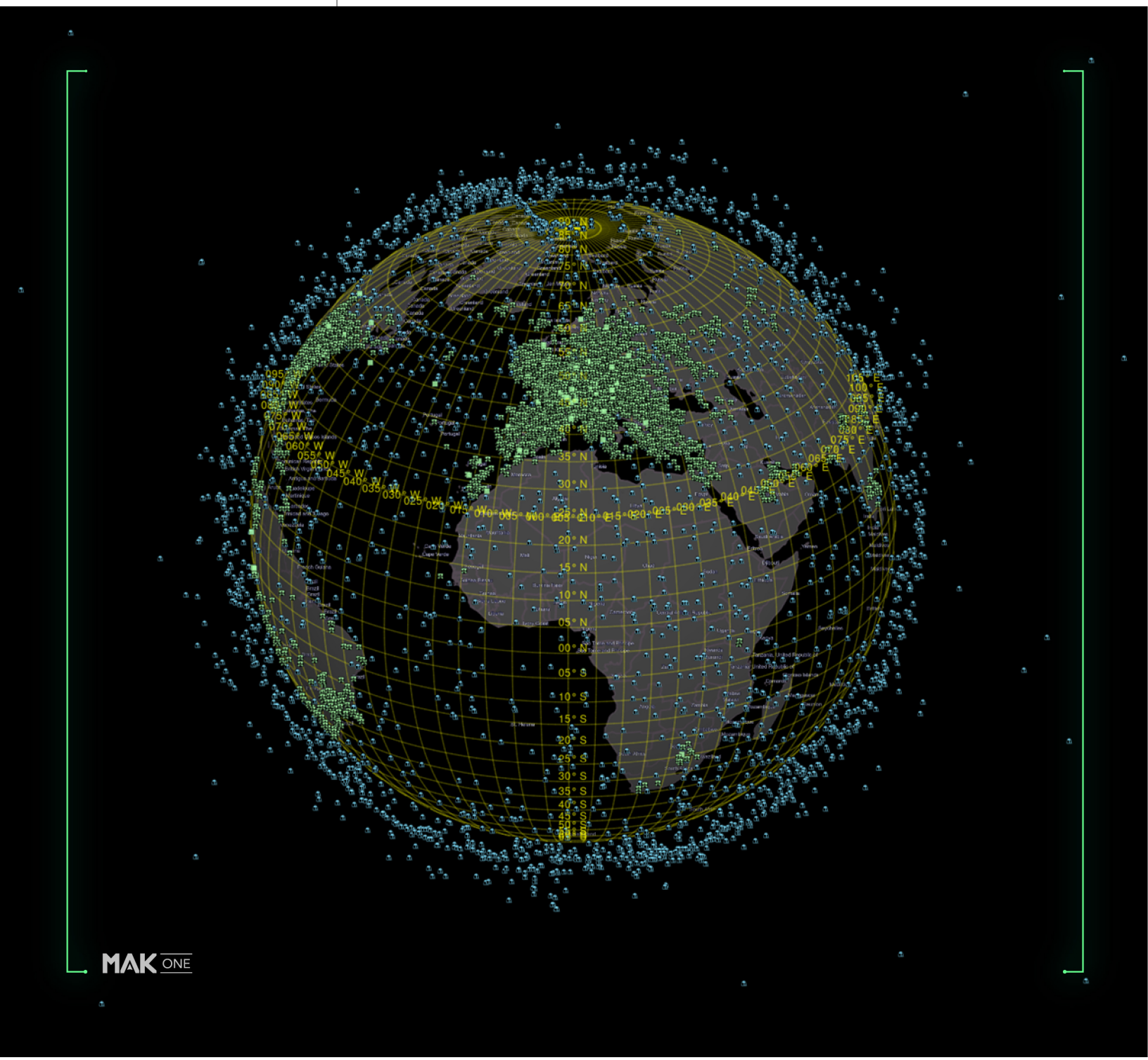
MAK TECHNOLOGIES

Adding Content to MAK ONE
Applications Reference Manual



VR-Vantage 3.0.3
VR-Forces 5.0.3
VR-Engage 2.0.3

ST Engineering
North America

Copyright © 2023 MAK Technologies
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 MAK Technologies.

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

GL Studio® is a registered trademark of The DiSTI® Corporation.

Portions of this software utilize SpeedTree® technology (©2008 Interactive Data Visualization, Inc.). SpeedTree is a registered trademark of Interactive Data Visualization, Inc. All rights reserved.

SilverLining™ is a trademark of Sundog Software.

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

3ds Max® is a registered trademark of Autodesk, Inc.

All other trademarks are owned by their respective companies.

For third-party license information, see "Third-Party Licenses" on page 21 in the Preface.

MAK Technologies
10 Fawcett Street, Suite 204
Cambridge, MA 02138 USA

Voice: +1 617.876.8085
Fax: +1 617.876.9208

info@mak.com

www.mak.com

Revision MAK-ART-1-230418



Contents

PREFACE

How This Manual is Organized 16
MAK Products 17
How to Contact Us 19
Telephone 19
E-mail 19
Internet 20
Post 20
Document Conventions 20
Mouse Button Naming Conventions 21
Third-Party Licenses 21
Third-Party Libraries Supported 21
Library Licensing 22
Boost License 24
libXML and libICONV 25
Third-Party Licenses for VR-Vantage Applications 25

I. INTRODUCTION

Adding Content to MAK ONE Applications 28
Screen Captures 28

II. TERRAIN

Introduction 30
Terrain Databases Provided with VR-Vantage Applications 30

1. Composing Terrains

1.1. Creating a Composed Terrain 36
1.1.1 Saving a Terrain 36
1.2. Adding Elevation Data to a Terrain 37
1.3. Adding Images to a Terrain 39
1.4. Feature Layers 39
1.4.1 Adding a Feature Layer 40
1.4.2 Removing a Feature Layer 41
1.5. Adding a Dynamic Ocean Layer 41
1.5.1 Extracting Water Textures and Adding an Ocean Layer 42
1.6. Adding a Dynamic Ocean Layer Directly 44
1.7. Setting the Ocean LOD Elevation 45
1.8. Configuring the Ocean Height Map 46
1.8.1 Setting the Ocean Height Map Resolution 46
1.8.2 Setting the Ocean Height Map Texture Size 47
1.8.3 Setting the Ocean Height Map Regeneration Threshold 47
1.9. Specifying Simulation Ocean Settings 48
1.10. Adding Props to a Terrain 49
1.10.1 Extracting Props from a Terrain Patch 49
1.10.2 Adding Props from a Feature Layer 51
1.10.3 Viewing a List of Props 53
1.10.4 Selecting Props 53
1.10.5 Changing a Prop's Type 53
1.10.6 Changing a Prop's Position or Orientation 54
1.10.7 Changing a Prop's Model Definition 54
1.10.8 Exporting Props and SpeedTrees to a Shapefile 54
1.10.9 Importing Props and SpeedTrees from Shapefiles 55

2. Paged and Streaming Terrains

2.1. Connecting to Terrain Servers 58
2.1.1 Adding Terrain Server Connections 60
2.1.2 Connecting to Terrain Servers through a Proxy 60
2.2. Loading MetaFlight Terrains 60
2.3. Preprocessing Paged Terrains 61
2.4. Manually Paging-In Terrain in VR-Forces 61

3. Loading Data Using earth Files

3.1. Earth Files 65
3.2. A Simple earth File 65
3.2.1 Including External Files in earth Files 67

3.3. Supported Drivers 68
3.3.1 Disabling Intersections for a Feature Geometry Layer 69
3.4. Loading Multiple DTED Files 69
3.5. Processing Streamed Data 69
3.6. Adding Feature Layers to an earth File 70
3.6.1 Configuring Point Features 72
3.6.2 Configuring Linear Features 73
3.6.3 Configuring Areal Features 74
3.6.4 Configuring Altitude Features 75
3.7. Extruded Buildings 75
3.8. Runway Offset to Avoid Z-Fighting 76
3.9. Specular Reflections on Water 77
3.10. Using Cut-in Sites for High Resolution Insets 77
3.10.1 Using the Boundary Generation Tool 78
3.11. Procedural Terrains 79
3.11.1 Exporting the Location of Procedural Vegetation 81
3.11.2 Configuring Procedural Trees for VR-Forces 82
3.12. Loading 3D Tiles 85
3.13. Options for earth Files 85
3.14. Referencing an Earth Gravitational Model (EGM) 86
3.15. Changing Visible Layers Based on the Date 86
3.16. Specifying Fading in earth Files 87
3.17. Differential Processing of Elements and Attributes 87
3.17.1 Differential Processing of model Elements 88
3.18. Indexing Elevation Layers 89
3.18.1 Configuring Elevation Layers for Indexing 89
3.18.2 Generating the Elevation Layer Indexes 90
3.19. Streaming Buoys and Beacons 90
3.19.1 Specifying Model Definitions for Streamed Buoys 92
3.19.2 Modeling Buoys Using Texture Atlases 94
3.19.3 Configuring Streaming Beacons 95
3.19.4 Configuring Streamed Navigation Lights 96
3.19.5 Configuring Seasonal Buoys and Beacons 97
3.19.6 Streaming Daymarks 97
3.19.7 Streaming Topmarks 98
3.19.8 Generating Signal Sequences 98
3.20. Land Use Data 98

4. Editing Terrains Using the Terrain Editor

4.1. Introduction to the Terrain Editor 100

4.1.1 Opening the Terrain Editor 100
4.2. Adding Data to a Terrain 101
4.3. Editing Terrain Properties in the Terrain Editor 101
4.3.1 Enabling and Disabling Layers 101
4.3.2 Editing the Properties of a Layer 102

5. Configuring CDB Terrains

5.1. Loading CDB Databases 104
5.2. Configuring CDB earth Files 104
5.2.1 Configuring the Elevation Dataset (001) 104
5.2.2 Configuring the Imagery Dataset (004) 106
5.2.3 Configuring the Raster Material Dataset (005-006) 108
5.2.4 Configuring the GeoSpecific (GS) Dataset (100) 110
5.2.5 Configuring the GeoTypical (GT) Dataset (101) 116
5.2.6 Configuring the Extended Imagery Dataset (900) 118
5.3. CDB Configuration Files 120
5.4. Disabling Intersections in Model Layers 121
5.5. Caching CDB Data (cdbTool) 121

6. Processing MetaFlight Files

6.1. Building Efficient MetaFlight Terrains 124
6.2. Processing MetaFlight Files for Use in MAK ONE Applications 124
6.3. Splitting Datasets into Individual MetaFlight Files 125
6.4. Converting Virtual Texture Datasets to Tiled Textures 125
6.4.1 The mftTool 125
6.4.2 Using the mftTool to Process MetaFlight Files 127
6.4.3 Convert Geometry Grid Datasets with Tiled Textures 128
6.5. Convert Source Data into MEDF Format 128
6.6. Create an MTF File for the MetaFlight Terrain 129

7. 2D View Layers

7.1. 2D View Layers Overview 132
7.2. Defining 2D View Layers in earth Files 132

8. Dynamic Terrain in VR-Forces

8.1. Introduction to Dynamic Terrain 136
8.2. Manually Changing the Terrain 136
8.3. Resetting Dynamic Terrain Changes 137
8.4. Removing Dynamic Terrain Damage 137

8.5. Resetting All Changes in Dynamic Terrain 137
8.6. Creating Dynamic Terrain Areas 138
8.7. Dynamic Terrain Switch Types 138
8.8. Craters 138

9. Configuring Feature Data in VR-Forces

9.1. Introduction 140
9.2. Specifying Feature Names for the Front-End and Back-End 140
9.3. Named Queries 141
9.4. Feature Configuration Attributes 141
9.5. Attribute Transformations 142
9.5.1 Transformations as Aliases 142
9.5.2 Transformations as Derived Attributes 142
9.5.3 Transformations as New Attributes 142
9.5.4 Writing Multiple Transforms to Account for Possible Failures 143
9.6. Source-Specific Configuration 143
9.6.1 Loading Source-Specific Configuration Files 143
9.6.2 Redefining Queries 143
9.7. Extracting Feature Data from GDB to Shapefiles 144
9.8. Converting S-57 Feature Data to Shapefiles 144
9.9. Configuring Aggregate-Level Movement Restrictions 145
9.10. Dynamic Features 146

10. Configuring Airport Settings in VR-Forces

10.1. Overview of Airports 150
10.2. Airport Settings Panel 150
10.2.1 Centering on an Airport 150
10.2.2 Setting Aerodrome Lights 150
10.2.3 Setting Runway Conditions 152
10.3. Setting Runway Lights in a Plan 152

11. Terrain Performance and Configuration

11.1. Configuring File Caching 154
11.1.1 Specifying the Location to Cache Terrain Server Data 155
11.1.2 Caching earth File Terrain Data 155
11.1.3 Enabling Texture Compression 156
11.1.4 Clearing the File Cache 157
11.2. Configuring Paged and Streaming Terrains 157
11.3. Debugging earth Files 157
11.3.1 Viewing a Terrain in Debug Mode 158

11.4. Using Shader-based Effect Maps 159
11.4.1 Composite Effects Maps 161
11.4.2 Debugging Shaders 162
11.4.3 Reloading Shaders 163
11.5. Indirect Rendering of Terrain 164
11.5.1 Scaling the LOD Distance for Indirect Rendering 164
11.5.2 Setting the LOD Transition Percentage 165
11.5.3 Disabling Indirect Rendering for Terrain 165
11.6. Specifying the Maximum Number of Paged LODs 166
11.7. Indirect Rendering for osgEarth Layers 166
11.8. Displaying Terrain Intersection Lines 166
11.9. Displaying DDS Textures Correctly 167
11.9.1 Flipping DDS Textures Globally 168
11.10. Enabling Occlusion Queries 168
11.11. Converting OpenFlight File Texture Data 169
11.12. Drawing Bounding Volumes for Scene Graph Nodes 170
11.13. Masking the Ocean 171
11.14. Debugging the VR-Forces Simulation Terrain 172
11.14.1 Starting the Simulation Terrain Debugging Tool 172
11.14.2 Refreshing the Debug Display 173
11.14.3 Viewing Terrain Skin, Normals, and Collision Objects 173
11.14.4 Viewing Simulation Terrain Features 176
11.14.5 Displaying the NavMesh 178
11.14.6 Displaying the Query Area 179
11.14.7 Displaying Elevation Tiles 180
11.14.8 Displaying Recent Intersections 181
11.14.9 Displaying Page-in and Page-out Nodes 182
11.14.10 Enabling Auto Refresh 183
11.14.11 Simulation Terrain Debugging from the Back-End Console 183

12. Terrain Tutorials

12.1. Creating an earth File to Load Terrain Data 186
12.1.1 Add a Map Element 186
12.1.2 Add Elevation Data to the earth File 186
12.1.3 Add Imagery to the earth File 186
12.1.4 Add Feature Data to the earth File 187
12.1.5 Add Options to the earth File 189
12.1.6 Save the earth File 189
12.1.7 Load the earth File and Save it as an MTF File 190
12.2. Extracting Props from a Terrain 191

12.2.1 Create a New Terrain and Add a Terrain Patch 191
12.2.2 Extract the Props 192
12.2.3 Change the Model Definition for a Prop 194

III. VISUAL MODELS IN MAK ONE APPLICATIONS

13. Introduction to Visual Models

13.1. Introduction to Object Modeling 200
13.1.1 Model Sets 202
13.2. Model Formats for 3D Objects, and Effects 202
13.3. OpenFlight and FBX Models 203
13.4. Managing Model and Element Definitions 203
13.5. Visual Model Composability 204
13.6. Indirect Rendering for Entities 208

14. Schemas

14.1. Creating and Editing Schemas 210
14.2. Creating Schemas 211
14.3. Saving Schemas 212
14.3.1 Exporting Selected Schemas 212
14.4. Deleting Schemas 213
14.5. Copying Schemas 213
14.6. Adding a Parameter to a Schema 214
14.7. Editing a Schema Parameter 215
14.8. Deleting a Parameter from a Schema 215

15. Model Definitions

15.1. Creating and Editing Model Definitions 218
15.2. Work Flow Issues for Model Definitions 219
15.3. Model Definition Files 220
15.4. Creating a Model Definition 221
15.5. Deleting a Model Definition 222
15.6. Copying a Model Definition 222
15.7. Filtering the List of Model Definitions 223
15.8. Editing a Model Definition 223
15.8.1 Adding Parameters to a Model Definition 224
15.8.2 Editing a Model Definition Parameter 225
15.8.3 Deleting Parameters from a Model Definition 225
15.9. Saving Model Definitions 226
15.9.1 Exporting Selected Model Definitions 226

15.10. Importing Model Definitions 226
15.10.1 Importing Model Definitions from the Command Line 227

16. Element Definitions

16.1. Creating and Editing Element Definitions 230
16.2. How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions Are Organized 230
16.2.1 Leaf File Descriptions 231
16.3. Work Flow Issues for Creating and Editing Element Definitions 232
16.4. Creating an Element Definition 233
16.5. Filtering the List of Element Definitions 234
16.6. Editing an Element Definition 234
16.6.1 Adding a Visual Definition 234
16.6.2 Editing a Visual Definition 235
16.6.3 Adding an Attribute to a Visual Definition 237
16.6.4 Deleting an Attribute from a Visual Definition 239
16.6.5 Copying and Pasting Visual Definitions 239
16.7. Jumping to a Model Definition 240
16.8. Changing an Element Definition's File Location 240
16.9. Deleting an Element Definition 241
16.10. Saving Element Definitions 241
16.10.1 Saving Entity, Unit, Spot Report, or Tactical Graphics Element Definitions 242
16.10.2 Saving Interaction and Smoke Element Definitions to .elem Files 243
16.11. Importing Element Definitions 243
16.11.1 Importing a Hierarchy File 244
16.11.2 Importing and Merging Element Definitions 244
16.11.3 Importing Element Definitions from the Command Line 244

17. Mapping Models and Effects

17.1. Introduction to Object Type Mapping 248
17.2. Work Flow Issues for Object Type Mappings 248
17.3. Adding an Object Type Mapping 249
17.4. Editing an Object Type Mapping 250
17.5. Filtering the Element Definition List 251
17.6. Changing an Object Type Mapping's File Location 252
17.7. Jumping to an Element Definition 252
17.8. Deleting an Object Type Mapping 253
17.9. Saving Object Type Mappings 253
17.9.1 Exporting Selected Object Type Mappings 253
17.10. Importing Object Type Mappings 254

17.11. Starting Without Default Object Type Mappings 254

18. Wakes, Cockpits, 2D Icons, and Other Models

18.1. Configuring 2D Icons 257
18.1.1 Editing Font-Based 2D Icons 257
18.1.2 Using Images for 2D Icons 258
18.1.3 Adding Images for Simulation Object Icons 263
18.2. Adding New Cockpit Display Models 264
18.2.1 Installing a Cockpit DLL 265
18.2.2 Creating a Model Definition for a Cockpit 265
18.3. Adding Object Models as Props 268
18.3.1 Adding a Category to the Props Palette 268
18.4. Configuring Light Points 268
18.4.1 Configuring Light Point Type Models 270
18.4.2 Configuring Light Point Group Models 270
18.4.3 Adding a New Light Point Type 271
18.5. Configuring a Light Source for a Specific Sensor Mode 271
18.6. Configuring Wakes in the Visual Model Editor 271
18.6.1 Configuring Tidal Stream Wakes 274
18.6.2 Displaying Propeller Wash 274
18.7. Configuring Persistent Tracks in Model Definitions 274
18.8. Configuring Floating Lifeforms 276
18.9. Adding Wind-based Controls to Models 276
18.10. Using Texture Atlases to Skin Models 278
18.10.1 Configuring Texture Mapping in the Model Definition 278
18.10.2 Metafiles for Texture Mapping 279
18.11. Adding Dynamically Changeable Markings to Models 280
18.11.1 Adding @dis annotations to the Model 281
18.11.2 The Metafile for Markings 282
18.11.3 The Texture Mapping File 284
18.12. Flipping DDS Textures for a Model 284
18.13. Configuring SpeedTree Trees 285
18.13.1 Randomizing the Size of SpeedTrees 286
18.13.2 Displaying SpeedTrees in Planar Reflections 288
18.14. Best Practices for Creating Models 288
18.15. Disabling Indirect Rendering for an OpenFlight Model 289
18.16. Compressing Model Files 289

19. The WRM Specification (DIS Notes)

19.1. Introduction 294

19.2. The OpenFlight File Format 294
19.2.1 Node Name and Comment Fields 294
19.2.2 External References and Instancing 295
19.3. Mapping FBX Model Nodes to DIS Notes 295
19.3.1 Specifying the FBX Texture Path 296
19.4. Modeling for DIS Interoperability 296
19.4.1 The DIS Attribute Lexicon (DAL) 296
19.4.2 DAL Keywords 297
19.4.3 Model Coordinate Systems 297
19.5. Articulated Parts 300
19.6. Entity Appearances 300
19.7. Animation 303
19.8. DIS Keyword Values 303

20. The MAK FBX Specification

20.1. Introduction 330
20.1.1 Meshes (Geometry Objects), Nulls (Dummies, Groups or Locators) 330
20.1.2 Node Name and Comment Fields 330
20.2. Modeling for Simulation Interoperability 330
20.3. The Simulation Attribute Lexicon (SAL) 331
20.3.1 SAL Syntax 331
20.3.2 SAL Keywords and Values 331
20.4. Model Coordinate System 338
20.4.1 Orientation of the Model Coordinate System 338
20.4.2 Origin of the Model Coordinate System 339
20.5. Modeling Best Practices for Efficient Mapping 340
20.5.1 Comment the Parent Model Header 340
20.5.2 Attached Parts 341
20.5.3 Articulated Parts 343
20.5.4 Entity Appearance 345
20.5.5 Animation Sequences 346
20.6. Tagging the Location of Weapon Fire Special Effects 348
20.7. Encoding Simulation Details for Non-Entity Dynamic Objects 349
20.7.1 Specific Appearance Keyword Values used by MAK 349
20.7.2 Specific Articulation Keyword Values used by MAK 366
20.7.3 Shader-based Effects Maps in FBX 371
20.7.4 Using Texture Atlases, Metafiles, and Markings 371
20.7.5 Using Material-Classified (Sensor) Textures 371
20.8. Encoding Damage to Non-Entity Dynamic Objects 372
20.9. Mapping FBX Model Nodes to DIS Notes 372

