* header file B-9
 - observer
 - altitude, filtering entities by 1-8
 - camera, changing 10-17
 - movement, functions E-2
 - Observer Name 10-8
 - obstacle, configuring avoidance of 8-27
 - obstruction, configuring avoidance of 8-27
 - occupancy-director-controller, role in embarkation 8-29
 - occupancy-director-controller component 8-30
 - OffsetX** parameter 13-25
 - OffsetY** parameter 13-25
 - OffsetZ** parameter 13-25
 - opacity
 - prop 11-24
 - setting 11-24
 - OPD, reloading 2-4
 - OPD Editor 6-2
 - adding
 - new object 6-7
 - resources 6-11
 - adding subordinate object 6-11
 - changing
 - component priority 6-9
 - list views 6-4
 - compared to Entity Editor 5-3
 - connecting components 6-10
 - deleting parameters and components 6-6
 - editing
 - components 6-9
 - multiple objects 6-6
 - preconfigured aggregate 7-24
 - editing parameters 6-4
 - editing systems 7-18
 - loading object parameter database 6-3
 - opening from Entity Editor 5-27
 - restoring values 6-6
 - saving object parameter database 6-14
 - starting 6-3
 - OPD editor, ingress and egress points 8-30
 - OPE file 7-27
 - variable bindings 7-22
 - Open Model Set, dialog box 5-7
 - OpenFlight, format F-2
 - OpenGL 11-27
 - opening, simulation model set 5-7
 - OpenSceneGraph, environment variables for
 - anaglyphic stereo 10-19
 - optimizing, performance 1-3
 - order of battle
 - file 3-11
 - default values in 3-12
 - Order-Of-Battle** parameter 3-5
 - orientation
 - camera 10-17
 - prop, changing 11-25
 - osgEarth 11-11
 - adding terrain servers 11-13
 - outline
 - segment 8-23
 - sensor volume segment 8-23
 - output, port group 7-14
 - out-the-window view 10-15
 - Overlay** parameter 3-5
- ## P
- paged terrain, flipping DDS textures 11-27
 - panel
 - Display Configuration 10-5
 - Frame Rate Monitor 1-4
 - parameter
 - adding, to model definition 13-12
 - additional-ops** 3-6
 - addMulticastAddr** 2-6
 - aggregate-formation-controller** 8-15
 - aggregateSpatialModelTickAsFastAsPossibleWhilePaused** 2-2
 - aggregateSpatialModelTickPeriod** 2-2
 - aggregateSpatialModelTickPeriodUsesRealTime** 2-2
 - aggregateSpatialModelTickVariance** 2-2
 - all-tasks-exclusive** 8-25
 - ammo** 8-10
 - angle-of-incidence** 8-13
 - angleSegmentMax** 8-22
 - appldRange** 2-6
 - appNumber** 2-2
 - assertOnBlockingTerrainCalls** 2-3
 - auto-reorganize** 3-5
 - batchScenarioFileName** 2-3

parameter (continued)

- catastrophic-kill 8-13
- category 7-17, 7-26
- cgfDispatcherReceivePort 2-3
- ChannelKeyword 13-25
- commenting out 2-10
- comm-model-name 8-19
- component list order 7-16
- Component-Attachment 3-5
- Component-Attachment-Table 3-5
- component-manager B-5, B-7
- cultural-feature-param 7-12, B-8
- daemon 2-3
- defaultHeartbeatThreshold 2-6
- defaultObjectTimeout 2-6
- defaultParameterDatabasePath 2-3, 3-7
- defaultTerrainDatabasePath 2-3, 3-7
- definitions 6-4
- deleting, from model definition 13-14
- deleting in OPD Editor 6-6
- 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 1-12
- DtNotifyLevel 2-3
- echelon-level 7-23, 7-26
- editing in Entity Editor 5-12
- editing in OPD Editor 6-4
- egress-points 8-29
- embarkation, editing 8-30
- embarkation-slot-entry 8-29
- embarkation-slots 8-29
- entity-type 8-11, A-5
- entry B-2
- entVrfDataTimeThreshold 2-4
- execName 2-5
- exerciseld 2-6
- exercise-start-time 3-6
- federateName 2-6
- fedFileName 2-6
- Filename 13-24
- fixed-wing-entity-param B-8
- fomMapperInitData 2-5
- fomMapperLib 2-5
- Force-Hostility 3-5
- forceParameterDbReload 2-4

parameter (continued)