20.9.1 Specifying the FBX Texture Path 373

21. Special Effects (Particle Systems)

21.1. Particle Systems 376
21.2. Adding a New Particle System 376
21.3. Editing Particle Systems 377
21.3.1 Editing the Properties of Keyframes 378
21.4. Creating a Model Definition for a Particle System 379
21.5. Saving and Reverting Particle Systems 379
21.6. Previewing a Particle System 380
21.6.1 Viewing a Particle System With or Without Textures 382
21.6.2 Viewing a Particle System's Wireframes 384
21.6.3 Viewing Debug Visuals 385
21.7. Particle System Properties 387
21.7.1 Preview Parameters for Particle Systems 392
21.8. Particle System Collision Objects 393

IV. DI-GUY

22. DI-Guy Content

22.1. Customizing DI-Guy Content 398
22.1.1 Encrypt Your Final Customized Geometry 398
22.1.2 Compressing Geometry Files 398
22.1.3 Where to Place Customized Files 398
22.1.4 The DI-Guy Data Directory Structure 399
22.1.5 Understanding DI-Guy Configuration Files 399
22.1.6 Autoload Configuration Files 402
22.2. Customizing Motion 402
22.3. Customizing Texture 402
22.4. Customizing Sound 402

23. Using the Character Viewer

23.1. Introduction to the DI-Guy Character Viewer 404
23.2. Viewing Characters, Appearances, Gestures, and Actions 405
23.2.1 Filtering the Character Type and Appearance Lists 406
23.2.2 Changing the Viewing Angle for a Character 406
23.2.3 Changing the Quality of the Character 407
23.2.4 Demonstrating a Character's Aim and Locomotion Capability 408
23.2.5 Viewing Different LOD Levels 408
23.2.6 Changing How a Character Is Rendered 409

23.2.7 Viewing Multiple Appearances at One Time (Army Mode) 411
23.2.8 Advanced Character Editing 413
23.2.9 Reloading Characters that Have Been Edited 414
23.2.10 Controlling Character Movement 415
23.3. Improving Performance 415
23.3.1 Displaying the Performance Profiler 416
23.4. Running a Lua Script 416
23.5. Viewing Character Geometry Details 416
23.6. The Appearance Editor 417
23.6.1 Visualizing Character Structure 419
23.6.2 Editing Configuration File Text 419
23.6.3 Editing Appearance Effects 419
23.7. Using the PBR Material Tester 420
23.7.1 Reloading Shaders 421
23.8. The Character Viewer Log Window 421
23.8.1 Setting the Notification Level 421
23.9. Broadcasting the Character Viewer over the Network 422
23.9.1 Joining a DIS Exercise 422
23.9.2 Joining an HLA Exercise 423

24. Adding DI-Guy Models

24.1. How MAK ONE Applications Map DI-Guy Models 426
24.2. Adding New DI-Guy Models to VR-Forces 426
24.2.1 Copy DI-Guy Files to the Data Directory 426
24.2.2 Add a New DI-Guy Entity in the Simulation Object Editor 427
24.2.3 DI-Guy Configuration Files 427
24.3. Adding New DI-Guy Models to VR-Vantage 428
24.3.1 Copy DI-Guy Files to the Data Directory 429
24.3.2 Add Model Definitions and Element Definitions 429

25. DI-Guy Tutorials

25.1. Customizing Appearances Using Texture Swaps 438
25.1.1 Select an Appearance to Modify 439
25.1.2 Change the Texture Map 441
25.1.3 Create Your New Texture Swap and Declare Your New Appearance 442

MAK PRODUCTS GLOSSARY

INDEX







Preface

This manual is written for persons who are creating content for MAK simulation and visualization applications or who must add new content to these applications. The manual assumes that you are familiar with your operating system and that you know how to work in your computer's graphical window environment. It also assumes that if you are creating 3D content that you are familiar with industry standard tools for doing so.

Since the visualization features of VR-Forces and VR-Engage are built using the VR-Vantage Toolkit, they are considered VR-Vantage applications. When that term or "MAK ONE application" are used in this manual, they refer to VR-Forces, VR-Engage, and all of the applications in the VR-Vantage family.

Please see release documentation for information about changes and updates to MAK ONE applications since this manual was issued.

How This Manual is Organized

The chapters are organized as follows:

- "Section I. Introduction" on page 27
- "Section II. Terrain" on page 29
  - "Chapter 1. Composing Terrains" on page 35 explains how to build terrain databases by importing elevation data, imagery, and feature data.
  - "Chapter 2. Paged and Streaming Terrains" on page 57 explains how to use terrain servers for paged and streaming terrains. It also provides information about preparing paged terrains for use in MAK ONE applications.
  - "Chapter 3. Loading Data Using earth Files" on page 63 explains the syntax of earth files so that you can customize them for viewing streaming terrain from terrain servers.
  - "Chapter 4. Editing Terrains Using the Terrain Editor" on page 99 explains how to use the terrain editor to test changes to a terrain.
  - "Chapter 5. Configuring CDB Terrains" on page 103 explains the syntax used in earth files to load CDB terrains.
  - "Chapter 6. Processing MetaFlight Files" on page 123 explains how to preprocess MetaFlight files so that MAK ONE applications can load them.
  - "Chapter 8. Dynamic Terrain in VR-Forces" on page 135 explains how to use the dynamic terrain capability in MAK ONE applications.
  - "Chapter 9. Configuring Feature Data in VR-Forces" on page 139 explains how to create configuration files for feature data whose attributes use different values than the ones that are preconfigured by MAK ONE applications.
  - "Chapter 10. Configuring Airport Settings in VR-Forces" on page 149 explains how to control airport settings either in a plan or by using the Airport Settings Panel.
  - "Chapter 11. Terrain Performance and Configuration" on page 153 covers miscellaneous issues that affect terrain performance and function.
  - "Chapter 12. Terrain Tutorials" on page 185 provides step-by-step instructions for creating and saving a terrain database.
- "Section III. Visual Models in MAK ONE Applications" on page 197
  - "Chapter 13. Introduction to Visual Models" on page 199 introduces visual modeling concepts.
  - "Chapter 14. Schemas" on page 209 explains how to create and edit model schemas.
  - "Chapter 15. Model Definitions" on page 217 explains how to create and edit model definitions.

  - "Chapter 16. Element Definitions" on page 229 explains how to create and edit element definitions.
  - "Chapter 17. Mapping Models and Effects" on page 247 explains how to map entity enumerations to the models configured in MAK ONE applications.
  - "Chapter 18. Wakes, Cockpits, 2D Icons, and Other Models" on page 255 explains how to configure some specific visual models, including 2D Icons, cockpits, and SpeedTree trees.
  - "Chapter 19. The WRM Specification (DIS Notes)" on page 293 explains the WRM specification for 3D modeling.
  - "Chapter 20. The MAK FBX Specification" on page 329 explains the FBX specification for 3D modeling.
  - "Chapter 21. Special Effects (Particle Systems)" on page 375 explains how to edit particle systems.
- "Section IV. DI-Guy" on page 395
  - "Chapter 22. DI-Guy Content" on page 397 explains how to customize DI-Guy models.
  - "Chapter 23. Using the Character Viewer" on page 403explains how to use the Character Viewer to examine DI-Guy animations and character types.
  - "Chapter 24. Adding DI-Guy Models" on page 425explains how to add DI-Guy models to VR-Forces and VR-Vantage.
  - "Chapter 25. DI-Guy Tutorials" on page 437 is a tutorial for editing a texture file.

"MAK Products Glossary" on page 445 defines terms common to DIS, HLA, and MAK ONE products.

MAK Products

The MAK Technologies product line includes the following:

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

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

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

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

  VR-Forces also functions as a plan view display for viewing remote simulation objects taking part in an exercise. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.
- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes VR-Vantage and the VR-Vantage Toolkit.
  - VR-Vantage displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to simulation objects so that it moves as they do.
  - SensorFX. SensorFX is an enhanced version of VR-Vantage that uses physics based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MAK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate

with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.

- VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- DI-Guy™. DI-Guy is an SDK that allows you to embed the DI-Guy library in your real-time application and populate your world with lifelike human characters. DI-Guy provides high-fidelity human characters that move realistically, respond to simple high-level commands, and travel throughout the environment as directed by the hosting visual application. It comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy character simulation is an integral part of all MAK's visual applications, including VR-Engage, VR-Forces, and VR-Vantage.
- RadarFX. RadarFX is a client-server application that simulates synthetic-aperture radar (SAR). The server application, which is based on VR-Vantage and SensorFX, loads a terrain database and, optionally, connects to simulations. A client application requests SAR images from the server. RadarFX includes a sample client application.
- VR-Engage. VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, a sensor operator, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- WebLVC Server. WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several sample applications that work with VR-Forces and VR-Vantage.
- MAK Legion. MAK Legion is a collection of applications and an SDK designed to support simulations with millions of simulation objects.

How to Contact Us

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

Telephone

Call or fax us at:	Voice:	617-876-8085
	Fax:	617-876-9208

E-mail

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

Internet

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

Post

Send postal correspondence to:	MAK Technologies 10 Fawcett Street, Suite 204 Cambridge, MA 02138 USA

When requesting support, tell us the product you are using, the version, and the platform on which you are running. If you are using VR-Vantage, VR-Forces, VR-Engage, or DI-Guy, please also include your data compatibility version and date.

Document Conventions

This manual uses the following typographic conventions:

Sans Serif	Indicates commands, arguments, parameters, or values you enter.
Sans Serif Bold	Indicates a key on the keyboard.
Monospaced Italic	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 MAK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames.
Italic	Indicates a file name, pathname, or a class name.
➤	Indicates a procedure.
Menu > Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose **File > Save**.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

Directory names are preceded with dot and slash characters that show their position with respect to the MAK ONE home directory. For example, the directory *./vrvantage or vrforces3.0.3/doc* appears in the text as *./doc*.

Mouse Button Naming Conventions

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

Third-Party Licenses

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

Third-Party Libraries Supported

VR-Vantage 3.0.3 uses the indicated versions of the following libraries:

- VR-Link 5.8
- Qt 5.15.2 (VC 15 and Red Hat 8)
- OpenSceneGraph 3.6.4
- Boost 1.63.0
- FreeGlut 2.4.0 (Windows), 2.6.0 (Linux)
- GL-Studio 7.0 (VC 15)
- DI-Guy 13.6 (development version)
- Triton 4.43
- SilverLining 6.20
- SpeedTree 6.3.2
- CDB API 2.2 (Windows), 2.1 (Linux)
- osgEarth 3.2
- CUDA 11.7

Library Licensing

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

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

The following libraries are covered by the ZLib license:

- Bullet Physics
- LibPNG
- LibSDL
- Minizip
- Zlib

The following libraries are covered by the MIT license:

- Expat
- FreeGLUT
- GDAL
- GeoTIFF
- GLM
- HarfBuzz
- LibCURL
- LibSquish
- LibXML
- LUA
- LuaBind
- Proj
- SMHasher
- TCLAP
- Qt Visual Graph Editor
- ZeroBrane Studio

The following libraries are covered by the BSD license:

- FreeType
- GLEW
- JPEG
- LibTiff
- Protobuf
- PCRE
- Ogg Vorbis

The following libraries are covered by the commercial licenses:

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

The following libraries are covered by custom licenses:

- NVIDIA CG Toolkit
- FreeImage (FreeImage Public License)
- Boost (Boost Software License)
- Poco (Boost Software License)

COLLADA DOM is covered by the SCEA Shared Source license.

Thread Building Blocks is covered by the GPL license:

SQLLite is in the public domain.

SWIG is covered by the GPL license.

Boost License

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

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

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE

DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libXML and libICONV

VR-Link, and all MAK ONE products that use VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MAK products. MAK has made no modifications to these libraries. For more information about these libraries, see the following websites:

- 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.

Third-Party Licenses for VR-Vantage Applications

VR-Vantage applications use a variety of third-party libraries. Developers who want to use these libraries may be required to purchase developer's licenses. Please see “Third-Party Software and Content” in Chapter 1, Introduction to VR-Vantage, in *VR-Vantage Users Guide* for details.







I. Introduction

Adding Content to MAK ONE Applications

Screen Captures

Adding Content to MAK ONE Applications

The MAK ONE visualization and simulation applications—VR-Vantage, VR-Forces, and VR-Engage—display content that can broadly be characterized as:

- Terrain – the physical environment including land, water, plants, buildings, and so on.
- Models – vehicles and cultural features, such as light poles, benches, barriers and so on. (Cultural features can also be part of the terrain.)
- DI-Guy characters – models of humans and animals that can move in and interact with the terrain and models.

MAK ONE applications come with several sample terrains and have access to worldwide terrain using VR-TheWorldServer or other terrain servers. They also come with many vehicle models and the DI-Guy library of human and animal characters. However, many MAK customers want to add additional content to their MAK ONE applications. This additional content may be highly detailed areas in particular countries, country-specific versions of the generic models provided with the applications, or even classified models that are not available to the general public.

This additional content is created in graphics and modeling software such as Autodesk Maya or 3ds Max. This manual does not explain how to use the software that you use to create content. It assumes you have expertise with these applications. This manual explains how to add your newly created content to MAK ONE applications. Although we do not teach you how to create content, we do provide guidelines for some aspects of terrain and model creation.

Screen Captures

This manual uses screen captures of dialog boxes from VR-Vantage and VR-Forces. When procedures are identical for both applications, we use screen captures from VR-Vantage. If you are using VR-Forces, the dialog boxes you see may have additional options and pages.



II. Terrain

Introduction

Terrain Databases Provided with VR-Vantage Applications

Chapter 1. Composing Terrains

Chapter 2. Paged and Streaming Terrains

Chapter 3. Loading Data Using earth Files

Chapter 4. Editing Terrains Using the Terrain Editor

Chapter 5. Configuring CDB Terrains

Chapter 6. Processing MetaFlight Files

Chapter 7. 2D View Layers

Chapter 8. Dynamic Terrain in VR-Forces

Chapter 9. Configuring Feature Data in VR-Forces

Chapter 10. Configuring Airport Settings in VR-Forces

Chapter 11. Terrain Performance and Configuration

Chapter 12. Terrain Tutorials

Introduction

Each simulation takes place within the extents of a terrain database. A terrain database provides three-dimensional information about a geographic area and provides the environment within which simulation objects function. With the information in a terrain database, MAK ONE applications calculate height, intervisibility, and other three-dimensional information.

MAK ONE applications allow you to load terrain data in the following basic ways:

- Compose a terrain from elevation, imagery, and feature data.
- Open an MTF file.
- Stream terrain from local files or a terrain server such as VR-TheWorld Online.
- Open a terrain database in a supported format, such as FLT, DTED, or GDB.

The recommended work flow is:

1. Create an earth file.
2. Connect to the earth file.
3. Save the terrain as an MTF file.
4. Create a scenario and choose the MTF file as the scenario database.



The decision-making process for choosing a terrain when you create a scenario is described in the chapter *Creating and Running Scenarios* in the *VR-Forces Users Guide*.

Terrain Databases Provided with VR-Vantage Applications

Table 1 describes the terrain databases provided with MAK ONE applications. Some have variants, which indicate the coordinate system or other variations. Terrains with “(online)” in the title connect to VR-TheWorld Server or another online server and require an internet connection. Some terrains have variations that emphasize a particular set of features or other data. MAK ONE applications also have a supplemental data package that includes additional terrains. Please contact your MAK salesperson for download details.

Table 1. Terrain databases shipped with MAK ONE applications

Variation	Description
Aeronautical Charts.mtf	US Sectional Air Charts for IFR (Instrument Flight Rules) with base imagery. This terrain is designed for airspace simulation using the 2D projection (plan view).
Ala Moana.mtf Ala Moana - optimized for VR.mtf	A high-resolution area of Honolulu, Hawaii, USA, set into a raster map of the world. This terrain is stored locally and does not require an internet connection. The optimized version is intended for use with Virtual Reality devices. It is much smaller than the standard Ala Moana terrain.
ArrivalsHall.mtf	An interior model of an international arrivals hall with passport control stations.
Bakersfield Topo Map (online).mtf	VR-TheWorldOnline with topographical maps for the area around Bakersfield, California, USA. This terrain contains some feature data that simulation objects in aggregate-level scenarios can take advantage of.
Brooklyn.mtf Brooklyn (online).mtf	A high resolution city center surrounded by an extruded cityscape. Located geographically in Brooklyn, NY, USA, but not an actual location in that city.
DestructibleTerrain.mtf	A small terrain for demonstrating destructible terrain based on switch nodes.
Ground_DB II	FBX. A simple, generic ground terrain, including a road and some buildings, surrounded by water.
LittlePond (online).mtf	Flat earth. Shape data. Imagery. A detailed suburban inset in a larger, less detailed terrain covering Boston, Massachusetts, USA and points west.
MAK Earth (online).mtf MAK Earth (online) - Optimized for VR.mtf	VR-TheWorld Server terrain with high resolution insets for California, Honolulu, Hawaii, and some other cities. Includes streaming feature data. The optimized version is intended for use with Virtual Reality devices. It is a more basic version of the MAK Earth (online) terrain.
MAK Earth Base (online).mtf	VR-TheWorld Server MAK Earth terrain with high resolution insets for Honolulu, Hawaii, and Boston, Massachusetts. Includes streaming feature data.
MAK Earth Space (online).mtf	A procedurally generated earth merged with high-resolution imagery and elevation.

Table 1. Terrain databases shipped with MAK ONE applications (continued)

Variation	Description
MAK Proving Ground.mtf	A non-realistic terrain with buildings, trees, roads, a runway, different types of soil, slopes, and other environments. It is designed specifically for testing.
MAK Proving Ground Island.mtf MAK Proving Ground Island Geocentric.mtf	A simple island you can use for testing. There is a flat earth version and a geocentric version.
Makland.mtf	UTM. OpenFlight. A fictional terrain created by MAK. It includes a variety of terrain and feature data, including underwater features. It is placed in the Indian Ocean.
Massachusetts S-57 (online).mtf	A terrain that has S-57 chart data for Boston Harbor in Massachusetts.
ProceduralEarth (online).mtf ProceduralEarth - OpenStreetMap (online).mtf	Procedurally generated terrains. Images are generated based on land use data, resulting in high quality visual imagery without the overhead of high resolution static images. Terrains include procedural airports with airport data for all airports in the world, and street data from OpenStreetMap.
RhoneValley (online).mtf	A procedurally generated view of the Rhone Valley in France, including procedural airports.
Sadr City.mtf Sadr City (online).mtf	An urban geotypical Middle Eastern terrain in Sadr City, Iraq, a suburb of Baghdad. Ideal for urban scenarios.
SanLuisObispo.mtf	Flat earth. DTED. Imagery. A terrain that covers a portion of California, USA.
SanLuisObispoTopoGdal.mtf	Geocentric. San Luis Obispo in topographic coordinates as local streaming terrain using an earth file.
Test terrains	Multiple terrains used for testing different terrain database capabilities.
Truman Carrier.mtf	A model in VR-Forces of an aircraft carrier as a terrain object rather than a simulation object.
VR-Village.mtf	GDB. Feature data. OpenFlight terrain. VR-Village is a fictional database created by MAK. It has a highly detailed small town in a larger, sparsely detailed desert environment. The town includes multilevel buildings, underground tunnels, and a road that tunnels through a mountain. The coordinates place it off the west coast of Africa.

Table 1. Terrain databases shipped with MAK ONE applications (continued)

Variation	Description
WorldFlatEarthNoElevation.mtf	Flat earth. Low resolution projected image of the world.
WorldGeocentric.mtf	Geocentric. Low resolution projection of the world.



1. Composing Terrains

This chapter explains how to combine various types of data to build a terrain. For basic conceptual information about terrain agility, see "Section II. Terrain" on page 29. For information about paged and streaming terrains, see "Chapter 2. Paged and Streaming Terrains" on page 57.

Using the procedures in this chapter, you can compose a simple terrain. The earth file format supports far more robust features. See "Chapter 3. Loading Data Using earth Files" on page 63.

1.1. Creating a Composed Terrain 36
1.1.1 Saving a Terrain 36
1.2. Adding Elevation Data to a Terrain 37
1.3. Adding Images to a Terrain 39
1.4. Feature Layers 39
1.4.1 Adding a Feature Layer 40
1.4.2 Removing a Feature Layer 41
1.5. Adding a Dynamic Ocean Layer 41
1.5.1 Extracting Water Textures and Adding an Ocean Layer 42
1.6. Adding a Dynamic Ocean Layer Directly 44
1.7. Setting the Ocean LOD Elevation 45
1.8. Configuring the Ocean Height Map 46
1.8.1 Setting the Ocean Height Map Resolution 46
1.8.2 Setting the Ocean Height Map Texture Size 47
1.8.3 Setting the Ocean Height Map Regeneration Threshold 47
1.9. Specifying Simulation Ocean Settings 48
1.10. Adding Props to a Terrain 49
1.10.1 Extracting Props from a Terrain Patch 49
1.10.2 Adding Props from a Feature Layer 51
1.10.3 Viewing a List of Props 53
1.10.4 Selecting Props 53

1.10.5 Changing a Prop's Type 53
1.10.6 Changing a Prop's Position or Orientation 54
1.10.7 Changing a Prop's Model Definition 54
1.10.8 Exporting Props and SpeedTrees to a Shapefile 54
1.10.9 Importing Props and SpeedTrees from Shapefiles 55

1.1. Creating a Composed Terrain

Composing a terrain means building a terrain in the GUI by importing various kinds of terrain data and then saving the terrain as an MTF file (MAK Terrain Format configuration file). You can also define terrains using an earth (*.earth*) file, which an MTF file can reference. (MTF files can combine various kinds of terrain data.)

For details about using earth file, see "Chapter 2. Paged and Streaming Terrains" on page 57 and "Chapter 3. Loading Data Using earth Files" on page 63. For details about using the Terrain Editor, see "Chapter 4. Editing Terrains Using the Terrain Editor" on page 99.

The general procedure for composing a terrain through the GUI is as follows:

1. Choose **File > New Terrain**, or click the New Terrain button ( ) on the Terrain Toolbar. If any terrain data is loaded, it is removed.
2. Add elevation data, as described in "1.2. Adding Elevation Data to a Terrain" on the facing page.
3. Optionally, add imagery, as described in "1.3. Adding Images to a Terrain" on page 39.
4. Optionally, add feature data, as described in "1.4. Feature Layers" on page 39.
5. Optionally, add a dynamic ocean layer, as described in "1.5. Adding a Dynamic Ocean Layer" on page 41.
6. Optionally, add props or extract props, as described in "1.10.2 Adding Props from a Feature Layer" on page 51 and "1.10.1 Extracting Props from a Terrain Patch" on page 49.
7. Save the terrain, as described in "1.1.1 Saving a Terrain" below.

For a list of supported terrain and feature formats, see:

- "Supported File Formats" in the *VR-Vantage Users Guide*
- "Supported File Formats" in the *VR-Forces Users Guide*

1.1.1 Saving a Terrain

When you save a terrain, you save information about the terrain patches, feature layers, and the dynamic ocean layer that you used to create the terrain. Saving a terrain does not save out data to a file format comparable to OpenFlight or DTED that can be loaded into other applications or converted into other formats. If accelerated file loading is enabled, the data gets saved in a format that can be loaded more

quickly than from its native format. (For details about accelerated file loading, see "11.1. Configuring File Caching" on page 154.) Terrains get saved with the extension *.mtf*.

When you save a terrain to MTF, if an MTD file of the same name exists, a dialog box explains your options for saving the file. (MTD is a legacy format that might be present if you have terrains created with older versions of MAK ONE applications.)

➤ **To save a terrain:**

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

1.2. Adding Elevation Data 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 a terrain patch.

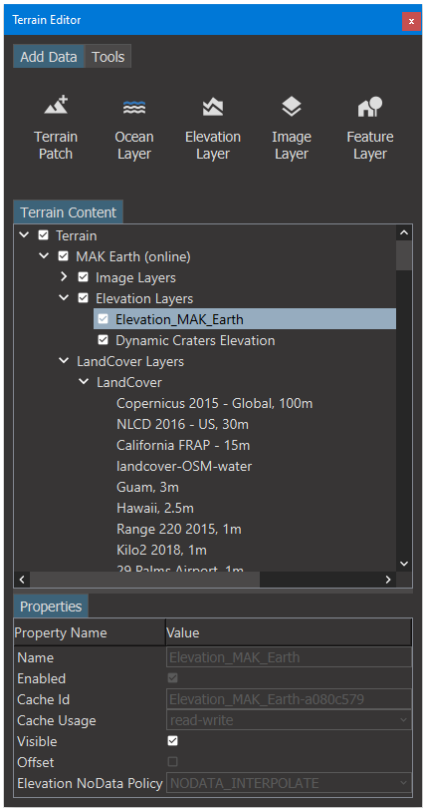
You can use this procedure to add a single *.earth* file and other terrain files. Some are found in the *./SharedData/16/latest/TerrainData/TerrainConfiguration* directory.

- Each time you add an elevation layer or terrain patch, the application 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.
- You can only have one *.earth* file loaded at any time. You cannot add multiple *.earth* files using the Add Terrain Patch dialog box. If you already have a *.earth* file loaded, you cannot add another *.earth* file. You can add multiple other non-*.earth* files.

➤ **To add an elevation layer to a terrain database:**

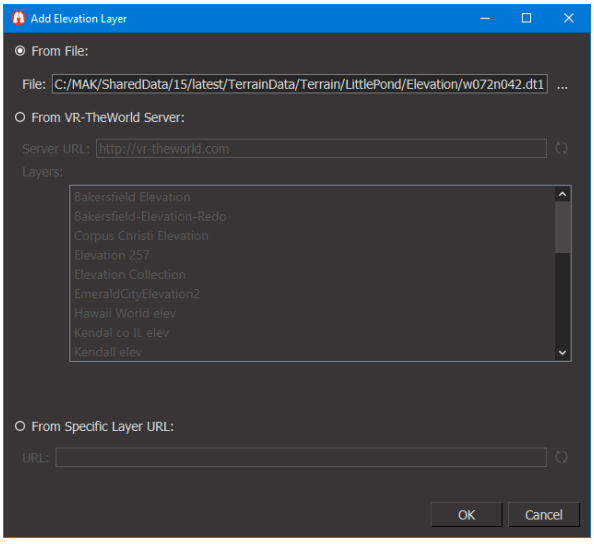
1. Choose **View > Terrain Editor Panel.** The Terrain Editor Panel opens.
2. Select the Add Data tab (Figure 1).

Figure 1. Terrain Editor Panel



3. Click Elevation Layer. The Add Elevation Layer dialog box opens.
4. Select the file, the layer from the VR-TheWorld Server, or the URL (1.2)

Figure 2. Add Elevation Layer dialog box



5. Click OK. The elevation data is added to the terrain.
6. Save the terrain.

1.3. Adding Images to a Terrain

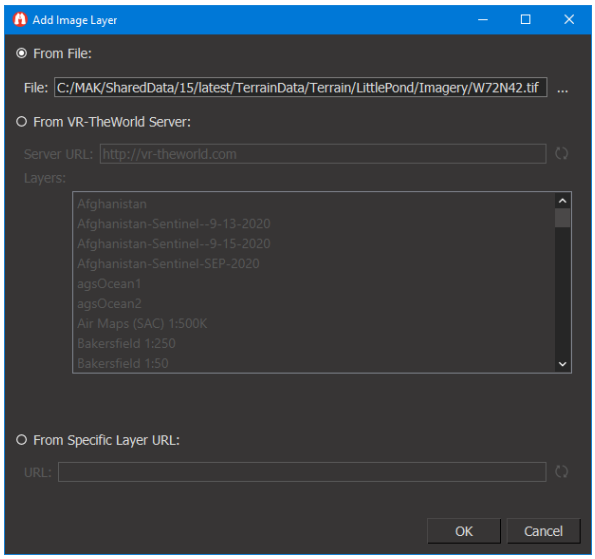
Depending on your terrain source, you may have terrain polygonal data, but no texture data or feature data to provide a realistic 3D display. You can drape images (raster maps) over the terrain to provide a more pleasing display.

Adding very large images (multi-100 GB) can cause the application to crash. If you need to load very large images, load them using an earth file and the GDAL driver. The *SanLuisObispo-Imagery.earth* file demonstrates how to load an image using this method. You can also use the GDAL driver to load raster formats that MAK ONE applications cannot load directly. For a list of formats supported by GDAL, go to https://www.gdal.org and select the Supported Formats link.

➤ **To add raster maps to the terrain:**

1. Choose **View > Terrain Editor Panel**. The Terrain Editor opens.
2. Click Image Layer. The Add Image Layer dialog box opens (Figure 3).

Figure 3. Add Image Layer



3. Click the Browse button for the File. The Terrain Layer Source dialog box opens.
4. Select the raster image that you want to add to the terrain and then click Open. The file name populates in the Add Image Layer dialog box.
5. Click OK. The image is listed under Image Layers in the Terrain Content.
6. To hide or display the layer in the main window, clear or select the check box to the left of the entry.

1.4. Feature Layers

A feature layer contains data about geographic features on the terrain. This includes linear features, such as roads, and point features, such as trees and buildings. You can create props from point features. A feature layer can be created from a MAK Terrain

Format (GDB) file, a shapefile (SHP), or DFAD data (DFD).

VR-Vantage applications (such as VR-Vantage or the VR-Forces front-end) and the VR-Forces simulation engine use different mechanisms to import feature data.

When a VR-Vantage application imports a shapefile, it uses the name of the shapefile to indicate what type of feature it contains. Filenames are mapped to feature types in *./appData/importConfig/filename_shp_map.txt*. For example, the file name *ROADL.shp* is mapped to feature 129, which is a road. If the shapefiles that you need to import are not listed in *filename_shp_map.txt*, you can add them by editing the file.



In VR-Forces, loading large feature layers into the front-end can cause it to hang and scenario data can be lost. To avoid this problem, you can choose to load feature layers only in the simulation engine. In this case, you will not be able to visualize the feature data, but the simulation engine will handle it correctly.

When the VR-Forces simulation engine imports feature data, it configures it based on a configuration file called *./appdata/settings/featureconfig.txt*. For details about this file, please see "Chapter 9. Configuring Feature Data in VR-Forces" on page 139.

The preferred method for adding feature data in the back-end is via shapefiles. Loading shapefiles in the back-end, as compared to loading feature data from a GDB file, provides better performance. If you have feature data in GDB files and you need to import it, you can extract it from the GDB to a shapefile using the vectorNetworkToShapeFiles utility. For details, please see "9.7. Extracting Feature Data from GDB to Shapefiles" on page 144.



- Adding a feature layer does not, by itself, make any visible difference to a terrain. To see the feature data, you must extract the data as props and map the props to model definitions. For details, please see "1.10. Adding Props to a Terrain" on page 49.
- The number of layers you can add depends on your graphics card. If you add a layer and it does not appear, try reducing the number of layers you have.
- Feature layers in earth files are added automatically.
- MAK ONE applications support S-57 data. The recommended way to load S-57 data is to convert it to shapefiles and load the shapefiles. You could load them as feature layers as described in this section. However, since each S-57 file is converted to multiple shapefiles, it may be easier to load them using earth files than as individual shapefiles. MAK ONE applications include one terrain that has S-57 data, *Massachusetts S-57 (online).mtf*.

1.4.1 Adding a Feature Layer

➤ **To add a feature layer:**

1. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.
2. Select the Add Data tab (Figure 1 on page 38).

3. Click Feature Layer. The Add Feature Layer dialog box opens.
4. Type the path of the feature data file you want to load, or click the Browse button and select a feature data file.
5. If you want the feature layer to be loaded in VR-Vantage, the VR-Forces front-end, or both, select Apply to Front End. Clear Apply to Front End to add a feature layer only in the back-end. (Apply to Simulation Engine is always selected and is not editable.)
6. Click OK.
7. Save the terrain.

1.4.2 Removing a Feature Layer

You can remove a feature layer from a terrain.

➤ **To remove a feature layer:**

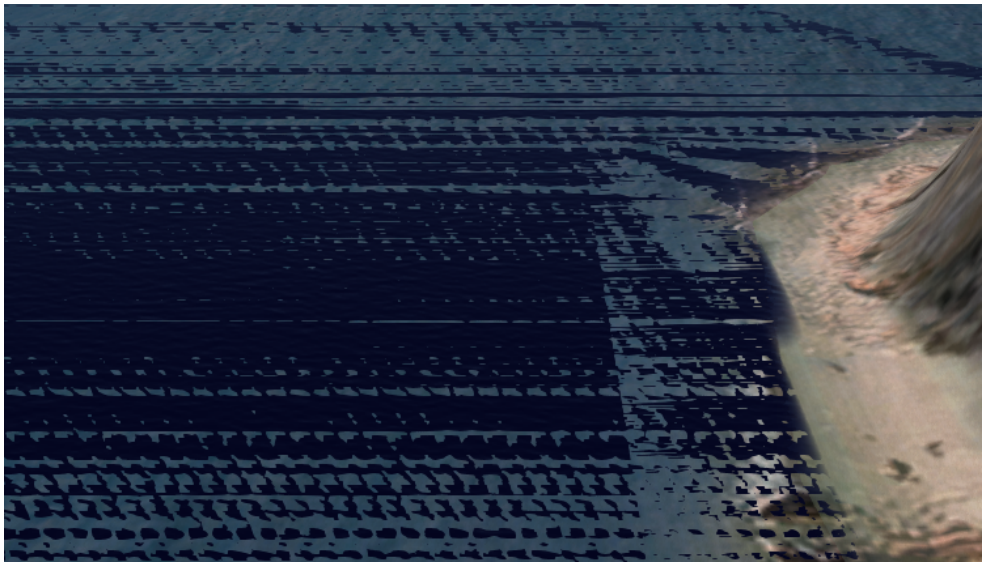
1. In the Terrain Editor Panel, right-click on the feature layer and then choose **Remove** from the context menu.
2. Save the terrain.

1.5. Adding a Dynamic Ocean Layer

MAK ONE applications can simulate dynamic ocean effects, such as waves, swell, and chop. To simulate the ocean, your terrain must have a dynamic ocean layer. You can add an ocean layer directly or you can add one as part of the process of extracting water textures.

If your terrain has textures assigned to water polygons, then you should extract them. They will be replaced by the ocean layer. If you do not extract the water textures and you simply add an ocean layer, the ocean layer and the water texture will interfere with each other in a process called Z-fighting and create an unacceptable display (Figure 4). This problem occurs because the ocean layer and the texture are both at the same height (0, or sea level).

Figure 4. Z-fighting



If your terrain does not have a specific water texture assigned to polygons, your only option is to add an ocean layer directly. However, since the ocean layer will be at the same elevation as the terrain texture, you may experience Z-fighting. You may also have to raise the ocean height to actually see the ocean effects.

Figure 5 shows the Ala Moana terrain with a water texture and an ocean layer. In the left view, the ocean layer has a height of -3 meters. You cannot see it. In the right view, the ocean layer is set to 0 meters. Now it is visible.

Figure 5. Ocean height



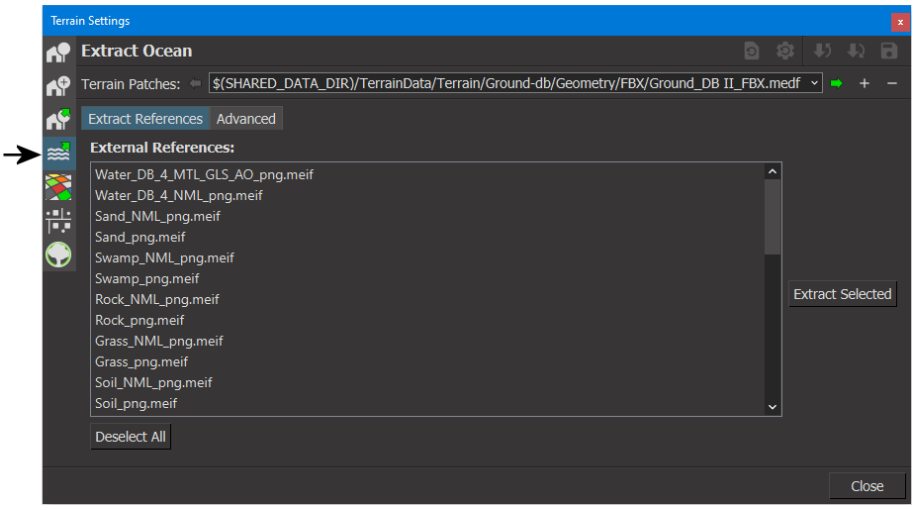
1.5.1 Extracting Water Textures and Adding an Ocean Layer

You cannot extract water textures when indirect rendering is enabled. You can temporarily disable direct rendering as described in "11.5.3 Disabling Indirect Rendering for Terrain" on page 165.

➤ **To extract textures and add an ocean layer:**

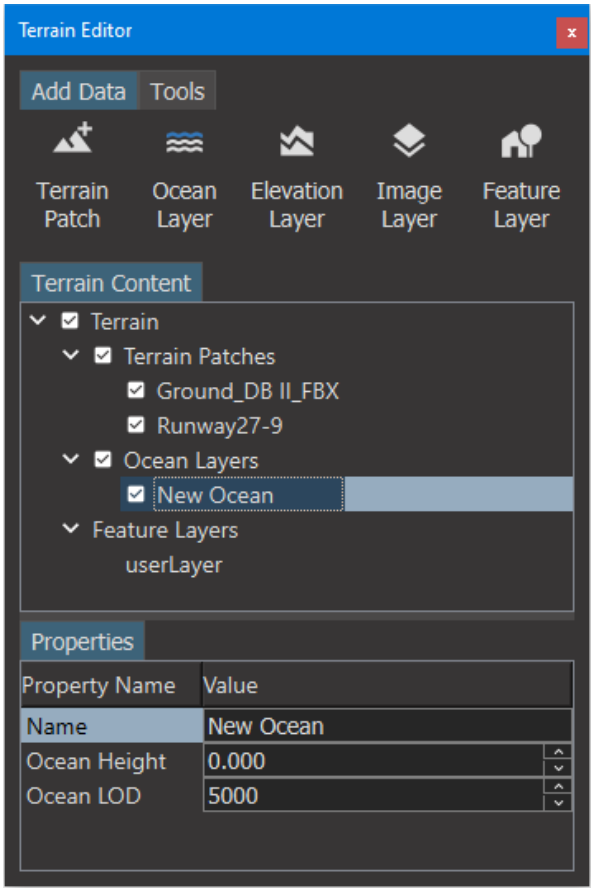
1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Extract Ocean page (Figure 6).

Figure 6. Terrain Settings dialog box, Extract Ocean page



3. In the External References list on the Extract References tab, select the water textures that you want to extract.
4. Click Extract Selected. The textures are removed and an ocean layer is added.
5. Choose **View > Terrain Editor Panel**. The Terrain Editor opens.
6. Select the Ocean Layer (Figure 7).

Figure 7. Ocean layer properties



7. Adjust the Ocean Height so that the dynamic ocean is visible. This might require raising the ocean above sea level.
8. Optionally, adjust the Ocean LOD elevation. For information about the Ocean LOD elevation, please see "1.7. Setting the Ocean LOD Elevation" on the facing page.
9. Save the terrain.

1.6. Adding a Dynamic Ocean Layer Directly

Ocean layers are created at a height of 0 meters, that is, mean sea level.

➤ **To add a dynamic ocean layer:**

1. Choose **View > Terrain Editor Panel**. The Terrain Editor opens (Figure 7 above).
2. Select the Add Data tab.
3. Click Ocean Layer. (You can only have one ocean layer. If you already have one, nothing happens.) An ocean layer is added and listed in the Terrain Content list.
4. Optionally, change the name of the ocean layer. When you create a new ocean layer, the default name is New Ocean.

5. Adjust the Ocean Height so that the dynamic ocean is visible. This might require raising the ocean above sea level.
6. Optionally, adjust the Ocean LOD elevation. For information about the Ocean LOD elevation, please see "1.7. Setting the Ocean LOD Elevation" below.

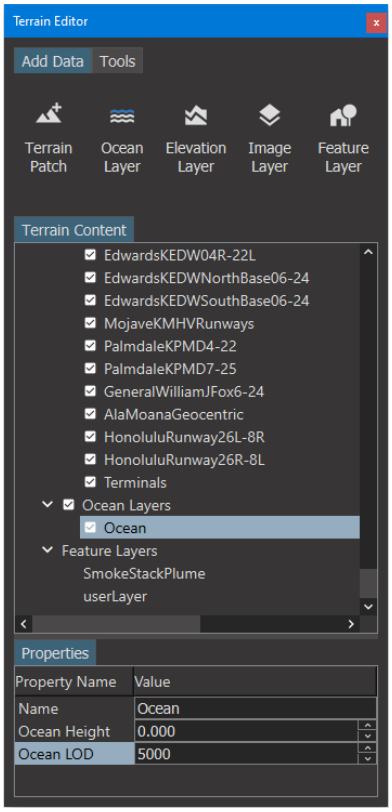
1.7. Setting the Ocean LOD Elevation

Enabling dynamic ocean affects performance. Since the effects are hard to see once the observer is high above the ocean, MAK ONE applications stop displaying them when the observer is more than 5000 meters above sea level. You can change this cutoff point by adjusting the ocean LOD elevation.

➤ **To set the ocean LOD elevation:**

1. Choose **View > Terrain Editor Panel**. The Terrain Editor opens.
2. Select the Ocean Layer (Figure 8).

Figure 8. Ocean layer properties



3. Edit the Ocean LOD property.
4. Save the terrain using the Save Terrain procedure.

1.8. Configuring the Ocean Height Map

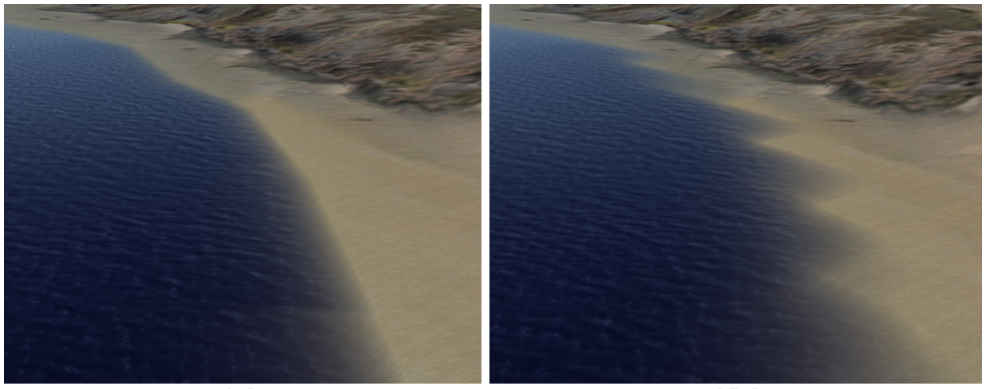
The ocean height map is used to blend the dynamic ocean with the coastline. You can configure the height map resolution, its texture size, and the threshold for regenerating the height map. All of these settings interact to affect visual quality and performance.