- force-type A-4, A-5
- frame-mode 3-6
- frame-time 3-6
- Global-Plan 3-5
- ground-vehicle-param 7-6, B-8
- Gui-Terrain-Database 3-5
- HLA 2-5
- human-param 7-12, B-8
- ignoreAdvisories 2-4
- ingress-points 8-29
- joy-device-type 8-26
- joy-rot-gain 8-26
- joy-rot-invert 8-26
- joy-rot-sensitivity 8-26
- joy-trans-gain 8-26
- joy-trans-invert 8-26
- joy-trans-sensitivity 8-26
- label A-5
- lateral-clearance 8-27
- leader-promotion-id 8-15
- load-entry 8-29
- load-entry-point 8-29
- loadPluginIfVersionMismatch 2-4
- local-objects 7-10, 7-11, B-4
- logFrameRateStatistics 2-4
- maxDrainInputReads 2-6
- maxDrainInputTime 2-4
- mcastTtl 2-6
- message-timeout 9-4
- minDrainInputTime 2-6
- min-tick-period 1-9
- min-tick-period-variance 1-9
- missile-param B-8
- model definition, editing 13-12, 13-14
- moving-object-param 7-23, B-8
- mutually-exclusive-task-groups 8-25
- net-interface-min-tick-period 1-10, B-5, B-7
- net-interface-min-tick-period-variance 1-10, B-5, B-7
- network-interface B-5, B-7
- object B-3
- objectConsoleNotifyLevel 2-4
- Object-Map 3-5
- object-type 7-8, B-3
- OffsetX 13-25
- OffsetY 13-25
- OffsetZ 13-25
- Order-Of-Battle 3-5
- Overlay 3-5

- parameter (continued)
 - parameter-type 7-6, 7-9, 7-27, B-2, B-3
 - Perspective 13-24
 - PerspectiveHeight 13-25
 - PerspectiveWidth 13-25
 - Plan 3-5
 - plan-manager B-5, B-7
 - pluginsDirectory 2-4
 - position-offset 8-15
 - promotion-id 8-15
 - publishSimulationTimeForStealth 2-4
 - publish-transmitter 9-3
 - random-number-seed 3-6
 - range-determinant 8-11, 8-13
 - range-list 8-11
 - recvBufferSize 2-7
 - remote-objects 7-11, B-6
 - rotary-wing-entity-param B-8
 - rprFomVersion 2-6
 - RSOName 13-24
 - runInTimeManagementMode 2-6
 - scenario, file 3-4
 - scenarioFileName 2-4
 - schema
 - adding 13-7
 - deleting 13-8
 - editing 13-8
 - SelectionGroups 3-5
 - sendBackendLogToNetwork 2-4
 - sendBufferSize 2-7
 - sendFedTime 2-6
 - sessionId 2-4
 - set-data-request-type 3-10
 - showLines 8-23
 - simMgmtPduProcessLevel 2-5
 - simulationModelSet 2-5, 5-6
 - Simulation-Model-Set-Files 3-6
 - siteld 2-5
 - startInRunMode 2-5
 - state-repository B-4, B-6
 - state-repository-min-tick-period 1-10
 - state-repository-min-tick-period-variance 1-10
 - statusUpdateHeartbeat 2-5
 - stopping-distance-factor 8-27
 - structure of entries B-2
 - substitute-message-size 9-4
 - surface-entity-param B-8
 - targetFrameRate 2-5
 - task-manager B-5, B-7
 - task-type 3-10
- parameter (continued)
 - Terrain-Database 3-5
 - terrainDatabase 2-5
 - time-multiplier 3-5
 - useAbsoluteTimeStamps 2-6, 2-7
 - use-actual-message-size 9-4
 - useAdvisories 2-6
 - useAsyncIO 2-7
 - useCustomMapDatum 2-5
 - useDIGuy 2-5
 - value, restoring in OPD Editor 6-6
 - vector-obstruction-type 8-28
 - vector-type 8-28
 - visualizer 13-19
 - vrf-base-object-param 7-27, B-3
 - warhead 8-10
- parameters database, component list order 7-16
- parameter-type
 - aggregate 7-23
 - control object 7-27
- parameter-type parameter 7-6, 7-9, 7-27, B-2, B-3
- parent
 - element 13-19
 - element definition 13-15
- Parent Channel 10-8
- pathname
 - interpreting by object parameter database 7-27
 - scenario files 3-6
- pause mode 2-5
- PDU
 - siman, configuring 2-5
- performance 1-6, 1-7
 - asynchronous I/O 1-8
 - compressing texture 11-26
 - factors affecting 1-2
 - frame rate 1-12
 - improving 12-7
 - interest management 1-6
 - intervisibility intersections 1-8
 - MetaFlight 11-15, 11-17, 11-18
 - network, tuning 1-11
 - optimizing 1-2, 1-3
 - SpeedTree profile 13-28
 - statistics, displaying 1-4
 - terrain 11-3
 - MetaFlight 11-31
 - tuning
 - network interface 1-10
 - state repository 1-10
 - tuning visual for 3D application 1-11