Height map resolution is specified as meters per pixel. The height map texture size is specified in pixels. Therefore these two settings determine the length of one side of the (square) area that the height map covers. For example, if the value for height map resolution is 10 meters per pixel and the texture size is 4096 pixels, this results in a height map that is 40960 meters on a side.

As height map resolution increases and texture size decreases, visual quality may improve, but performance can degrade. Conversely, decreasing resolution and increasing the texture size can improve performance at a cost of visual quality. You will need to experiment to determine the best values for your simulation.

If you are working with a small terrain or scenes in which the observer is close to sea level, you may want to use a higher resolution. If you are less interested in how the coastline looks, you can use a lower resolution. Figure 9 shows a coastline at different levels of resolution.

Figure 9. Ocean height map resolution



1.0 15.0

Setting the surface transparency also affects visualization of the coastline. Experiment with these values to achieve the best results. For details about water transparency, see "Configuring Marine Conditions" in the *VR-Vantage Users Guide* or "Configuring Marine Conditions" in the *VR-Forces Users Guide*.

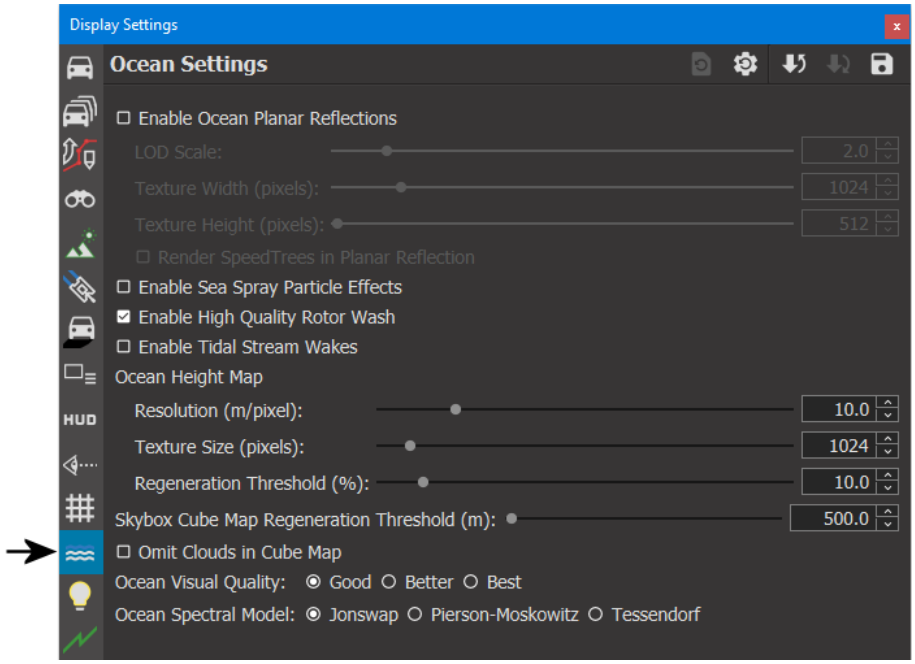
1.8.1 Setting the Ocean Height Map Resolution

The ocean height map resolution is specified at meters per pixel.

➤ **To set the ocean height map resolution:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 10).

Figure 10. Ocean Settings



3. Adjust the Ocean Height Map Resolution slider, or enter a value in the box.

1.8.2 Setting the Ocean Height Map Texture Size

The ocean height map texture size is specified in pixels.

➤ **To set the ocean height map texture size:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 10 above).
3. Adjust the Ocean Height Map Texture Size slider, or enter a value in the box.

1.8.3 Setting the Ocean Height Map Regeneration Threshold

The ocean height map is recalculated as the observer moves. Generating the height map is computationally expensive. Therefore, it is done only when the observer moves by more than a specified threshold. You can adjust this threshold if you want to recalculate the height map more frequently. The default is 10% (0.1). This means that the observer must move by 10% of the length of one side of the height map before the height map is regenerated.

➤ **To set the ocean height map regeneration threshold:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 10 above).
3. Adjust the Ocean Height Map Regeneration Threshold slider, or enter a value in the box.

1.9. Specifying Simulation Ocean Settings

VR-Forces can determine the depth of the ocean floor from:

- Underwater polygons, if present.
- Feature data, such as S-57 data.
- As a fixed depth.

If a terrain has underwater polygons, VR-Forces always automatically uses that data. You can enable and disable use of feature data that sets the ocean depth. If you have both types of data, VR-Forces uses the shallowest depth of the two types of data for any given point.

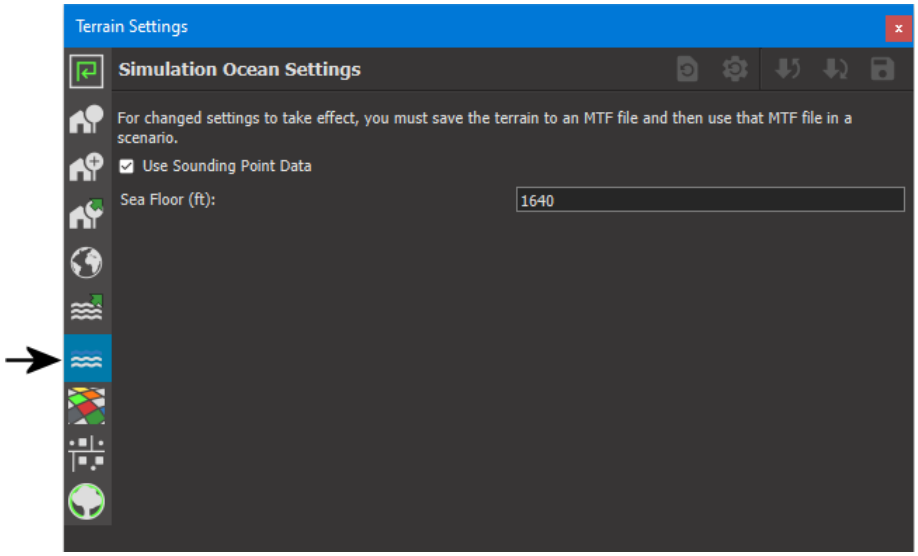
You can also specify that VR-Forces use a fixed depth for any area that does not have underwater polygons or feature data.

Changing the simulation ocean setting is not a dynamic change. After changing the settings, you must save the terrain to an MTF file and reload the scenario.

➤ **To specify ocean depth:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Simulation Ocean Settings page (Figure 11).

Figure 11. Simulation Ocean Settings page



3. To enable or disable use of feature data, select or clear the Use Sounding Point Data check box.
4. To specify the ocean depth for areas that do not have underwater polygons or feature data, type a value in the Sea Floor text box.
5. Save the terrain.
6. Reload the scenario.

1.10. 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 Simulation Objects Palette that look like similarly named props in the 3D view. Cultural features are not the same things as props. Props are saved as part of the terrain. Cultural features are simulation objects and are saved as part of the scenario. They are a mechanism for placing objects into a scenario for visualization or simulation purposes.

1.10.1 Extracting Props from a Terrain Patch

When you extract props from a terrain patch, you have to tell MAK ONE applications 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. The application 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 the application 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 the application auto-generate prop types. If you choose Auto Generate, the application makes the prop type match the name of the referenced file. For example, a prop extracted from the file *Bridge_Arch_Stone.flt* would be given the prop type *Bridge_Arch_Stone.medf*.
- Model definition to use: You can select a model definition from a list, or choose Use Extracted Geometry. If you accept the default value, Use Extracted Geometry, the application creates props using the geometry in the terrain patch. For example, if you chose Trees.flt as the referenced file, the application creates props using the tree geometry in that file.

MAK ONE applications extract 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 "12.2. Extracting Props from a Terrain" on page 191.

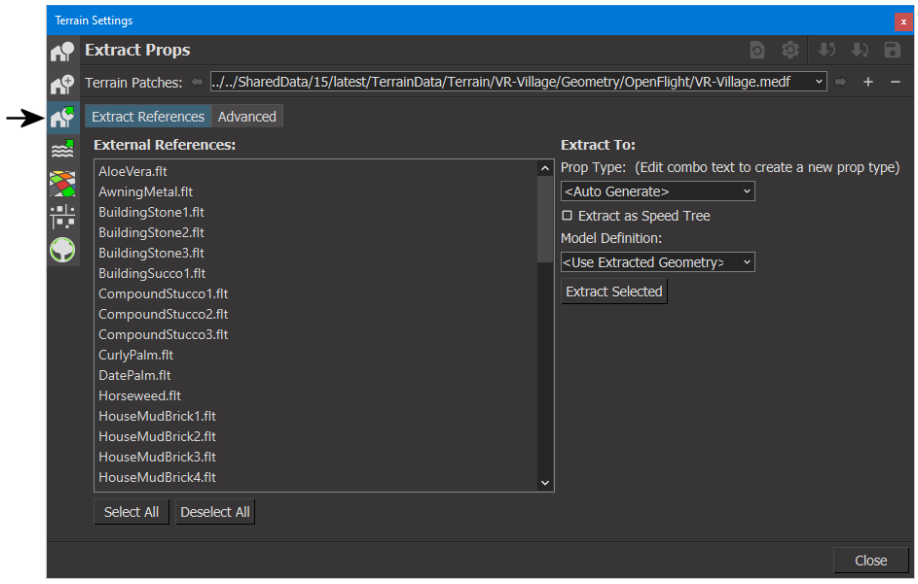


You cannot extract props when indirect rendering is enabled. You can temporarily disable direct rendering as described in "11.5.3 Disabling Indirect Rendering for Terrain" on page 165.

➤ **To extract props from a terrain patch:**

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

Figure 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.

   MAK ONE applications use 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. The props, if any, are extracted. You can view the list of props in the Props list on the Edit Existing Props page.

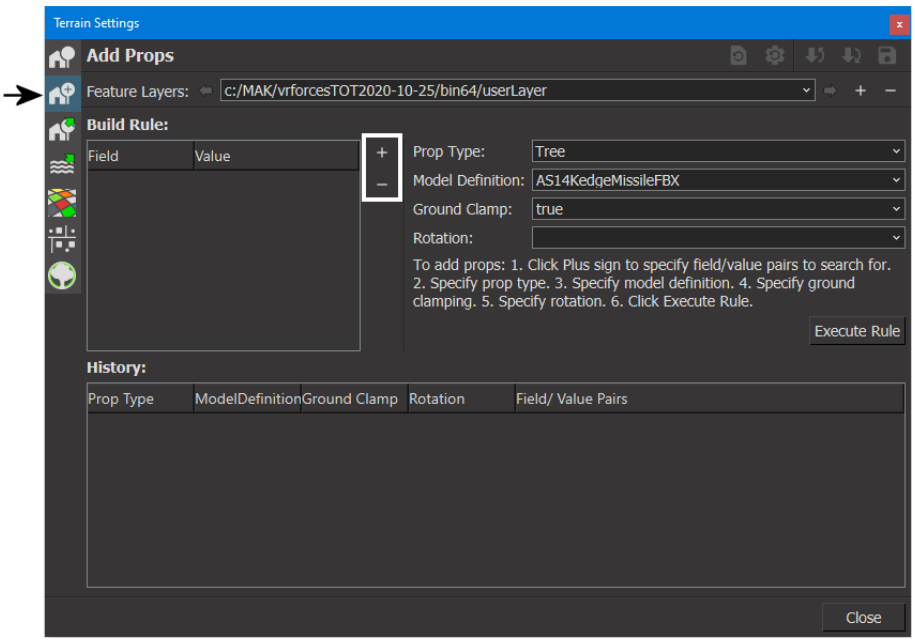
1.10.2 Adding Props from a Feature Layer

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

➤ **To add a prop from a feature layer:**

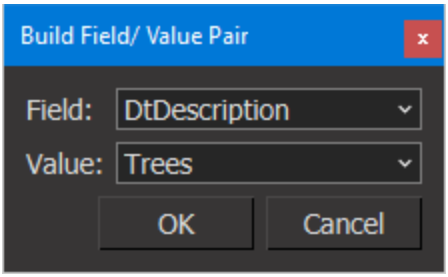
1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Add Props page (Figure 13).

Figure 13. Terrain Settings, Add Props page



3. In the Feature Layers list, select the feature layer from which you want to import the props. The layer must be a point or linear layer; it cannot be areal.
4. Click the Add button (+) to the right of the Field/Value window (Figure 13). The Build Field/Value Pair dialog box opens (Figure 14).

Figure 14. Build Field/Value Pair dialog box

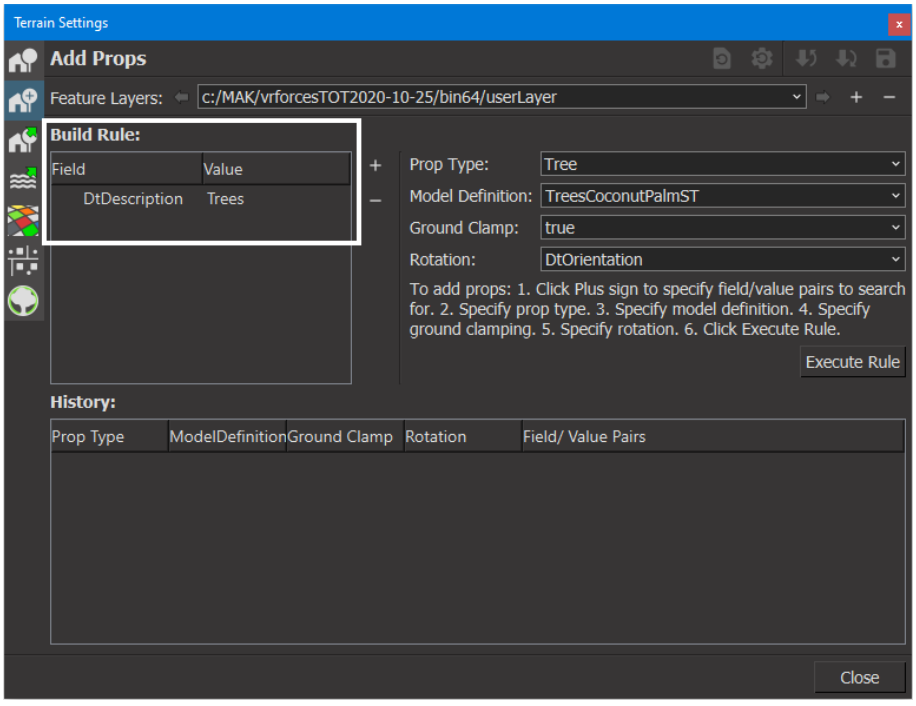


5. In the Field list, select a field. Fields are the column headings in the feature definition.
6. In the Value list, select a value. Values are one or more values from the definition table that are available for the column heading.

> It is your responsibility to understand the feature definitions for the features that you want to import as props.

7. Click OK. The Field/Value pair is added to the window (Figure 15).

Figure 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 list, select the prop type to assign to the prop you want to add.

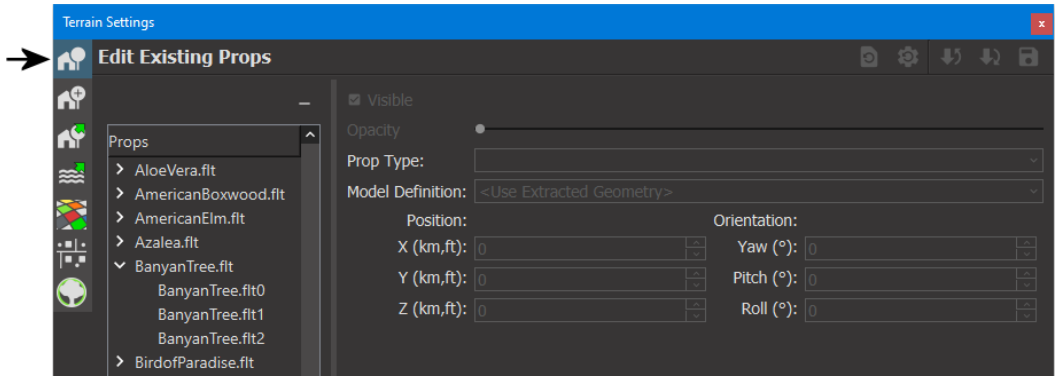
10. In the Model Definition list, select the model definition you want to use for this prop.
11. In the Ground Clamp box, specify whether or not to ground clamp this prop.
12. Optionally, set the rotation.
13. Click Execute Rule. The prop is extracted from the feature layer and is listed in the History window.

1.10.3 Viewing a List of Props

➤ **To view a list of the props in the terrain:**

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

Figure 16. Edit Existing Props page



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

1.10.4 Selecting Props

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

Use **Ctrl**+click or **Shift**+click to select multiple props.

1.10.5 Changing a Prop's Type

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

➤ **To change a prop's type:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 16 above).
3. In the list of props, select the props whose type you want to change.
4. Select a prop type from the Prop Type list, or type a new type name. The selected props are moved to the new prop type in the Props window.

1.10.6 Changing a Prop's Position or Orientation

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

➤ **To change a prop's position or orientation:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 16 on the previous page).
3. In the list of props, select the prop whose position or orientation you want to change.
4. To change position, type or select new values in the X, Y, or Z boxes.
5. To change the orientation, type or select new values in the Yaw, Pitch, or Roll boxes.

1.10.7 Changing a Prop's Model Definition

You can change the model definition used to render a prop. For information about model definitions, please see "15. Model Definitions" on page 217.

➤ **To change a prop's model definition:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Edit Existing Props page (Figure 16 on the previous page).
3. In the list of props, select the prop whose model definition you want to change.
4. In the Model Definition list, select a model definition. The selected props are rendered using the new model definition.

1.10.8 Exporting Props and SpeedTrees to a Shapefile



In VR-Forces, you can export props and SpeedTrees to a shapefile. This is particularly useful if you have added props from the Props Palette and want to include them in a terrain, but do not want to save them as part of the terrain. You can then add the props to a terrain from an earth file.



When you export props, VR-Forces automatically exports props and SpeedTrees as one operation. If you are exporting to a new file, VR-Forces takes the name you enter and appends BuildingsP.shp and VegetationP.shp to create the shape files. For example, if you specify the file name myShapeFiles, VR-Forces saves to *myShapeFilesBuildingsP.shp* and *myShapeFilesVegetationP.shp*.

If you select an existing file as the export target, VR-Forces uses that file name. It still saves both buildings and trees.

➤ **To export props or SpeedTrees to a shapefile:**

1. Choose **File > Terrain > Export Props to Shapefile.** The Save Props as Shapefile dialog box opens.
2. Type a name for the shapefile.
3. Click Save.

1.10.9 Importing Props and SpeedTrees from Shapefiles



If you have exported props or SpeedTrees to a shapefile, you can import those shapefiles into VR-Forces.



You can only import shapefiles that were created using the VR-Forces Export Props to Shapefile process.

➤ **To import props or SpeedTrees from a shapefile:**

1. Choose **File > Terrain > Import Props From Shapefile.** The Import Props From Shapefile dialog box opens.
2. Select the shapefile you want to import.
3. Click Open.





2. Paged and Streaming Terrains

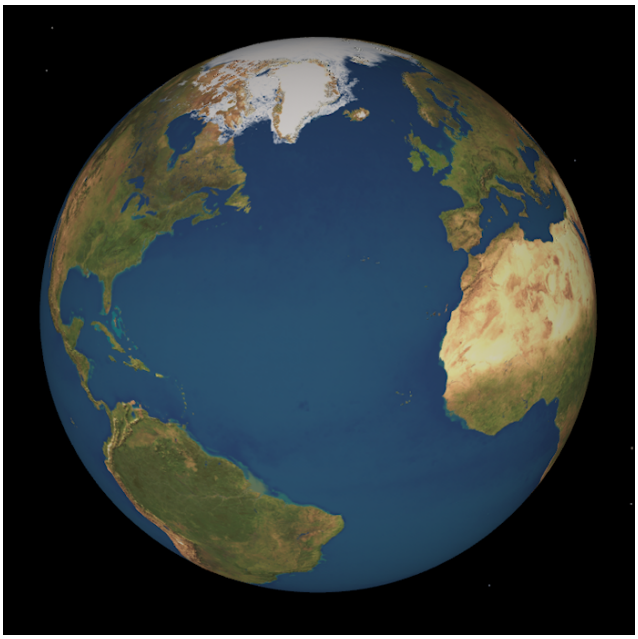
This chapter explains how to use terrain servers for paging and streaming terrains. It also provides information about preparing paged terrains for use with MAK ONE applications.

2.1. Connecting to Terrain Servers 58
2.1.1 Adding Terrain Server Connections 60
2.1.2 Connecting to Terrain Servers through a Proxy 60
2.2. Loading MetaFlight Terrains 60
2.3. Preprocessing Paged Terrains 61
2.4. Manually Paging-In Terrain in VR-Forces 61

2.1. Connecting to Terrain Servers

MAK ONE applications support the osgEarth libraries for loading terrains from remote or local data sources (Figure 17). Terrain servers are configured in earth files. They support streaming of terrain data and are, therefore, ideal for displaying terrains that have large amounts of data.

Figure 17. Geocentric terrain



You can manipulate streamed terrains with the keyboard and mouse and can save them as an MTF terrain for quick loading.

MAK ONE applications include several terrain server configurations for VR-TheWorld Online, a server hosted by MAK Technologies. You must have an internet connection to connect to these servers. It also includes server configurations that stream local terrain files included with the applications. These servers, such as, *Local World.earth* and *SanLuisObispo-Topo.earth*, are examples of how you can use earth files to load local data.



Integration of osgEarth into MAK ONE applications 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.

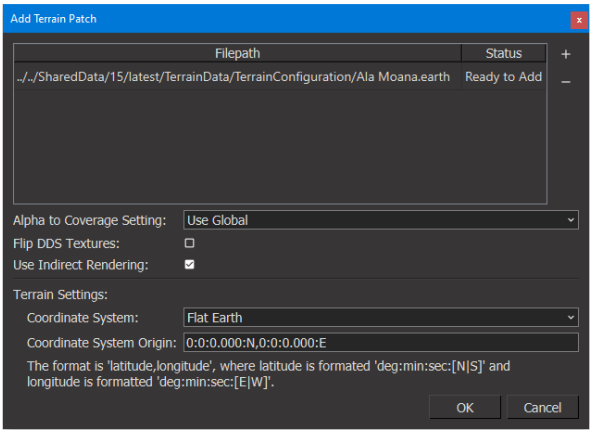
"12.1. Creating an earth File to Load Terrain Data" on page 186 is a tutorial that shows how to create a terrain server connection that serves local data.

When you view a geocentric terrain and move the observer more than 6 million meters or so from the Earth's surface, the view is forced to focus toward the center of the Earth. The practical effect of this mode is that the left and right movement keys orbit the observer around the Earth, rather than moving the entire Earth to the left or right.

➤ **To connect to a terrain server:**

1. In VR-Vantage, choose **File > New Scene**. In VR-Forces, choose **File > Terrain > New Terrain**. The current terrain closes.
2. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens (Figure 1 on page 38).
3. Select the Add Data tab.
4. Click Terrain Patch. The file selection dialog opens.
5. Navigate to *./SharedData/16/latest/TerrainData/TerrainConfiguration*.
6. Select an earth file. The Add Terrain Patch Object dialog box opens (Figure 18).

Figure 18. The Add Terrain Patch dialog box.



7. Optionally, select the Use Indirect Rendering check box. If the check box is selected, the terrain patch uses the high performance indirect rendering system. If after loading a terrain there are rendering artifacts that were not present in a previous release of the application, try loading the terrain patch again with Use Indirect Rendering cleared. For more information, please see "11.5. Indirect Rendering of Terrain" on page 164.
8. If necessary, change the Terrain Settings to match the terrain's coordinate system. (When the selected terrain settings are not correct, an error message displays at the bottom of the dialog box.)
9. Click OK. If the terrain data is available (remote sources might not always be accessible), it displays.

You can save a terrain server as part of a terrain or scene. For details, please see "1.1.1 Saving a Terrain" on page 36.

2.1.1 Adding Terrain Server Connections

You can add additional terrain server connections to the list provided with the application. Terrain server connections are configured in XML files according to the requirements of the osgEarth libraries. The server configurations included with MAK ONE applications 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.



All earth files must be placed in *./SharedData/16/latest/TerrainData/TerrainConfiguration*.

"12.1. Creating an earth File to Load Terrain Data" on page 186 is a tutorial that shows how to create a terrain server that serves local data.

2.1.2 Connecting to Terrain Servers through a Proxy

If you connect to the network through a proxy, you may need to set some environment variables to be able to connect to external terrain servers. MAK ONE applications have limited proxy capability. It supports basic authentication and may support domain authentication if your proxy allows it to be accessed through basic authentication. To set up authentication, set the following environment variables:

- OSG_CURL_PROXY. Specifies the proxy machine.
- OSG_CURL_PROXYPORT. Specifies the proxy port.
- OSGEARTH_CURL_PROXYAUTH=username:password. Specifies the user name and password for the proxy.

2.2. Loading MetaFlight Terrains

MAK ONE applications supportMetaFlight terrain databases. To load a MetaFlight terrain, add a terrain patch as you would for any other terrain. Before you can load a MetaFlight file, however, you must process it using the mftTool and medfTool. For details, see "Chapter 6. Processing MetaFlight Files" on page 123.

When you load a MetaFlight file, the application is actually loading many individual OpenFlight files. If file caching is enabled, the application 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, see "11.1. Configuring File Caching" on page 154.



Geometry files referenced by a MetaFlight file must be in OpenFlight format, and not in Vega Prime's binary format (*.vsb*). MAK's terrain libraries do not support *.vsb* files.

2.3. Preprocessing Paged Terrains

MAK ONE applications support preprocessing external references used by paged terrains. MAK ONE applications do not require terrain formats such as TerraPage or pageable IVE to be preprocessed. However these terrains may use external references in formats such as OpenFlight, which load slowly. Although MAK ONE applications can accelerate these files at run-time using the accelerated cache feature, the first time you load one of these terrains you can experience significant performance issues. You can use the medfTool to preprocess these external references and save them in MEDF format. Thereafter, the application loads the accelerated version at all times.

For example, if a TerraPage terrain references */SharedData/16/latest/ModelData/Vegetation/Trees/tree.flt*, preprocessing creates a file called *tree.medf* in the same folder. When the application loads the terrain, it loads the MEDF version instead of the OpenFlight version.

We recommend that you preprocess all data to minimize load times and maximize run-time performance. For details about the medfTool, please see "18.16. Compressing Model Files" on page 289.

2.4. Manually Paging-In Terrain in VR-Forces



VR-Forces pages in terrain based on the location of simulation objects and an algorithm that predicts where terrain will be needed. In most cases VR-Forces can predict the need for terrain and page it in without affecting the performance of the exercise. However there may be cases where it cannot, for example if there are many simulation objects over a wide area or many fast-moving simulation objects. Therefore, VR-Forces also allows you to specify areas where you want the terrain to be paged in.

Terrain page-in areas are created just like any other tactical graphic area. You can create them as the scenario progresses or you can include them in plans so that the terrain gets paged in when you know the scenario will need it. You can also attach a terrain page-in area to a simulation object to ensure that terrain in front of the simulation object gets paged in.

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



Terrain page-in areas only affect the back-end.

➤ **To create a terrain page-in area:**

1. Open the Tactical Graphics Palette.
2. In the Tags list, select Simulation Effect Object  .
3. In the Force list, select the force you want this area to be assigned to.
4. In the list of control objects, select Terrain Page-in Area.
5. Create the area.



3. Loading Data Using earth Files

This chapter provides an introduction to the syntax of earth files.

3.1. Earth Files 65
3.2. A Simple earth File 65
3.2.1 Including External Files in earth Files 67
3.3. Supported Drivers 68
3.3.1 Disabling Intersections for a Feature Geometry Layer 69
3.4. Loading Multiple DTED Files 69
3.5. Processing Streamed Data 69
3.6. Adding Feature Layers to an earth File 70
3.6.1 Configuring Point Features 72
3.6.2 Configuring Linear Features 73
3.6.3 Configuring Areal Features 74
3.6.4 Configuring Altitude Features 75
3.7. Extruded Buildings 75
3.8. Runway Offset to Avoid Z-Fighting 76
3.9. Specular Reflections on Water 77
3.10. Using Cut-in Sites for High Resolution Insets 77
3.10.1 Using the Boundary Generation Tool 78
3.11. Procedural Terrains 79
3.11.1 Exporting the Location of Procedural Vegetation 81
3.11.2 Configuring Procedural Trees for VR-Forces 82
3.12. Loading 3D Tiles 85
3.13. Options for earth Files 85
3.14. Referencing an Earth Gravitational Model (EGM) 86
3.15. Changing Visible Layers Based on the Date 86
3.16. Specifying Fading in earth Files 87
3.17. Differential Processing of Elements and Attributes 87
3.17.1 Differential Processing of model Elements 88
3.18. Indexing Elevation Layers 89

3.18.1 Configuring Elevation Layers for Indexing 89
3.18.2 Generating the Elevation Layer Indexes 90
3.19. Streaming Buoys and Beacons 90
3.19.1 Specifying Model Definitions for Streamed Buoys 92
3.19.2 Modeling Buoys Using Texture Atlases 94
3.19.3 Configuring Streaming Beacons 95
3.19.4 Configuring Streamed Navigation Lights 96
3.19.5 Configuring Seasonal Buoys and Beacons 97
3.19.6 Streaming Daymarks 97
3.19.7 Streaming Topmarks 98
3.19.8 Generating Signal Sequences 98
3.20. Land Use Data 98

3.1. Earth Files

An earth file is an XML format configuration file used to load terrain databases using osgEarth. The file format is documented at http://docs.osgearth.org/en/latest. The information in this chapter supplements the information on the osgEarth web site. In particular, it explains how to configure streaming features, for which there is no GUI in MAK ONE applications.



MAK ONE applications only support geocentric terrains in earth files.

3.2. A Simple earth File

This is an edited example of an earth file. It has one image layer and one elevation layer.

```
<map name="MAP Little Pond (Online)" type="geocentric" version="2">

    {% include options.terrain-rex-default.xml %}
    {% include elevation.worldwide.90m.online.xml %}
    {% include imagery.worldwide.online.xml %}
    {% include buildings.boston.online.xml %}

    <FeatureModel name="Little Pond Buildings" driver="feature_geom"
vrfsim:enabled="true">
        <feature_indexing enabled="false"/> <!-- include this with
every model section for VR-Vantage. If enabled feature_indexing causes
performance issues and crashes on Linux -->
        <OGRFeatures name="buildings" driver="ogr">
            <url>../../TerrainData/Terrain/LittlePond/Features/Buildin
gsP.shp</url>
        </OGRFeatures>
        <layout>
            <tile_size_factor>3</tile_size_factor>
        </layout>
        <styles>
            <style type="text/css">
                HousesVictorianMansard {
                model:              "../../ModelData/Structures/Houses
VictorianMansardClapboardGray.medf";
                model-heading:       0-[rotation];
                model-scale:        1;
                model-placement: vertex;
                altitude-clamping:   terrain;
                altitude-resolution: 0.00008583068847656250;
                altitude-offset:     0;
                }
                HousesColonialGarrison {
```

```
                model:              "../../ModelData/Structures/Houses
ColonialGarrisonClapboardGreen.medf";
                model-heading:       0-[rotation];
                model-scale:        1;
                model-placement: vertex;
                altitude-clamping:   terrain;
                altitude-resolution: 0.00008583068847656250;
                altitude-offset:     0;
                }
                ...
            </style>
            <selector class="HousesVictorianMansard">
                <query>
                    <expr>TYPE = 'HousesVictorianMansard'</expr>
                </query>
            </selector>
            <selector class="HousesColonialGarrison">
                <query>
                    <expr>TYPE = 'HousesColonialGarrison'</expr>
                </query>
            </selector>
            ...
        </styles>
        <clustering>false</clustering>
        <lighting>true</lighting>
    </FeatureModel>
...
</map>
```

Table 2 describes the XML elements in the file and their attributes.

Table 2. earth file elements and attributes

Element	Attribute	Description
<!-- -->		Comment block.
map		Root level element. All other elements must be in the map block.
	name	The name of the map.
	type	The projection. MAK ONE applications only support geocentric terrains, which can be specified as **geocentric** or **globe**.
	version	The VR-TheWorld Server version.

Table 2. earth file elements and attributes (continued)

Element	Attribute	Description
image		Specifies an image layer. Layer names are specific to the image type: • TMSImage • GDALImage • WMSImage • XYZImage • FeatureImage • CompositeImage.
	name	The name of the layer.
	url	The location of the files. This could be a web address or a local path.
elevation		Specifies an elevation layer. Names are specific to the type of layer. • TMSElevation • GDALElevation • MBTilesElevation • CompositeElevation.
	name	The name of the layer.
	url	The location of the files. This could be a web address or a local path.
FeatureModel		Base element for specifying a feature layer. For more information, see "3.6. Adding Feature Layers to an earth File" on page 70.
	name	Name of the model element. This can be any text.
	driver	Driver to use for creating models.
options		Specifies options that affect the entire map. For more information, see "3.13. Options for earth Files" on page 85.

3.2.1 Including External Files in earth Files

Each earth file can include external files. This can make management of complex terrains easier. The included files must be XML files. There are two ways to include external files: the `xi:include` element and use of earth file templates.

The following example shows how to use `xi:include`:

```
<map name="readymap.org" type="geocentric">
  <xi:include href="readymap_imagery.xml"/>
  <xi:include href="readymap_elevation.xml"/>
</map>
```

When you reference a document using `xi:include`, it can only contain a single XML node, for example:

```
<image>
  ...
</image>
```

Since using an `xi:include` element for each node that you want to import might require many such elements, you can include multiple nodes in a single file by using earth file templates, an alternative include format. Use the syntax `{% include` *`filename`* `%}`. This syntax is used in many of the earth files shipped with MAK ONE applications. The following lines are from *MAK Earth (online).earth*:

```
<!-- Hawaii area models -->
  {% include maritime-navigation.hawaii.xml %}
  {% include buildings.oahu.online.xml %}
  {% include airports.oahu.xml %}
```

The included XML files are located in the same directory.



In an earth file, if you comment out an include line that uses the `{% include` *`filename`* `%}` syntax, the included file must not have any comments or it will not be processed properly. This can result in terrain errors that are not easily diagnosed. To avoid this problem, if you want the earth file to ignore a line, instead of commenting it out, change **include** to **exclude**.

Files included with the `xi:include` syntax do not have this restriction.

3.3. Supported Drivers

Drivers are software components that allow you to load one or more types of terrain data. MAK has not tested all of the drivers supported by osgEarth. MAK ONE applications support the following drivers:

- tms
- gdal
- composite
- feature
- ogr
- feature_geom
- feature_custom
- wfs
- aglite

Other drivers supported by osgEarth may work. If you have questions about driver support, contact support at *https://www.mak.com/mak-one/support/portal*.

3.3.1 Disabling Intersections for a Feature Geometry Layer

In some cases, you may find that a feature geometry layer (`driver="feature_geom"`) negatively affects performance. This can occur because there are a number of text labels that do not need to be intersected at all. In those cases, you can disable intersections for the entire layer using the **intersectable** key field. For example:

```
<FeatureModel name = "Country-Names" driver="feature_geom">
    <intersectable>false</intersectable>
    <feature_indexing enabled="false"/>
    <OGRFeatures name=Country-Names" driver="ogr">
        <url>../../TerrainData/Terrain/World/Features/world.shp</url>
    </OGRFeatures>

    ...

</FeatureModel>
```

3.4. Loading Multiple DTED Files

You can load multiple DTED files from an earth file. To do so, use one of the following methods:

- Create a Composite Image Layer (desirable if there are a few images).
- Create a Virtual Dataset (desirable if there are not just a few images). To create a virtual dataset, you need the gdalbuildvrt program. For more information, go to: https://gdal.org/programs/gdalbuildvrt.html.
- Create an overview image (better for performance). To create an overview image, you need the gdaladdo program. For more information, go to: https://gdal.org/programs/gdaladdo.html.

3.5. Processing Streamed Data

MAK ONE applications have several software modules, which we generically call "processors," that search through streamed-in data looking for things like SpeedTree trees, objects that need effects textures attached, buoys, beacons, and so on. In order to do this efficiently, processors register feature names that they want to process. The registration data is in *./factory/importConfig/registrant_feature_list.csv*.

For example, in *registrant_feature_list.csv*, it states that the processor for SpeedTree trees, speedTreeManager, is mapped to **Vegetation**. In *groundcover.vegetation.oahu.online.xml*, SpeedTree trees are configured as follows:

```
<model name="model-point-trees-oahu"
           driver="feature_custom"
           cacheid="model-point-trees-oahu"
           vrfsim:enabled="true">
     <WFSFeatures name="Vegetation" driver="wfs">
           <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
           <typename>5</typename>
</WFSFeatures>
```

The feature name is Vegetation. This means that it will be processed by the speedTreeManager.

Similarly, the featureDataBroker, which is responsible for creating streamed buoys, beacons, and the lights and marks that might be attached to them, is mapped to **s57-data** and **dynamicFeatures**. (Processors can be mapped to more than one feature type.) Examples of using this processor (in *maritime-navigation.hawaii.xml*) are:

```
 <model name="oahu-LateralBuoysUS5HA52M" driver="feature_custom"
enabled="true">
      <OGRFeatures name="s57-data" driver="ogr">
```

and (in *airports.oahu.xml*):

```
 <model name="model-taxiway-lights-HNL"
            driver="feature_custom"
            cacheid="model-taxiway-lights-HNL">
      <OGRFeatures name="dynamicFeatures">
```

Lastly, the indirectGeometryManager, which is responsible for adding models that replace point features for the indirect rendering system, is mapped to buildings and other static streamed geometry. This allows those models to be rendered using the high performance indirect rendering system. An example of using this processor for a model section (in *buildings.oahu.online.xml*) is:

```
 <FeatureModel name="oahu-buildings"
            driver="feature_geom"
            vrfsim:enabled="true"
            node_caching="true"
            cacheid="model-buildings-oahu">
      <WFSFeatures name="buildings" driver="wfs">
```

registrant_feature_list.csv file is fully commented and provides additional information about this scheme.

3.6. Adding Feature Layers to an earth File

You can add these types of features to a terrain:

- Point features, such as trees (point trees), buoys, and building
- Linear features, such as roads and waterways.
- Areal features, such as forests (areal trees), bodies of water, and extruded buildings.

You can add the XML for feature layers to an earth file.

Feature layers are added to an earth file using the **model** element. The example below shows the hierarchical relationships of a **model** element and its subelements. Table 3 describes the **model** element, its attributes, and some of its subelements. (A particular **model** element might not have all of the subelements described.)

```
 <model>
     <features>
         <url></url>
```

```
    </features>
    <layout>
        <tile_size_factor></tile_size_factor>
        <level></level>
    </layout>
    <styles>
        <library>
            <url></url>
        </library>
        <style></style>
        <selector></selector>
    </styles>
    <lighting></lighting>
</model>
```

Table 3. model element attributes

Element	Attribute	Description
model		Base element for specifying a feature layer.
	name	Name of the model element. This can be any text.
	driver	Driver to use for creating models.
	vrfsim:enabled	Optional. For use in terrains used by VR-Forces if the simulation engine needs a different value for an attribute than that used by the front-end. If present, it is ignored by VR-Vantage and the VR-Forces front-end. For details, please see "3.17.1 Differential Processing of model Elements" on page 88.
The **features** and **layout** elements specify how to get and load the feature data. The **features** block is mandatory. **layout** is optional.		
features		Specifies source of data: • OGRFeatures • TFSFeatures • XYZFeatures • WFSFeatures.
	name	Name of the element.
url		Location of the feature data.
layout		Specifies how to load the data.
tile_size_ factor		The data specified in the features element gets cut into tiles. This attribute specifies how aggressively to tile it.
level	name	The level specifies the level of detail (LOD) beyond which not to display the features.

Table 3. model element attributes (continued)

Element	Attribute	Description
	max_range	Distance beyond which the feature is not displayed.
selector	class	Specifies the type of feature.
styles		The **styles** element and sub-elements specify how to render the scene. This element is mandatory.
library		A texture library.
	name	Name of the texture library.
	url	The location of the library.
style		A specific style. The name, which precedes the curly brace ({), can be any text you want. For information about the attributes that you can specify in a style, please see http://docs.osgearth.org/en/-latest/references/symbology.html.
lighting		Specifies whether or not to enable lighting. Usually true.

You can add buildings to a terrain using the **model** element or the **buildings** element. They are functionally similar, but the **buildings** element has additional attributes that are specific to buildings. For an example of the **buildings** element, see *./SharedData /16/latest/TerrainData/TerrainConfiguration/buildings.worldwide.osm.base.online.xml*. (The VR-Forces simulation engine does not process some types of buildings when they use this element. If you want to use it with VR-Forces, contact support at https://www.mak.com/mak-one/support/portal.)



When you load a file using the OGR driver (specified in the **feature** element), the layer name is the name of the file (minus the extension). Other files (like S-57) can have multiple layers, and the names are decided by a format-specific algorithm. For example, an S-57 file has an enumerated list of layer names.

3.6.1 Configuring Point Features

Point features, such as trees, use the **model** element. The following code, from *groundcover.vegetation.oahu.online.xml*, adds point trees:

```
<!-- streaming point trees -->
<model name="model-point-trees-alamoana"
        driver="feature_custom"
        cacheid="model-point-trees-alamoana"
        vrfsim:enabled="true">
    <feature_indexing enabled="false"/>
    <!-- include this with every model section for VR-Vantage. If
enabled
     feature_indexing causes performance issues and crashes on Linux -
->
```

```
    <!-- include this with every model section for VR-Vantage. If
enabled
     feature_indexing causes performance issues and crashes on Linux -
->
    <clustering>true</clustering>
    <lighting>true</lighting>
    <OGRFeatures name="Vegetation" driver="ogr">
     <url>../../TerrainData/Terrain/Hawaii/Features/AlaMoanaVegetation
P.shp</url>
        <typename>5</typename>
    </OGRFeatures>
    <layout>
        <tile_size_factor>3</tile_size_factor>
    </layout>
    <styles>
        <style type="text/css">
            Vegetation {
            model:              [veg];
            model-scale:        1;
            model-placement: vertex;
            altitude-clamping:   terrain;
            altitude-resolution: 0.00008583068847656250;
            altitude-offset:     0;
        }</style>
    </styles>
</model>
```

3.6.2 Configuring Linear Features

Linear features, such as roads, rivers, or railways, use the **model** element. The following code, from *roads.hawaii.online.xml*, configures roads:

```
<FeatureModel name="oahu-roads (vrf-sim)" driver="feature_geom"
overlay="true" enabled="false" node_caching="true" cacheid="Oahu_
roads">
    <feature_indexing enabled="false"/>
    <!-- include this with every model section for VR-Vantage. If
enabled feature_indexing causes performance issues and crashes on
Linux -->
    <WFSFeatures name="Roads" driver="wfs">
        <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
        <typename>55</typename>
    </WFSFeatures>
    <feature_name>[name]</feature_name>
    <styles>
        <style type="text/css">
        Oahu_Roads {
            stroke: #ff0000;
            stroke-width: 3.0;
```

```
            altitude-offset: 10;
            altitude-clamping: terrain-drape;
        }</style>
    </styles>
</FeatureModel>
```

3.6.3 Configuring Areal Features

Areal features, such as forests, use the **model** element. The following code, from *groundcover.vegetation.oahu.online.xml*, configures a forest with three types of trees:

```
<model name="model-forests-oahu"
       driver="feature_custom"
       cacheid="model-forests-oahu"
       vrfsim:enabled="true">
    <feature_indexing enabled="false"/>
    <!-- include this with every model section for VR-Vantage. If
enabled
 feature_indexing causes performance issues and crashes on Linux -->
    <clustering>true</clustering>
    <lighting>true</lighting>
    <WFSFeatures name="Vegetation" driver="wfs">
        <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
        <typename>7</typename>
        <disable_tiling>true</disable_tiling>
    </WFSFeatures>
    <layout>
        <tile_size_factor>3</tile_size_factor>
        <crop_features>true</crop_features>
        <level max_range="8000" class="trees-1"> </level>
        <level max_range="2000" class="trees-2"> </level>
        <level max_range="1000" class="trees-3"> </level>
    /layout>
    <styles>
        <style type="text/css">
        trees-1 {
        model:              "TreesNorwaySpruceST";
        model-placement:    random;
        model-density:      800;
        model-scale:        1;
        altitude-clamping:   terrain;
        altitude-resolution: 0.0001;
        model-random-seed:  1;
        }
        trees-2 {
        model:              "TreesColoradoBlueSpruceST";
        model-placement:    random;
        model-density:      800;
```

```
        model-scale:        1;
        altitude-clamping:   terrain;
        altitude-resolution: 0.0001;
        model-random-seed:  2;
        }
        trees-3 {
        model:              "TreesAmericanElmST";
        model-placement:    random;
        model-density:      800;
        model-scale:        1;
        altitude-clamping:   terrain;
        altitude-resolution: 0.0001;
        model-random-seed:  3;
        }</style>
    </styles>
</model>
```

3.6.4 Configuring Altitude Features

You can use a vertical datum to specify the altitude for mask and feature layers, in addition to elevation layers. For example:

```
    <profile srs="wgs84" vdatum="egm96"/>
```

3.7. Extruded Buildings

If you are simulating in an urban environment, you want your terrain to have realistic 3D buildings. However, modeling hundreds or thousands of buildings by hand would be an arduous and expensive task. osgEarth can extrude buildings from areal features and apply textures to them to create a realistic environment with relatively little work. Publicly available GIS data that has footprint and height data can provide the input.