- Performance Options dialog box 1-11, 1-12
- Performance Options page 1-11
- perspective, in display 10-13
- Perspective** parameter 13-24
- PerspectiveHeight** parameter 13-25
- PerspectiveWidth** parameter 13-25
- Pitch, updater 13-25
- pixel 13-29
- 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 10-18
 - configuring 10-19
- polygon 13-29
 - count 11-3
- port 7-14
 - for scenario files 2-3
 - group 7-14
- port number 2-6
- position
 - camera 10-17
 - prop, changing 11-25
 - window attribute 10-6
- position-offset** parameter 8-15
- precedence, components in parameters database 7-16
- priority
 - component, changing 6-9
- probability table, direct and collateral damage 8-13
- probability-coefficient list 8-11
- profile, SpeedTree performance 13-28
- project
 - adding scenarios to 4-3
 - creating Scenario Merge 4-3
 - file 4-5
 - Scenario Merge 4-3
 - loading Scenario Merge 4-5
 - removing scenarios from 4-4
 - saving Scenario Merge 4-5
 - specifying ignore list for 4-4
- Projection Resize Policy 10-8
 - configuring 10-11
- Projection Units 10-8
- promotion-id** parameter 8-15
- prop

- prop (continued)
 - adding
 - from feature layer 11-21
 - to terrain 11-19, C-21
 - extracting 11-19, C-17
 - model definition, changing 11-25, C-20
 - opacity 11-24
 - opaque 11-24
 - transparent 11-24
 - type 11-24
 - viewing list of 11-23
- properties, editing system 5-27
- property
 - channel 10-8
 - window 10-6
- publishing, simulation time 2-4
- publishSimulationTimeForStealth** parameter 2-4
- publish-transmitter** parameter 9-3
- pulse repetition rate 8-20

Q

- QualNet
 - example scenario 9-6
 - using with VR-Forces 9-5
- Qualnet, scenario 9-6

R

- radio
 - configuring 8-19
 - message, configuring 2-9
- radio transmitter, enabling 9-3
- radius, sensor volume 8-21
- random, DI-Guy appearance 5-14
- random-number-seed** parameter 3-6
- range-determinant** parameter 8-11, 8-13
- range-list** parameter 8-11
- raster map
 - display order 11-9
 - displaying 11-6
- receive buffer size 2-7
- receiving, radio messages 8-19
- recvBufferSize** parameter 2-7
- reference ellipsoid, configuring 2-5
- region
 - DDM 1-7
 - update threshold 1-7
- relative path 3-6

- remote
 - object 7-10, B-6
 - network interface 7-13
 - remote attachment, key-names 8-33
 - remote entity, attaching components 8-32
 - remoteAttachment example scenario 8-32
 - remoteConfigurations directory 8-33
 - remote-configurations entry 8-33
 - remote-objects** parameter 7-11, B-6
 - removing
 - category 5-34, 5-35
 - channel 10-6
 - force 5-38
 - formation 5-20
 - scenarios from Scenario Merge project 4-4
 - subordinate 5-18
 - system 5-27
 - window 10-5
 - rename, element definition 13-17
 - renaming
 - category 5-34, 5-35
 - entity definition D-6
 - formation 5-20
 - terrain server 11-14, C-13
 - Render Settings page 1-5, 10-11
 - reorganizing
 - Create menu A-4
 - element definitions 13-20
 - repeat image attribute 11-7
 - replace image attribute 11-7
 - replacing, subordinate 5-18
 - repository B-4
 - resizeable window 10-2
 - resizing, windows 10-11
 - resolution, tree 13-27
 - resource, adding to object 6-11
 - restoring, parameter values in OPD Editor 6-6
 - reusable software object 13-23
 - reverting
 - category changes 5-35
 - Create menu 5-34
 - Roll, updater 13-26
 - rotary-wing-entity-param 7-12
 - rotary-wing-entity-param** parameter B-8
 - rotary-wing-entity-state-repository 7-12
 - RPR FOM 12-2
 - specifying version 2-6
 - rprFomVersion** parameter 2-6
 - RSO 13-23
 - RSOName** parameter 13-24
 - run mode 2-5
 - runInTimeManagementMode** parameter 2-6
- ## S
- saving
 - display configuration 10-6
 - element definition 13-20
 - model definition 13-15
 - object parameter database 6-14
 - Scenario Merge project 4-5
 - schemas 13-3
 - terrain 11-4
 - scenario
 - adding to Scenario Merge project 4-3
 - closing in Scenario Merge 4-7
 - displaying scenario file in Scenario Merge 4-6
 - editing 4-7
 - file
 - editing 3-3
 - parameters 3-4
 - pathnames 3-6
 - specifying which to use 2-4
 - merging 4-2, 4-6
 - parameter, **version** 3-6
 - Qualnet 9-6
 - remoteAttachment 8-32
 - removing from Scenario Merge project 4-4
 - temporary directory 3-12
 - terrain database 3-8
 - Scenario Merge 4-2
 - closing scenario file 4-7
 - command line syntax 4-8
 - console output 4-7
 - creating project 4-3
 - displaying scenario file 4-6
 - editing scenario file 4-7
 - ignore list 4-4
 - loading project 4-5
 - project file 4-3, 4-5
 - saving project 4-5
 - scenario merge, API 4-2
 - scenarioFileName** parameter 2-4
 - schema 13-3
 - copying 13-6
 - creating 13-5
 - deleting 13-6
 - model definition 13-3
 - parameter
 - adding 13-7