The following code excerpt (from *buildings.oahu.online.xml*) adds extruded buildings to the terrain:

```
<FeatureModel name="oahu-buildings"
        vrfsim:enabled="true"
        node_caching="true"
        cacheid="model-buildings-oahu">
    <feature_indexing enabled="false"/>
    <!-- include this with every model section for VR-Vantage. If
enabled
 feature_indexing causes performance issues and crashes on Linux -->
    <cache_policy usage="read_write"/>
    <lighting>true</lighting>
    <cluster_culling>false</cluster_culling>
    <WFSFeatures name="buildings" driver="wfs">
        <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
        <typename>105</typename>
```

```
            <vrfsim:typename>106</vrfsim:typename>
        </WFSFeatures>
        <layout>
            <tile_size_factor>3</tile_size_factor>
        </layout>
        <styles>
            <library name="us_resources">
                <url>./osgEarthCatalogs/default.buildings-style-library-
    atlas.osgb.xml</url>
            </library>
            <style type="text/css">
            buildings {
                extrusion-height:      [hgt2d];
                extrusion-flatten:     true;
                extrusion-wall-style:  building-wall;
                extrusion-roof-style:  building-rooftop;
                extrusion-random-seed: 1;
                altitude-clamping:     terrain;
            }
            building-wall {
                skin-library: us_resources;
                skin-tags:    building;
                skin-random-seed: 1;
                fill:         #ffffff;
            }
            tanks {
                extrusion-height:      [hgt2d];
                extrusion-flatten:     true;
                extrusion-wall-style:  tanks-wall;
                extrusion-roof-style:  tanks-rooftop;
                extrusion-random-seed: 1;
                altitude-clamping:     terrain;
            }
            ...
            }</style>
            <selector class_expr="getStyleClass()"/>
        </styles>
    </FeatureModel>
```

For descriptions of the elements in a **model** block, please see Table 3 on page 71.

3.8. Runway Offset to Avoid Z-Fighting

You can set a runway offset elevation to avoid Z-fighting. You simply need to specify a non-zero offset. In your *airports.worldwide.xml* file, set the following:

```
    <FeatureElevation name="elevation-worldwide-airports"
            min_level="10"
            max_data_level="16"
```

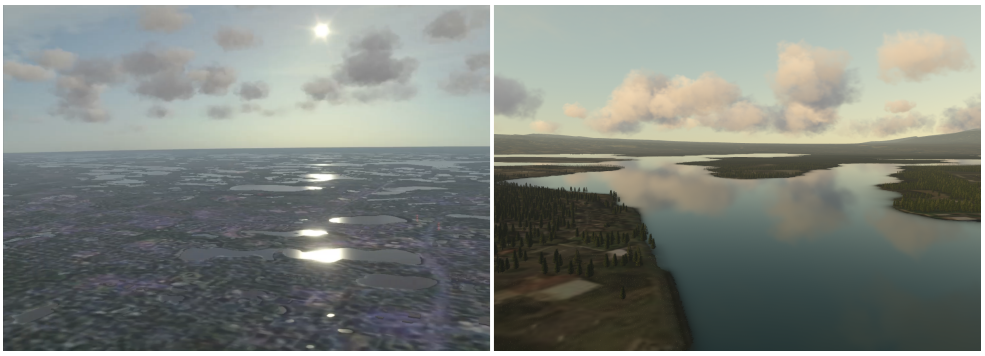
```
vrfsim:max_level="16"
enabled="true"
offset="0.005"
attr="ELEVATION">
```

3.9. Specular Reflections on Water

For inland bodies of water, you can add specular map textures to osgEarth image layers based on feature data. This gives water a more realistic appearance by specifying where the inland bodies of water are, making them more reflective than the rest of the terrain (Figure 19). The following code enables specular reflections:

```
<effectmap effect_map_type="SPECULAR_MAP">
   {Image Layer}
</effectmap>
```

Figure 19. Specular reflections on water



3.10. Using Cut-in Sites for High Resolution Insets

Users of terrain servers, such as VR-TheWorld Server, often want to add high resolution sites of particular interest while using the lower resolution terrain provided by the terrain server for the rest of the world. You can add a high resolution terrain patch to a terrain server, but it overlays the terrain provided by the server and this can cause unexpected visual problems. Ideally, you should cut out the relevant areas of terrain provided by the server and add your high resolution terrain to the cut out area. For example, the terrain *MAK Earth (online).mtf* has two terrain patches, *MAK Earth (online).earth*, which provides the base terrain from VR-TheWorld Server, and *./SharedData/16/latest/Terrain/Hawaii/Geometry/OpenFlightAlaMoana /AlaMoanaGeocentricProxy.osg*, which is a locally stored high resolution inset.

The **mask** element lets you cover the base terrain data with your hand-modeled data. A mask element in an earth file looks similar to the following:

```
<FeatureMask name="HonoluluRunway26L-8R" min_level="6">
    <OGRFeatures>
```

```
        <geometry>POLYGON((-157.91489143 21.32914026 3.016, -
157.91525638 21.32914026 3.016, -157.91666238 21.32912541 3.016, -
157.91697822 21.32934847 3.016, -157.91709263 21.32968744 3.016, -
157.91709263 21.32968744 7.016, -157.91709263 21.32968744 11.016, -
157.91709263 21.32968744 15.016, -157.91710262 21.32968706 15.016, -
157.91697822 21.32934847 16.016, -157.91709263 21.32968744 16.016, -
157.91710262 21.32968706 15.016))</geometry>
    </OGRFeatures>
</FeatureMask>
```

MAK ONE applications provide a tool for generating masks. The osgEarth Boundary Generation Tool is an offline tool that can calculate the geometry of an OpenFlight file and send it to a file. To cut out a section of terrain from the terrain server, you add a **mask** element to the earth file and use the output from the boundary tool for the data in the geometry element.

Table 4 describes the **mask** element and its sub-elements.

Table 4. mask element

Element	Attribute	Description
mask		Element for cutting in geometry.
	name	Name of the mask.
	driver	The driver to use.
features		List of features in the mask.
	driver	The driver to use. OGR reads raw vector feature data from shapefiles and other formats.
geometry		Specifies the 3D boundaries of the site. You must know this information.
	geometry_url	The location of the file with the feature data.

3.10.1 Using the Boundary Generation Tool

The Boundary Generation Tool takes an OpenFlight file and calculates the bounding geometry of the file. It returns the coordinates in a text file. The general process of cutting in a site is as follows:

1. Run the Boundary Generation Tool against an OpenFlight file.
2. Insert the output of the boundary generation tool in a **mask** element in your earth file. (For details about **mask** elements, see "3.10. Using Cut-in Sites for High Resolution Insets" on the previous page.)
3. Load the terrain.
4. Add the FLT file as a terrain patch.
5. Save the terrain.

Running the Boundary Generation Tool

The output from the Boundary Generation Tool, in *boundary.txt* or the output file you specify, looks similar to the following:

```
  POLYGON((-157.91489143 21.32914026 3.016, -157.91525638 21.32914026
3.016, -157.91666238 21.32912541 3.016, -157.91697822 21.32934847
3.016, -157.91709263 21.32968744 3.016, -157.91709263 21.32968744
7.016, -157.91709263 21.32968744 11.016, -157.91709263 21.32968744
15.016, -157.91710262 21.32968706 15.016, -157.91697822 21.32934847
16.016, -157.91709263 21.32968744 16.016, -157.91710262 21.32968706
15.016))
```

The first two vertices are the latitude and longitude. The third vertex is the height, in meters. The remainder of the vertices map the outline of the geometry.

➤ **To run the boundary generation tool:**

1. Open a console window.
2. Change directory to *./bin64*.
3. Run the tool:

   ```
   osgearth_boundarygen [--out file_name --no-geocentric --convex_
   hull --verbose --view --precision precision --tolerance tolerance]
   model_file
   ```

Table 5 describes the options.

Table 5. osgEarth Boundary Generation Tool command-line options

Option	Description
`--convex_hull`	Calculates a convex hull instead of a full boundary.
`model_file`	The input model file.
`--no-geocentric`	Skips geocentric reprojection. Use for flat databases.
`--out` *`file_name`*	Specifies the output file for boundary geometry. Default: *boundary.txt*.
`--precision` *`precision`*	Specifies the number of significant digits in the output coordinates. Default: 12.
`--tolerance` *`tolerance`*	Specifies the tolerance for combining similar vertices along a boundary. Default: 0.005.
`--verbose`	Prints progress messages to the console.
`--view`	Shows the result in a 3D window.

3.11. Procedural Terrains

Procedural terrains use land use data to generate geotypical vegetation and ground cover. A procedural terrain provides a visually pleasing terrain, particularly when zoomed in, without the overhead of high-resolution imagery. Procedural imagery can be combined with other types of terrain data.

Figure 20 illustrates a portion of Sanibel Island, Florida, USA using the MAK Earth (online) terrain. This terrain uses relatively low resolution imagery. Figure 21 illustrates the same location using the Procedural Earth (online) terrain. Notice that the procedurally generated vegetation makes this a much more attractive terrain. Figure 22 illustrates the same location with OpenStreetMap data included in the terrain. Figure 23 shows a closeup of the terrain.

Figure 20. Sanibel Island, MAK Earth (online)



Figure 21. Sanibel Island, Procedural Earth (online)

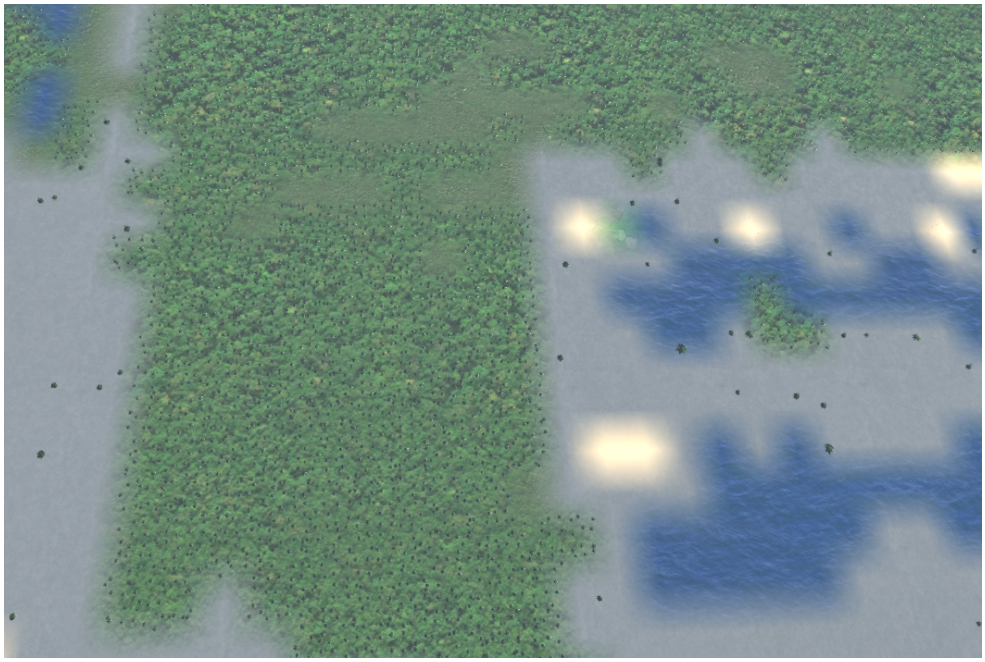


Figure 22. Sanibel Island, ProceduralEarth - OpenStreetMap (online)

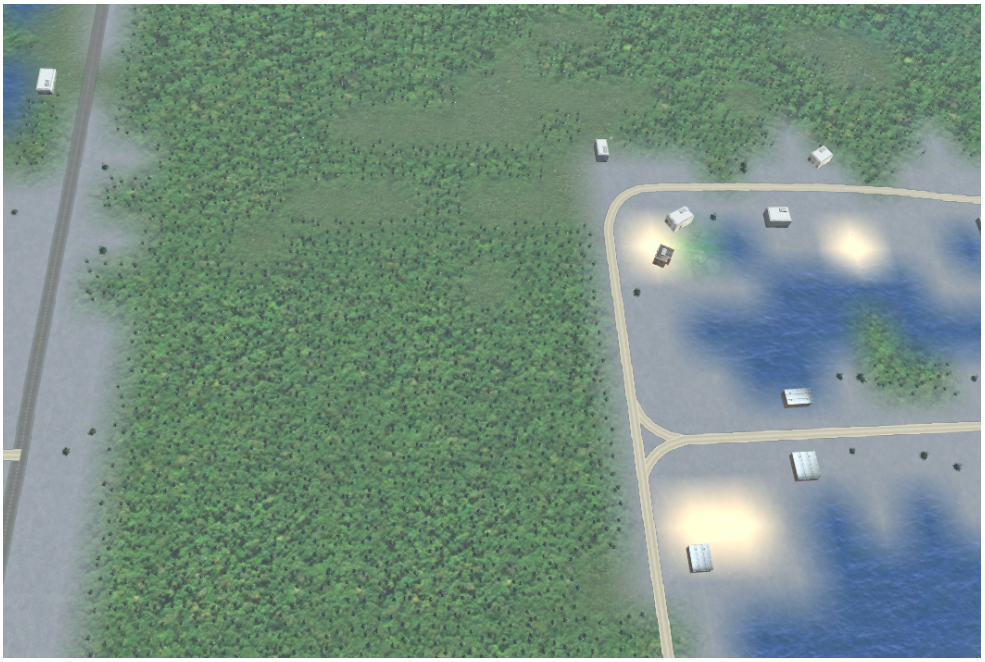
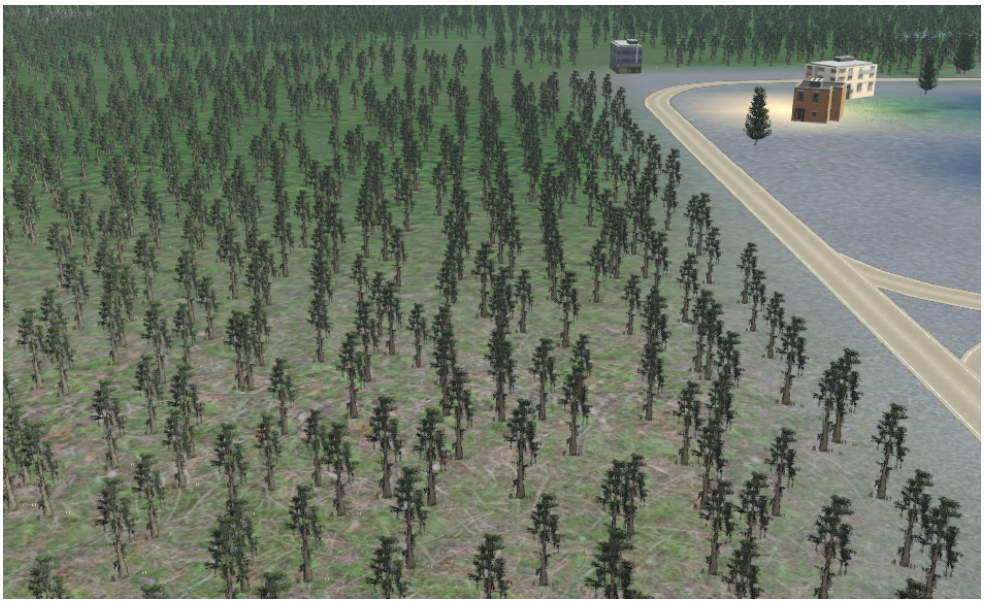


Figure 23. Sanibel Island closeup, ProceduralEarth - OpenStreetMap (online)



3.11.1 Exporting the Location of Procedural Vegetation

Simulation applications, such as VR-Forces, need to know the location, height, and width of objects in the environment, such as trees, bushes, and buildings. In hand-modeled terrains, these objects are fixed and when an application loads the terrain, it knows where they are. However procedural terrains create vegetation dynamically using an algorithm. MAK ONE applications use internal APIs to ensure synchronization between the simulation engine and the procedurally generated terrain. However, non-MAK ONE applications do not have access to these APIs.

To allow interoperability between MAK ONE and non-MAK ONE applications when using procedurally generated terrains, VR-Forces includes a tool, osgearth_exportgroundcover, that can export the location of procedurally generated vegetation to a shapefile. Non-MAK ONE applications can load the shapefile and know where the vegetation is located.

osgearth_exportgroundcover is a command-line tool, located in *./bin64*. Its usage is as follows:

```
osgearth_exportgroundcover earth_file --layer layername --extents
swlong swlat nelong nelat --out shapefile.shp --include-billboard-
property property
```

where:

- *earth_file* is the earth file you are processing.
- *layername* is the name of the GroundCover layer.
- *swlong swlat nelong nelat* are the extents of the terrain in degrees.
- *shapefile.shp* is the output shapefile.
- *include-billboard-property* is a billboard property name as an attribute.

The shapefile will contain point features, one per instance of GroundCover. Each feature will have the following attributes:

- height, in meters.
- width, in meters,
- elevation, in meters, in the same vertical datum as the map (usually WGS84 geodetic.)

The Spatial Reference System of the feature geometry will be the same as that of the earth file. By default, this is WGS84.

If there are any additional earth file properties (on the <billboard> symbol in your GroundCover definition) that you want included as feature attributes, you can use the `--include-billboard-property` options to name them.

3.11.2 Configuring Procedural Trees for VR-Forces



VR-Forces can support procedural trees. This allows the simulation engine to understand the location of the trees so that they can affect line-of-sight and obstacle avoidance.

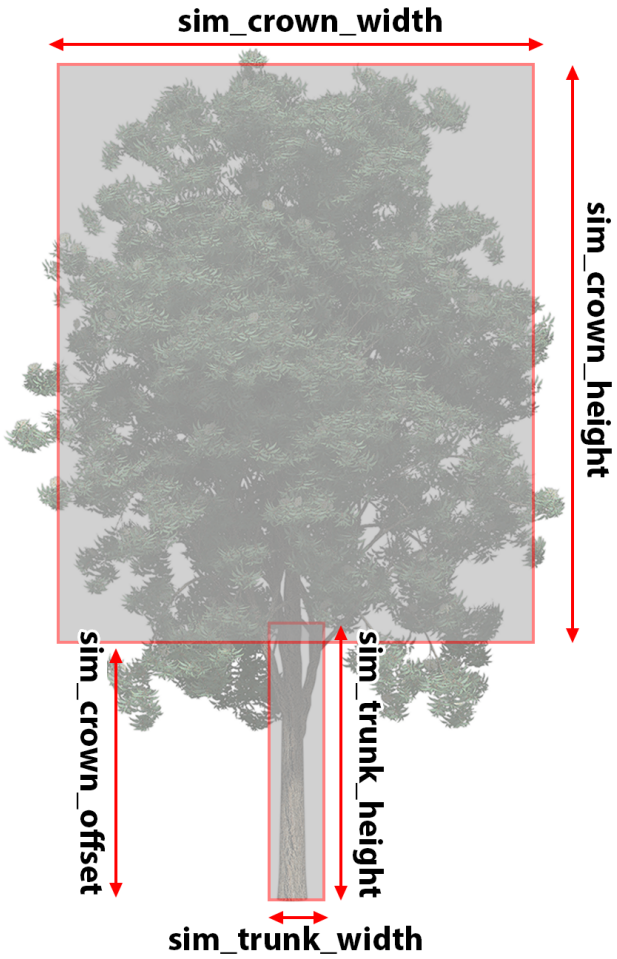
To enable procedural trees, in the `GroundCover` element you must set vrfsim:enabled to true:

```
  <GroundCover name="Trees, Medium" land_cover_layer="LandCover"
vrfsim:enabled="true">
```

In the `billboard` element for each tree that you want to configure, you set one mandatory and several optional attributes:

- **id**: a unique integer for each billboard (mandatory).
- **sim_trunk_height**: The tree trunk height in meters (optional).
- **sim_trunk_width**: The tree trunk width in meters (optional).
- **sim_crown_offset**: The height at which the crown starts in meters (optional).
- **sim_crown_height**: The tree crown height in meters (optional).
- **sim_crown_width**: The tree crown width in meters (optional).