- schema 13-3 (continued)
 - parameter
 - deleting 13-8
 - editing 13-8
 - saving 13-3
 - visual definition 13-3
- Schema Editor page 13-5, 13-6, 13-7, 13-8
- Screen Number attribute 10-6
- segment
 - angle, setting 8-22
 - arc size 8-22
 - edge 8-23
 - outline, displaying 8-23
 - sensor volume 8-22
- 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 8-19
- sensor 8-3
 - adding new 8-5
 - attaching to remote entity 8-32
 - configuring 8-3
 - connecting 8-3
 - signature 8-5
 - signature-object-sensor 8-3
 - system definition file, editing 8-4
- sensor signature 5-25
- sensor system 5-24
- sensor volume
 - color 8-20
 - configuring 8-20
 - outlines 8-23
 - pulse rate 8-20
 - radius 8-21
 - segment angle 8-22
 - segment size 8-22
- sensors 7-18
- session ID, configuring 2-4
- sessionId** parameter 2-4
- set statement 3-9
- set-data-request-type** parameter 3-10
- setting
 - communication model 9-3
 - prop opacity 11-24
 - segment arc size 8-22
 - sensor volume radius 8-21
- sharing, resources 12-7
- Show Performance Stats Overlay check box 1-6
- showHuds 10-8
- showLines** parameter 8-23
- signature, sensor 8-5
- signature-object-sensor 8-3
- 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 5-4
 - creating 5-5
 - deleting entity 5-29
 - file 5-4
 - new, creating 5-6
 - opening 5-7
- simulation time, publishing 2-4
- simulationModelSet** parameter 2-5, 5-6
- Simulation-Model-Set-Files** parameter 3-6
- site ID 3-11
 - configuring 2-5
- siteld** parameter 2-5
- size
 - region 1-7
 - window 10-6
- slider
 - visual quality, configuring 1-12
- slot, embarkation 8-29
- snap to grid, formation 5-24
- soil type, adding to object 6-12
- specifying
 - damage model 5-26
 - default *default.sms* 5-6
 - frame length 3-6
 - frame rate mode 3-6
 - movement model 5-26
 - segment angle 8-22
- specifying default 5-6
- SpeedTree
 - adding to terrain C-21
 - configuring 13-27
 - model definition 13-10
 - performance profile 13-28
- SpeedTree Settings page 13-27, 13-28
- spot report, affect on performance 1-8
- starting
 - Entity Editor 5-4
 - OPD Editor 6-3

- starting (continued)
 - Scenario Merge tool 4-2
 - startInRunMode** parameter 2-5
 - state repository B-4
 - hierarchy 7-11, 7-12
 - statement
 - conditional 3-9
 - set 3-9
 - state-repository** parameter B-4, B-6
 - state-repository-min-tick-period** parameter 1-10
 - state-repository-min-tick-period-variance** parameter 1-10
 - statistics, displaying 1-4
 - statusUpdateHeartbeat** parameter 2-5
 - Stealth, simulation clock 2-4
 - stereo display, types of 10-18
 - stopping-distance-factor** parameter 8-27
 - subcomponent B-4
 - classes 7-10
 - subordinate
 - adding to aggregate 5-17
 - editing 5-19
 - order, changing 5-18
 - removing from aggregate 5-18
 - replacing 5-18
 - subordinate object, adding 6-11
 - substitute-message-size** parameter 9-4
 - super-type 7-7
 - surface, damage specification 8-13
 - surface-entity-param 7-12
 - surface-entity-param** parameter B-8
 - surface-entity-state-repository 7-12
 - switch bead F-5
 - synchronizing, VR-Forces and Stealth simulation
 - time 2-4
 - sysdef file 7-18
 - system 7-18
 - adding to entity 5-26
 - connection 7-19
 - editing 5-27, 6-13
 - editing properties 5-27
 - entity 5-24
 - removing 5-27
 - sensor 8-3
 - system definition file 7-18
 - configuring connections 7-19
 - creating 7-19
 - sensor 8-4
 - variable bindings 7-22
 - system-definition block 7-20
 - Systems List, OPD Editor 6-4
 - Systems List tab 6-13
- ## T
- tactical graphics 12-2
 - element definition 13-15
 - targetFrameRate** parameter 2-5
 - target-type parameter 8-10
 - task, configuring mutually exclusive 8-25
 - task-manager** parameter B-5, B-7
 - task-type** parameter 3-10
 - TDB Tool C-21
 - temporary directory, scenario 3-12
 - terrain
 - building 11-4
 - creating 11-3
 - deleting cached 11-27
 - displaying raster image on 11-6
 - DTED support 11-3
 - file caching 11-26
 - MetaFlight
 - building 11-31
 - loading 11-15
 - paged, flipping DDS textures 11-27
 - paging, MetaFlight 11-15
 - patch
 - adding to terrain 11-4
 - extracting props from 11-19
 - prop
 - adding 11-19
 - extracting C-17
 - list 11-23
 - saving 11-4
 - server 11-11
 - adding 11-13
 - editing 11-14
 - editing configuration 11-14
 - 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 11-14, C-13
 - Terrain-Database** parameter 3-5
 - terrainDatabase** parameter 2-5