Figure 24. Billboard attributes

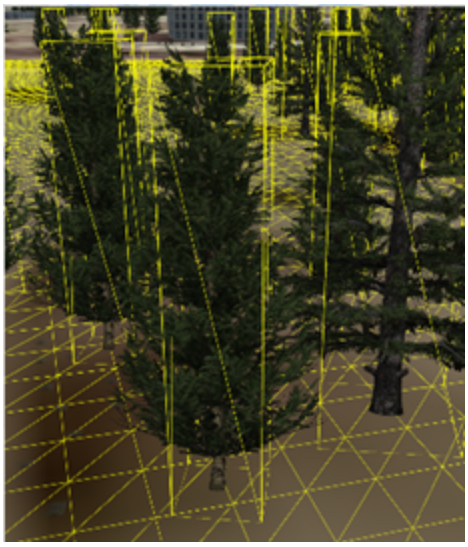


Although most of the attributes are optional, if you do not set any of them, the bounding box around the tree will be too large to be useful. At a minimum, you should set the **sim_crown_height** and **sim_crown_width**, for example:

```
  <billboard id="511" url="../../ModelData/Vegetation/Trees/FraserFir_
2.7m_png.meif" top_url="../../ModelData/Vegetation/Trees/FraserFir_
2.7mTop_png.meif" width="12" height="12"
selection_weight="2"
size_variation="0.25"
sim_crown_height="12.0"
sim_crown_width="2.5"/>
```

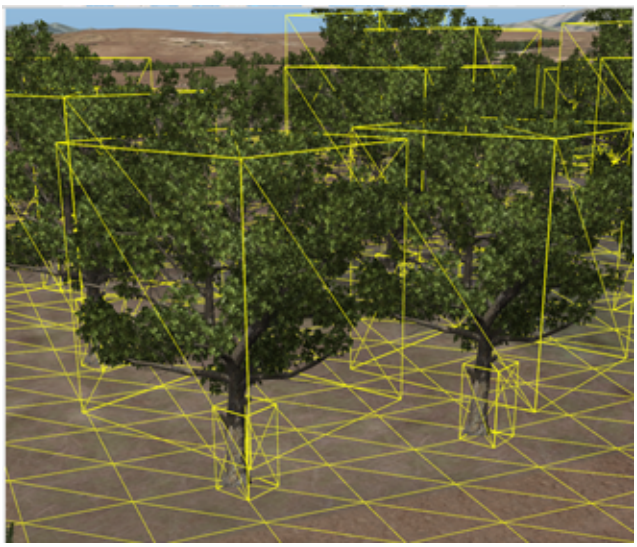
With these two optional parameters, VR-Forces creates one bounding box for the entire tree using the height and width values (Figure 25).

Figure 25. Procedural tree, one bounding box



If you use all of the attributes, VR-Forces creates separate bounding boxes for the trunk and the crown (Figure 26).

Figure 26. Procedural tree, two bounding boxes



The following code shows a procedural tree configured with all attributes:

```
  <billboard id="461"
url="../../ModelData/Vegetation/Trees/AmericanSycamore_9.13m_png.meif"
top_url="../../ModelData/Vegetation/Trees/AmericanSycamore_9.13mTop_
png.meif"
  width="9.13"
  height="9.13"
  sim_trunk_height="3.0"
  sim_trunk_width="1.5"
```

```
sim_crown_offset="3.0"
sim_crown_height="5.5"
sim_crown_width="5.0"/>
```

Examples of configuring procedural trees are in *groundcover.vegetation.worldwide.xml* and *groundcover.vegetation.wordwide.Combined.online.xml*.

3.12. Loading 3D Tiles

VR-Vantage applications can load 3D tiles as a streaming data source. Your earth file references the data source in a **ThreeDTiles** element and adds a mask for the area covered by the tiles. VR-Vantage, VR-Forces, and VR-Engage include an example terrain, *Test-3DTiles.mtf* (and *Test-3DTiles.earth*), that shows how to load 3D tiles.

```
<!-- 3D Tiles dataset of Ft Hood TX -->
    <ThreeDTiles name="Ft Hood, TX" vrfsim:geometricError=2.0>
        <url>http://pelican-
public.s3.amazonaws.com/3dtiles/FtHood/data/
            tileset.json</url>
    </ThreeDTiles>
    <!-- Cut-out taken directly from the Ft Hood bounding volume -->
    <FeatureMask>
        <OGRFeatures>
            <geometry>
                POLYGON((-97.9500006314 31.311573605 249.28837631, -
97.9500006149 31.2032081428 287.466499442, -97.9500003817
31.1579549246 328.979375438, -97.9500002948 31.1474227897
313.274364261, . . . 31.3922786588 271.912850835))
            </geometry>
        </OGRFeatures>
    </FeatureMask>
```



The 3D tiles format is evolving. Therefore, support for it may change over time.

3.13. Options for earth Files

An earth file can have **option** elements that affect the entire map. Table 6 describes the option elements.

Table 6. option element

Element	Attribute	Description
lighting		Enables or disables lighting for the entire map. This is usually true.
elevation_ interpolation		Specifies how OSG interprets elevation between grid posts. If you are using VR-Forces or visualizing a VR-Forces simulation, set this to triangulate to ensure correlation.
cache		Specifies caching for terrain data.
	type	The type of caching to use.
	path	The location of the data cache.

3.14. Referencing an Earth Gravitational Model (EGM)

In MAK ONE applications, earth files can support loading elevation data that references an earth gravitational model (EGM). To do so, add the following line to the **elevation** or **heightfield** element:

```
<vdatum>egm_model</vdatum>
```

where *egm_model* can be egm96, egm84, or egm2008. For example:

```
<elevation driver="gdal">
   <url>file.dt0</url>
   <vdatum>egm96</vdatum>
</elevation>
```

3.15. Changing Visible Layers Based on the Date

You can change the layers that get displayed based on the current date of the simulation. For example, you might want to display a lush foliage during the rainy season and a dry foliage during the dry season. Or you might want to show a snow covered terrain in winter and a different terrain in summer. To do this, insert **start_ date** and **end_date** attributes in any **image**, **model**, or **splat_groundcover** element, for example:

```
<splat_groundcover>
  <start_date day="21" month="12"/>
  <end_date day="19" month="3"/>
...
<splat_groundcover>
```

The day is the day of the month. The month is the number of the month, where January is 1 and December is 12.

3.16. Specifying Fading in earth Files

As the eyepoint reaches a new LOD for the terrain, terrain layers can pop into view, which is disconcerting. You can specify that layers fade into and out of view to avoid this problem.

You can set fading for all layers and for specific model and building layers.

To specify fading for all layers, add the **feature_fading** element to the options element of the map level, as follows:

```
<options>
   <feature_fading duration  = "0.3"
       max_range = "6000"
       attenuation_distance = "1000" />
</options>
```

where:

- **duration** is the time it takes to fade in on initial load or page in.
- **max_range** is the distance that the model is completely transparent.
- **attenuation_distance** is the distance over which the model goes from completely opaque to completely transparent.

You can set **fading** for indirectly rendered objects in **model** and **building** layers as follows:

```
<fading duration  = "0.3"
   max_range = "2500"
   attenuation_distance = "500" />
```

All other indirect objects have fading enabled by default. Only the duration is set. The default is .3 seconds.

Non-indirect feature layers work with the osgEarth method of fading for **model** layers, but do not respect the **map** level **feature_fading** option. **effectmap** layers also fade. They do not use the global **feature_fading** option and only the **max_range** and **attenuation_distance** parameters apply to **effectmap** layers. Non-indirect **building** layers do not fade.

3.17. Differential Processing of Elements and Attributes



VR-Vantage shares earth files with VR-Forces. The differential processing described in this section would not be used by VR-Vantage. It ignores **vrfsim:** syntax. This section is included to help you understand why the earth files provided with VR-Vantage are written this way.



There are times when VR-Forces users want the front-end and back-end to process terrain differently. You can specify different values for elements and attributes with the **vrfsim:** syntax. Anyplace that VR-Forces sees **vrfsim:** when it processes an earth file, it uses that value for the back-end. For example, if you want different values for the **name** attribute in a model element, specify it as follows:

```
<model name="front_end_name" vrfsim:name="back_end_name">
```

To specify differential elements, the element names must be the same and be the only two elements of that name in the scope of the earth file or an element block. For example, the following code would mean use the **<trees>** element in the front-end and the **<vrfsim:trees>** element in the back-end:

```
<trees>
   ...
</trees>

<vrfsim:trees>
   ...
</vrfsim:trees>
```



The earth files included with VR-Vantage and VR-Engage also use this syntax because they share terrains with VR-Forces. However, they do not process the VR-Forces-specific syntax.

3.17.1 Differential Processing of model Elements

You can enable or disable differential processing within **model** elements. By default, all **model** elements except those that use the **simple** driver have differential processing disabled.

To enable differential processing, add **vrfsim:enabled="true"** to the **model** element.

The following example of differential feature processing is from *buildings.oahu.online.xml*. It is the section used to extrude buildings. **vrfsim:enabled** is set to true. The value for the **typename** attribute is applied to front-end processing and the value for the **vrfsim:typename** attribute is applied to back-end processing.

```
<FeatureModel name="oahu-buildings"
       driver="feature_geom"
       vrfsim:enabled="true"
       node_caching="true"
       cacheid="model-buildings-oahu">
   <WFSFeatures name="buildings" driver="wfs">
       <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
       <typename>105</typename>
       <vrfsim:typename>106</vrfsim:typename>
   </WFSFeatures>
```

If a model element uses the **simple** driver, it uses the value of the **enabled** attribute to determine whether or not to use differential processing (in other words, there is no default setting). If you want to explicitly enable or disable differential processing, add **vrfsim:enabled** with the appropriate value to the **model** element. Then add an attribute element in which the attribute name is prefaced by **vrfsim:** as in the example in this section.

3.18. Indexing Elevation Layers



In VR-Forces, calculating terrain intersections over large distances (hundreds of miles), such as in air-to-air and space-to-ground scenarios can result in loading many terrain tiles, which may reduce performance to unacceptable levels. You can reduce the likelihood of this problem by indexing elevation layers and configuring earth files to use the indexes.

When you index elevation layers, the indexes allow the back-end to know the minimum and maximum altitudes of the elevation in each tile so that terrain intersects that are above or below the tile do not cause the tile to load (since they would not intersect with it anyway). The end result is that only the portions of intersections that are near the terrain cause terrain to load and the rest of the intersection query can be efficiently ignored.

3.18.1 Configuring Elevation Layers for Indexing

To configure an index for an elevation layer, you add a **<mak_terrain_index>** element to the **<elevation>** element in the earth file. The following example shows an elevation layer configured for indexing:

```
<elevation
   name="Worldwide 90m Elevation with Bathymetry Collection"
   driver="tms">
   <cache_policy usage="none"/>
   <url>http://vr-theworld.com/vr-theworld/tiles/1.0.0/122/</url>
   <mak_terrain_index name="worldwide90m_bathymetry">
       <url>path/to/worldwide90m_bathymetry.index.bin</url>
       <max_lod>7</max_lod>
       <max_tolerance>500.0</max_tolerance>
       <min_tolerance>-500.0</min_tolerance>
   </mak_terrain_index>
</elevation>
```

If the layer does not require any non-default configuration you do not need to add any configuration to the earth file.

Indexing is done for each leaf elevation layer. If you have a composite elevation layer you must configure an index for each individual elevation layer that makes up the composite, not for the composite as a whole.

If the layer is local (that is, the URL is a local file, not a streaming URL) then the name of the index file will be the same as the file name in the URL, with the extension *.index.bin* added to the end. It is created in the same directory as the local file.

If the layer is not local, then a file name is generated based on the URL and it is put in *./userData/terrainIndices*.



The index files provided with MAK terrains are in the appropriate data directory, not in *./userData/terrainIndices*.

An elevation layer can have multiple indexes. For example, one index may be a high resolution index around a specific site, while another one may be a lower resolution index that covers the whole terrain. To configure multiple indexes for a layer, you add multiple **<mak_terrain_index>** elements.

The attributes of a **<mak_terrain_index>** element are as follows:

- **url**. The location to put the index file if you do not want to put it in the default location. The file must end with *.index.bin* or *.index.txt*. (Text format is human readable, but takes longer to load.)
- **name**. Label for the index layer. This is used so that you can specify specific layers to generate with the terrainIndexTool.
- **max_lod**. The maximum LOD to put in the index. This allows you to generate a lower resolution index for a terrain that is too large to generate the full index, or for which the generated index would take too long to load.
- **max_tolerance**. A value to add to the max elevation value when reading in this index. If an index is generated with a **max_lod** that is less than the actual maximum LOD of the terrain, then the elevation values in the index will be based off of lower resolution elevation data than what will actually be used in the simulation. This means they will not necessarily correlate. This parameter lets allows you to adjust these values.
- **min_tolerance**. Similar to **max_tolerance**, this value is added to the minimum elevation value when reading the index. It should be negative. The minimum altitudes are not particularly important since they really only come into play when something is below the terrain.

There is no good way to compute good values for **max_lod** and the tolerances. They are generally arrived at by trial and error testing. **max_lod** should usually be set as large as it can and still be able to generate and load an index in a reasonable amount of time. The tolerances should be set as small as possible. Setting them too small can result in terrain not being loaded when it should.

3.18.2 Generating the Elevation Layer Indexes

You generate indexes for elevation layers with the terrainIndexTool. The tool reads the earth file specified in a terrain. It generates indexes for the elevation layers in the earth file. The syntax is as follows:

```
terrainIndexTool --terrain terrain [--name name]
```

where:

- `--terrain` specifies the name of the terrain file (*.mtf*) to process.
- *`--name`* specifies a specific elevation layer to process. You can use *`--name`* more than one time on a command line. If you omit this parameter, all layers in the terrain file are processed.

3.19. Streaming Buoys and Beacons

MAK ONE applications can display models of buoys and beacons specified in shapefiles and S-57 data, which is streamed in using earth files.

Streamed features (including buoys) are organized into layers. In the earth file, the names in the **styles** block specify the layer the streamed features will belong to. By default, MAK ONE applications define seven layers for buoys. They correspond to the different buoy layers in an S-57 file. These layers are named LateralBuoys, SafeWaterBuoys, ISDBuoys, RegulatoryBuoys, InstallationBuoys, CardinalBuoys, and SpecialPurposeBuoys.

The configuration file *./appData/importConfig/s57_layer_list.csv* specifies which style names in an earth file correspond to recognized streamed feature types. The file has two lines. The first starts with the keyword “Buoys” and is followed by a comma-separated list of all the style names that will be recognized (and streamed in) as buoys. The second line starts with the keyword “Beacons” and is followed by the list of recognized beacon styles. They can be anything you like, but need to correspond with whatever style names you use in the earth file.

The following is an example of a **style** block for a buoy (Special Purpose Buoys):

```
<model name="oahu-SpecialPurposeBuoysUS5HA52M" driver="feature_custom"
enabled="true">
    <OGRFeatures name="s57-data" driver="ogr">
        <url>../../TerrainData/Terrain/Hawaii/Features/US5HA52M/BOYSPP
.shp</url>
    </OGRFeatures>
    <layout>
     ...
    </layout>
    <styles>
        <style type="text/css">
    SpecialPurposeBuoys {
        model:              "Buoys[BOYSHP]"
        + ":BOYSHP=" + [BOYSHP]
        + ":COLOUR=" + [COLOUR]
        + ":COLPAT=" + [COLPAT]
        + ":PERSTA=" + [PERSTA]
        + ":PEREND=" + [PEREND]
        + ":CATSPM=" + [CATSPM]
        + ":OBJNAM=" + [OBJNAM] ;
        model-scale:        1;
        model-placement: vertex;
        altitude-clamping:   terrain;
        altitude-resolution: 0.00008583068847656250;
        altitude-offset:     0;
        }
        </style>
        <selector class_expr="getStyleClass()"/>
    </styles>
```

3.19.1 Specifying Model Definitions for Streamed Buoys

Within each style block, you specify the model definition to use for a streamed buoy or beacon. Since buoys and beacons have different styles and within each style have different colors and markings, MAK ONE applications can map the S-57 data to buoy and beacon models in the following ways:

- Specify a model definition for each type of buoy. Buoy color and numbering is the same for all buoys.
- Generate a model definition name using a pattern and S57 attributes.
- Specify a model definition for a unique buoy based on the buoy's name.
- Use a texture atlas to model buoys. (A texture atlas is an image that has sub-images that can be referenced by a coordinate system. Those sub-images can be applied to 3D models.)

All of these methods are supported by MAK ONE applications and are described in this section. However, MAK ONE applications use texture atlases for all streaming buoys and beacons.

Specifying a Model Definition for a Type of Buoy

To specify the model definition for a type of buoy, you specify it as the marker name:

```
<styles>
    <style type="text/css">
       SpecialPurposeBuoys {
       marker: "BuoysSpecialPurpose" ;
       ...
```

In this case, the model definition “BuoysSpecialPurpose” is used for all buoys streamed in for that SpecialPurposeBuoys layer.

This approach is the simplest way to specify model definitions for buoys.

Generating a Model Definition Name for a Buoy

Instead of specifying a model definition, you can have the application build a model definition using the buoy’s attributes. This approach is more complex than specifying one model definition for a type of buoy, but it is more flexible. You can accommodate many variations of a given type of buoy without having to specify many model definitions.

Buoy attributes can be strings or enumerated values (such as shape (BOYSHP), color (COLOUR), and color pattern (COLPAT)).

The following code results in model definitions like “Buoys352”.

```
<styles>
   <style type="text/css">
       SpecialPurpose {
       marker: "Buoys" + [BOYSHP] + [COLOUR] + [COLPAT];
```

Creating and managing model definitions with coded numbers in this way is difficult. Therefore, MAK ONE applications provide a mechanism to map the numeric attribute values to strings.

The enumerated value-to-string mappings are in *./appData/importConfig/s57_attr_map.csv*. You can change them to any strings that you want to use. Given this information, you can specify your model definition as a pattern and provide the enumerated attribute values. MAK ONE applications use them to decode and build the requested model definition. The pattern and attribute values are specified as a colon (:)-separated list of the following form:

```
"pattern:Attribute1=value:Attribute2=value" ...
```

The pattern itself indicates where attribute values should be replaced using '[Attribute]' substrings. For example, consider an input string such as:

```
"Buoys[COLOUR][BOYSHP]:COLOUR=5:BOYSHP=3"
```

The default mappings for COLOUR and BOYSHP in *s57_attr_map.csv* are as follows:

```
BOYSHP,,Nun,Can,Spherical,Pillar,Spar,Barrel,Super,Ice

COLOUR,,White,Black,Red,Green,Blue,Yellow,Grey,Brown,Amber,Violet,Orang
e,Magenta,Pink
 COLPAT,,HorizontalStriped,VerticalStriped,DiagonalStriped,Squared,
Striped,BorderStriped
```

Therefore, the model definition would be BuoysBlueSpherical, since the enumerated COLOUR value '5' (the fifth value in the list) is mapped to "Blue", and the BOYSHP value '3' is mapped to "Spherical". Some of these attributes, such as COLOUR, may be lists of enumerated values. The strings for enumerated lists are concatenated. The following input string:

```
"Buoys[BOYSHP][COLOUR][COLPAT]:COLOUR=3,1:BOYSHP=2:COLPAT=2
```

results in the model definition "BuoysCanRedWhiteVerticalStriped".

To generate the input string like the one above from an earth file, use:

```
marker: "Buoys[BOYSHP][COLOUR][COLPAT]"
            + ":COLOUR=" + [COLOUR]
            + ":BOYSHP=" + [BOYSHP]
            + ":COLPAT=" + [COLPAT]
```

The order that the actual attribute/value pairs are encoded (after the pattern) does not matter.

Designators

Many buoys have some form of one to three numbers, letters, or both that appear on the buoy itself. S-57 does not provide an attribute that indicates what this symbol is, but it is usually found at the end of the OBJNAM attribute, such as "Boston South Channel Buoy 13" or "President Roads Junction Lighted Buoy PR". If the OBJNAM attribute is provided, the model definition builder tries to extract this symbol. It must be 3 characters or less, and be all numerics, all capital letters, or a mix of the them. You can add this to a model definition pattern using the [DESIGNATOR] substring, for example:

```
 marker: "Buoys[BOYSHP][COLOUR][COLPAT][DESIGNATOR]"
+ ":COLOUR=" + [COLOUR]
+ ":BOYSHP=" + [BOYSHP]
+ ":COLPAT=" + [COLPAT]
+ ":OBJNAM=" + [OBJNAM]
```

This code, when streaming in “Boston South Channel Buoy 13”, would result in a model definition of “BuoysCanGreen13”, since the BOYSHP attribute for that buoy maps to “Can” and COLOUR attribute to “Green”.

Unique Buoys

There may be cases where you want to use a specific model definition for a particular buoy. The buoy might be a special landmark that needs to be recognizable. The file *appData/importConfig/s57_modeldef_by_name_map.csv* contains a list of buoy names (the OBJNAM attribute) and model definition pairs. If the OBJNAM attribute value is provided in the marker string and if it matches one of the buoy names in *s57_modeldef_by_name_map.csv*, then the specified model definition is used instead of a generated definition.

3.19.2 Modeling Buoys Using Texture Atlases

This approach is more complex than specifying a model definition per type of buoy or building one from attributes. It requires understanding how to use texture atlases. However, this method allows you to have a large number of buoys in a scene with a wide variety of colors, patterns, and designators using a very limited number of models. This dramatically reduces the scene complexity (thereby increasing frame rate) when you use instancing. This is the approach that MAK ONE applications use for all buoys and beacons it ships with.