- texture 13-29
 - compression, enabling and disabling 11-26
 - coordinate system for image 11-8
 - DDS 11-27
 - flipping, paged terrain 11-27
 - sharing 12-7
- threshold
 - data updates 2-4
 - DIS heartbeat 2-6
 - object timeout 2-6
- tick
 - rate, tuning 1-9
- tile, MetaFlight 11-15
- tile-matching 11-15
- Time Multiplier toolbar 3-5
- time stamp order, configuring 2-6
- time-multiplier** parameter 3-5
- timeout, object 2-6
- trailing effects 12-2
- transparency 13-29
 - prop, setting 11-24
- tree
 - prop type 11-24
 - SpeedTree 13-27
- tuning
 - components 1-9
 - network performance 1-11
 - performance
 - component 1-9
 - network interface 1-10
 - state repository 1-10
 - visual performance 1-11
- tutorial
 - adding feature as a prop C-21
 - adding model D-2
 - extract props C-17
 - terrain creation C-2
- type, prop 11-24

U

- undoing
 - category changes 5-35
 - changes to Create menu 5-34
 - entity edits in Entity Editor 5-14
- update, frequency of 2-4
- updater 13-23
 - AOA 13-25
 - attribute 13-23
 - FeetAGL 13-25

- updater 13-23 (continued)
 - FeetMSL 13-25
 - FeetPerMinute 13-25
 - Headlight 13-26
 - KilometersPerHour 13-25
 - KnotsPerHour 13-25
 - MilesPerHour 13-25
 - Pitch 13-25
 - Roll 13-26
 - Value 13-26
 - Yaw 13-26
- Use Extracted Geometry 11-19
- useAbsoluteTimeStamps** parameter 2-6, 2-7
- use-actual-message-size** parameter 9-4
- useAdvisories** parameter 2-6
- useAsyncIO** parameter 2-7
- useCustomMapDatum** parameter 2-5
- USE-DEFAULT keyword 3-12
- useDIGuy** parameter 2-5
- user-defined, formation 8-16
- UV, coordinates 11-8

V

- Value, updater 13-26
- variable
 - bindings 7-22
 - tick rate 1-9
- variable frame mode 1-12
- variable-frame 3-6
- Variable-Frame Run-To-Complete 3-6
- vector
 - feature, configuring avoidance of 8-28
- vector-obstruction-type** parameter 8-28
- vector-type** parameter 8-28
- version, RPR FOM 2-6
- version** parameter, scenario file 3-6
- video card, processor speed 1-3
- viewing, list of props 11-23
- Viewport 10-8
- viewport, changing 10-14
- virtual texture
 - compressing 11-18
 - MetaFlight 11-31
- visual definition, schema 13-3
- Visual Model Editors dialog box 13-9
- visual quality, MetaFlight terrain 11-15
- visual quality slider, configuring 1-12
- visualizer 13-3, 13-15
 - adding 13-18

- visualizer 13-3, 13-15 (continued)
 - editing 13-18, 13-19
- vrf A-2
- vrf-base-object-param** parameter 7-27, 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 7-12
- vrf-object-param 7-12
- vrf-object-param parameter type B-7
- vrf-object-state-repository 7-12
- vrf-overlay-object-state-repository 7-12
- vrfSim, running as daemon 2-3
- vrfSim.mtl* file 2-2, 5-6
- vrfSim.opd, remote-configurations block 8-33
- vrfSim.opd* file 7-2
- vrfSimSettings.xml* file 1-11

W

- warhead** parameter 8-10
- warning message 2-3
- weapon system 5-24, 5-25
- weapons 7-18
- wildcard 7-8
- window
 - adding 10-3
 - adding channel, adding 10-5
 - attributes, editing 10-6
 - location, changing 10-6
 - property 10-6
 - removing 10-5
 - size, changing 10-6
 - types 10-2
- Window Name attribute 10-6
- Window Type attribute 10-6
- WRM Entity Model Specification F-1, F-6

X-Y-Z

- Yaw, updater 13-26
- Z Far 10-8, 10-10
- Z Near 10-8, 10-10



Link - Simulate - Visualize

VRF-4.0-7-110117