Each buoy shape (can, pillar, nun, and so on) is a separate model. Each model has a texture atlas that has all the different colors and patterns needed for that buoy shape.

In the **marker** field (in the **model** statement) a simple model definition pattern is used:

```
Buoy[BOYSHP]
```

The other attributes are passed in the same fashion as described in "Generating a Model Definition Name for a Buoy" on page 92 when mapping attributes to model definition substrings:

```
marker: "Buoys[BOYSHP]"
+ ":BOYSHP=" + [BOYSHP]
+ ":COLOUR=" + [COLOUR]
+ ":COLPAT=" + [COLPAT]
+ ":OBJNAM=" + [OBJNAM] ;
```

In this example, all the attributes are mapped to strings, but the COLOUR and COLPAT strings are used to look up the appropriate uv texture coordinate offsets for the buoy. The texture-uv mapping file has the same name as the buoy model plus the extension *dtxc* (for example, *PillarBuoy.dtxc* and *Can.dtxc*).

The *.dtxc* files can support multiple atlases that may be used on the same model. MAK ONE applications use multiple atlases for buoys with topmarks, and beacons with daymarks and topmarks. (For information about topmarks and daymarks, please see "3.19.6 Streaming Daymarks" on page 97 and "3.19.7 Streaming Topmarks" on page 98.) The model definition for the buoy specifies a key (found in the *.dtxc* file) to be used for the main texture (**mainTextureKey**) and for the topmark's texture (**topmarkTexturekey**).

The buoy is also built with special geometry and a texture atlas for displaying the alpha-numeric designator. The geometry includes a switch to select between sites for placing one, two, or three alphanumeric characters. The characters themselves are then mapped from a texture atlas using the designator string found and a uv-coordinate texture mapping file that is shared by all buoys (and daymarks), named *LettersNumbers.ctxc*.

3.19.3 Configuring Streaming Beacons

You can specify beacon feature styles the same way you specify buoys. You can add any number of beacon styles names. The S57 attribute enumeration-to-string mappings for beacons are in *s57_layer_list.csv*.

The following code is an example of a beacon model block:

```
<model name="oahu-LateralBeaconsUS5HA52M" driver="feature_custom"
enabled="true">
     <OGRFeatures name="s57-data" driver="ogr">
         <!-- From S-57 data -->
         <url>../../TerrainData/Terrain/Hawaii/Features/US5HA52M/BOYLA
T.shp</url>
    </OGRFeatures>
    <layout>
        <tile_size_factor>3</tile_size_factor>
    </layout>
    <styles>
        <style type="text/css">
    LateralBeacons {
        model:              "Beacons[BCNSHP]"
        + ":BCNSHP=" + [BCNSHP]
        + ":COLOUR=" + [COLOUR]
        + ":COLPAT=" + [COLPAT]
        + ":HEIGHT=" + [HEIGHT]
        + ":OBJNAM=" + [OBJNAM] ;
        model-scale:        1;
        model-placement: vertex;
        altitude-clamping:   terrain;
        altitude-resolution: 0.00008583068847656250;
        altitude-offset:     0;
        }</style>
    </styles>
</model>
```

Model definitions for beacons work the same way as for buoys. They are configured in *s57_attr_map.csv*. It can contain enumeration mappings for any enumerated S-57 attribute that you want to use to create model definitions. Default values are in *s57_defaults.csv*. This contains the default name and default model definition for each style group specified in *s57_layer_list.csv*. The delivered *s57_defaults.csv* contains:

```
# If the final model definition generated does not exist,
# one of these model definitions will be used
Buoys,DefaultModelDefinition,BuoysGreenPillar1
```

```
Beacons,DefaultModelDefinition,BuoysRedPillar2

# If the feature has no name, use one of these
Buoys,DefaultName,Unnamed Buoy
Beacons,DefaultName,Unnamed Beacon
```

3.19.4 Configuring Streamed Navigation Lights

Navigation lights are managed somewhat differently from other streamed features. These lights do not result in actual objects added to the scene. The information is used to create and attach light controllers to existing objects. The lights themselves need to already exist as part of the buoy or beacon model they appear on.

Each model that has a light should have a comment-tagged switch for choosing a light group of the specified color. Each light in turn has a switch for turning the light itself on or off. We use the tag **@dis light_color** to tag a color switch, with values 0-3 corresponding to white, green, red, and yellow. The switch to turn lights off and on is **@dis navigation_lights**.

MAK ONE applications stream in light data using **model** blocks similar to those used for buoys and beacons. You can apply style names. *light_layer_list.csv* controls which styles will be read. *maritime-navigation.massachusetts.xml* uses a single style name for all lights, **NavigationLights**. The following example shows a **model** block for a light:

```
<model name="LightsUS5MA10M" driver="feature_custom"  enabled="true">
    <OGRFeatures name="s57-data" driver="ogr">
        <!-- From S-57 data -->
        <ogr_driver>S57</ogr_driver>
<url>../../TerrainData/Terrain/Massachusetts/Features/US5MA10M/LIGHTS.
shp</url>
    </OGRFeatures>
    ...
    <styles>
         <style type="text/css">
            NavigationLights {
                marker: "COLOUR=" + [COLOUR] + ":SIGSEQ=" + [SIGSEQ] +
":EXCLIT=" + [EXCLIT];
                marker-scale:        1;
                marker-placement: vertex;
                altitude-clamping:   terrain;
                altitude-resolution: 0.00008583068847656250;
                altitude-offset:     0;
            }
        </style>
    </styles>
</model>
```

Since we are not building model definitions, we do not need to map enumerated values to strings. However, we do map S-57 light colors to the colors that we support. This mapping is in *light_attr_map.csv*.

MAK ONE applications associate light controllers with other streamed features based on the S-57 or shape file locations of the lights and features, as follows:

- If the COLOUR attribute is provided, the attached light controller sets the light color using the light color switch.
- If the SIGSEQ attribute is provided, the light is turned on and off according to the specified sequence.
- If the EXCLIT attribute is provided, the light controller enables or disables the light based on daylight and fog conditions.

The light color switch and light switch values can be changed by specifying different ones through the API, specifically using:

```
de.scene().terrain().navigationLightManager().
       setLightColorSwitchNumber()
```

and

```
de.scene().terrain().navigationLightManager().setLightSwitchNumber()
```

3.19.5 Configuring Seasonal Buoys and Beacons

Seasonal buoys and beacons are supported. Include the PERSTA/PEREND attributes in the earth file as part of the marker field, and then buoys and beacons only appear during the correct season. Example:

```
marker: "Buoys[COLOUR][COLPAT][BOYSHP][DESIGNATOR]"
   + ":BOYSHP=" + [BOYSHP]
   + ":COLOUR=" + [COLOUR]
   + ":COLPAT=" + [COLPAT]
   + ":PERSTA=" + [PERSTA]
   + ":PEREND=" + [PEREND]
   + ":OBJNAM=" + [OBJNAM] ;
```

3.19.6 Streaming Daymarks

MAK ONE applications support streamed daymarks for beacons. A limited number of daymark models and textures are provided as examples. *Massachusetts S-57 (online).earth* has a **model** statement to stream daymarks from the Boston Harbor S-57 data. As with buoys and beacons, the **model** entry specifies a style name (Daymarks), which corresponds to a layer specified in *daymark_layer_list.csv*. OSGEarth models that use the feature_custom driver and have a style name that is in the layer list file, are processed as daymarks. S-57 attribute values can be mapped using the *daymark_attr_map.csv* file.

The beacon models provided with MAK ONE applications contain multiple daymark placard shapes (Diamond, Square, and Triangle), which are selected based on the TOPSHP attribute. The TOPSHP attribute is mapped to '0', '1', or '2', which are the switch values for the diamond, square, or triangle shaped marks. Other attribute values, including TOPSHP, COLOUR, COLPAT, and CATSPM are used to look up UV coordinates in the model's texture atlas using a uv-texture coordinate mapping file similar to the one described for buoys. The mappings are in the *./SharedData/16/latest/ModelData/Objects/Buoys*: *Can.dtxc*, *IceBuoy.dtxc*, *Nun.dtxc*, *PillarBuoy.dtxc*, and *Sonobuoy.dtxc*. These daymark shapes, plus textures provided,

represent all the daymarks attached to buoys. The model definition for daymarks includes a parameter that specifies a string key to use in the *.dtxc* file ("daymarkTextureKey").

3.19.7 Streaming Topmarks

Topmarks can be streamed in and attached to buoys and beacons similarly to daymarks. The shape is determined by the TOPSHP attribute. The attribute is mapped to a switch value using the *topmark_attr_map.csv* file. Textures for the topmarks can be selected from a texture atlas similarly to buoys and daymarks. The model definition for a buoy or beacon that needs to map attributes to uv-texture coordinates using that model's *.dtxc* file will have a string key (**topmarkTextureKey**) that identifies the mapping entries in the file that pertain to the topmark texture atlas.

3.19.8 Generating Signal Sequences

The S-57 attribute SIGSEQ (signal sequence) specifies the flashing sequence that a navigational aid light on a buoy or beacon displays. These sequences (along with color) are used to distinguish among different buoys and beacons at night. However, this is not a required S-57 attribute. If the SIGSEQ attribute is not present, MAK ONE applications try to generate a signal sequence using the LITCHR (light characteristic), SIGPER (signal period) and SIGGRP (signal group) attributes for the most commonly seen sequences.

As part of the class responsible for generating these sequences, the API includes a virtual function for each sequence type that is not implemented, so that a developer can implement it. The assumption is that not many of the other sequence types will be used, but it is data-specific. Once we know what sequences are needed for the S-57 data in use, it should be relatively straight-forward to implement those additional sequence generation methods.

The class used to generate signal sequences is *DtLightSignalSequenceGenerator*, in the vrvCore library. *DtLightSignalSequenceGenerator.h* provides documentation, including which methods are implemented and which are not. To extend this class, a creator is provided, *DtLightSignalSequenceGeneratorCreator*, an extended instance of which can be registered through a plug-in with the display engine.

3.20. Land Use Data

You can specify a land use layer for osgEarth terrains. VR-Vantage-based applications can use this information to decide how to render the terrain in procedurally generated terrains. The VR-Forces simulation engine can use the land use data to determine soil types, which may affect where vehicles and ships can travel and how fast they can go.



4. Editing Terrains Using the Terrain Editor

This chapter explains how to use the terrain editor to test changes to earth files.

4.1. Introduction to the Terrain Editor 100
4.1.1 Opening the Terrain Editor 100
4.2. Adding Data to a Terrain 101
4.3. Editing Terrain Properties in the Terrain Editor 101
4.3.1 Enabling and Disabling Layers 101
4.3.2 Editing the Properties of a Layer 102

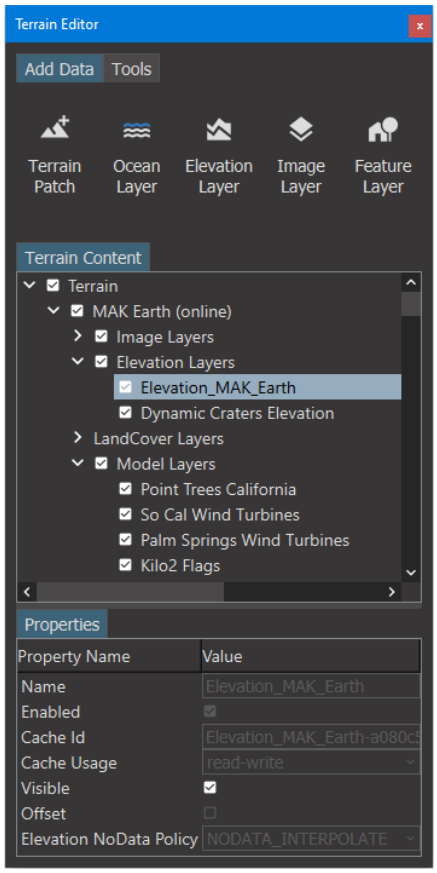
4.1. Introduction to the Terrain Editor

VR-Vantage, VR-Forces, and VR-Engage let you define and load terrains by composing them in the Terrain Editor Panel or by creating earth files that specify all of the terrain data to be loaded. These approaches to creating terrains are described in "Chapter 1. Composing Terrains" on page 35 and "Chapter 3. Loading Data Using earth Files" on page 63 the topics in the Terrain section of this help file.

The Terrain Editor (Figure 27) is a graphical user interface for composing terrains. It also lets you test changes to terrain properties in real time. If you are satisfied with a change, you can then edit the earth file itself for the terrain, or for terrain patches and server connections save the terrain to MTF, to make the change permanent.

This chapter explains how to use the Terrain Editor. It assumes that you are familiar with the types of data used to create a terrain and that you are familiar, at least at a high level, with the different types of layers and their properties that are used in an earth file. These concepts are all discussed in the chapters referenced in the previous paragraph.

Figure 27. Terrain Editor Panel



4.1.1 Opening the Terrain Editor

➤ To open the Terrain Editor, choose **View > Terrain Editor Panel**. The Terrain Editor is added to the window.

4.2. Adding Data to a Terrain

The Terrain Editor lets you add the following types of data to a terrain:

- Terrain patch
- Elevation layer
- Image layer
- Ocean layer

You can also add a server connection. This allows you to see the data available from the server.

The procedures for adding data and server connections are described in "Chapter 1. Composing Terrains" on page 35 and "2.1. Connecting to Terrain Servers" on page 58 .

When you add these types of data, they are not saved to an earth file. If you want to save these changes, you must save the terrain to an MTF file using the Save Terrain procedure.

4.3. Editing Terrain Properties in the Terrain Editor

The Terrain Editor displays a terrain's layers in the Terrain Content window. You can enable and disable the display of any layer and the application display window redisplays to reflect the change you made. In addition to enabling and disabling layers, you can add and remove them.

When you select a layer, the Properties window displays the properties of the layer. You can delete properties and change property values.

The changed properties do not get saved to the earth file. If you make a change that you want to keep, you must edit the earth file in a text editor. The advantage of using the Terrain Editor is that you can try out many changes in real time without having to edit the earth file and reload the terrain to see the effect of your edits.

If you change the following properties, you can save them to the MTF file using the standard Save Terrain procedure:

- Terrain description.
- Maximum paged LOD (osgEarth terrain only).
- Height of an ocean layer.
- LOD distance of an ocean layer.

4.3.1 Enabling and Disabling Layers

You can enable and disable the display or a layer. This provides a quick way for you to examine the visual impact of a layer. You can also use this feature to test the performance impact of a layer.

➤ To enable or disable a layer, in the Terrain Content window, select or clear the check box for the layer you want to enable or disable.

4.3.2 Editing the Properties of a Layer

When you add a layer to a terrain, the Terrain Editor automatically provides values for some basic properties, such as a name, the source file or URL, and driver. If you are editing a terrain that uses an earth file created in a text editor, the layer may have many more properties. The Terrain Editor lets you change their values.

Changed properties are only valid during the current application session.

Deleting a Layer Property

Properties are listed in a tree view. You can delete a property at any level of the tree. If a property has values or sub-properties, they are deleted, too.

➤ **To delete a layer property:**

1. In the Terrain Content window, select the layer that you want to edit. Its properties are listed in the Properties window.
2. Right-click the property or value that you want to delete and select Remove. The property or value is removed from the list.

Editing a Property's Value

➤ **To edit a property's value:**

1. In the Terrain Content window, select the layer that you want to edit. Its properties are listed in the Properties window.
2. Edit the property you want to change by typing a new value, changing the status of a check box, or selecting a file, as appropriate.



5. Configuring CDB Terrains

This chapter explains how to configure earth files to load CDB terrain databases.

5.1. Loading CDB Databases 104
5.2. Configuring CDB earth Files 104
5.2.1 Configuring the Elevation Dataset (001) 104
5.2.2 Configuring the Imagery Dataset (004) 106
5.2.3 Configuring the Raster Material Dataset (005-006) 108
5.2.4 Configuring the GeoSpecific (GS) Dataset (100) 110
5.2.5 Configuring the GeoTypical (GT) Dataset (101) 116
5.2.6 Configuring the Extended Imagery Dataset (900) 118
5.3. CDB Configuration Files 120
5.4. Disabling Intersections in Model Layers 121
5.5. Caching CDB Data (cdbTool) 121

5.1. Loading CDB Databases

MAK ONE applications can display multiple CDB datasets (elevation, imagery, geotypical points). MAK products load the datasets using an earth file. The earth file entries point to the CDB root path of the database and specify the CDB driver and the CDB dataset codes.

Because MAK uses a set of osgEarth plug-ins to load CDB databases, you have the same flexibility with CDB databases that you do with other database types. For example, you can:

- Use streaming data (from VR-TheWorld Server), or other source imagery to build a skirt around a CDB inset.
- Place an OpenFlight inset into a CDB database.
- Add lights, buildings, forests, grass, and other clutter on top of an existing CDB database.

MAK ONE applications include an example CDB terrain – *Test-CDB.mtf*. This terrain demonstrates the ability to load data in CDB format. For details about the CDB API, please see https://portal.presagis.com/support/solutions/articles/19000014917-cdb-api-.



Out of the box, osgEarth itself does not support CDB. The plug-ins used by MAK ONE applications enable support of CDB.

5.2. Configuring CDB earth Files

This section explains how to configure your CDB earth file to load the different datasets you need. You may need to configure the following datasets:

- Elevation
- Imagery
- Raster Material
- Geospecific
- Geotypical
- Extended imagery

5.2.1 Configuring the Elevation Dataset (001)

To load an elevation dataset, use the **elevation** element, as in the following example:

```
<elevation name="cdb_elevation"
            driver="cdbraster"
            cdb_dataset="1"
            cdb_component_selector1="1"
            cdb_component_selector2="1"
            cdb_cache_file="25"
            cacheid="camppend1-cdb-elevation" >
    cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
```

```
</elevation>
```

It tells osgEarth to use the CDB raster driver.

Table 7 describes the attributes for the **elevation** element.

Table 7. Attributes for elevation and imagery datasets

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be **cdbraster**).
cdb_dataset	(Optional) Specifies the CDB dataset to read. For elevation always use "1". Default: 1.
cdb_ component_ selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 8. Default: 1.
cdb_ component_ selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cdb_cache_ file	(Optional) Specifies how many CDB tiles to keep in memory while building an OSG tile. Default: 20.
cdb_max_lod	(Optional) Specifies the maximum LOD of this dataset. Set this value if you have a CDB database that is larger than 500 geocells. For example, if you have the CDB World Wide database, set the value to 0. This will restrain the CDB Driver to pre-parse all of the LOD folder. If the value is not set (default) it reads the CDB to find the maximum LOD of the elevation dataset.
cacheid	Specifies where to save the cache files, if caching is enabled.
cdb_root	Specifies the top folder of the CDB database.

Table 8 lists the **elevation** sub-dataset codes.

Table 8. elevation sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
1	1	1	Primary Elevation.
1	1	2	Primary Elevation Control (deprecated in CDB 3.2). (Not supported.)
1	1	3	Primary Alternate Elevation. (Not supported.)
1	2-99	1	Subordinate Elevation (deprecated in CDB 3.2). (Not supported.)

Table 8. elevation sub-dataset codes (continued)

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
1	2-99	2	Subordinate Elevation Control (deprecated in CDB 3.2). (Not supported.)
1	100	1	Subordinate Bathymetry.
1	100	2	Subordinate Alternate Bathymetry. (Not supported.)
1	101	1	Subordinate Tide Elevation. (Not supported.)

Configuring the Elevation Sub-dataset (Bathymetry)

Adding the bathymetry allows you to add dynamic ocean. The following example shows how to add bathymetry:

```
<elevation name="cdb_bathymetry"
            driver="cdbraster"
            cdb_dataset="1"
            cdb_component_selector1="100"
            cdb_component_selector2="1"
            cdb_cache_file="10"
            cacheid="camppend1-cdb-elevation" >
    <cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
    <offset>true</offset>
</elevation>
```

5.2.2 Configuring the Imagery Dataset (004)

The **image** element tells osgEarth to use the CDB driver to load the imagery dataset, as shown in the following example:

```
<image name="cdb_imagery"
            driver="cdbraster"
            cdb_dataset="4"
            cdb_component_selector1="1"
            cdb_component_selector2="1"
            cdb_cache_file="25"
            cacheid="camppend1-cdb-imagery" >
    <cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
</image>
```

Table 7 on the previous page describes the attributes for the **image** element. They are the same as for the elevation attribute, except that **cdb_dataset** must be 4.

Table 9 lists the codes for the imagery sub-dataset.

Table 9. Imagery sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
4	1	1	Yearly Visual Spectrum Terrain Imagery (VSTI) representation.
4	2	1	Alternate Spring representation (deprecated in CDB 3.2).
4	2	2	Alternate Summer representation (deprecated in CDB 3.2).
4	2	3	Alternate Fall representation (deprecated in CDB 3.2).
4	2	4	Alternate Winter representation (deprecated in CDB 3.2).
4	3	1-12	Alternate Monthly representation.
4	4	1-4	Quarterly VSTI representation.
4	5	1	Subordinate VSTLM. (Corresponds to the terrain light maps draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. It may be single-channel monochrome or 3-channel color image.) See "Configuring the Imagery Lightmap Sub-dataset (004-005)" below for more information.

Configuring the Imagery Lightmap Sub-dataset (004-005)

The following example shows how to add a lightmap. This code is different from the primary CDB imagery dataset.

```
<lightmap>
    <image name="cdb_imagery_lightmap"
                driver="cdbraster"
                cdb_dataset="4"
                cdb_component_selector1="5"
                cdb_component_selector2="1"
                cacheid="camppend3-cdb-lightmap"
                cache_enabled="false">
    <cache_policy usage="none"/>
    <cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
   </image>
</lightmap>
```

5.2.3 Configuring the Raster Material Dataset (005-006)

The **landcover** element tells osgEarth to use the CDB driver to load the raster material dataset, as shown in the following example:

```
<!-- Load the landcover catalog -->
<xi:include href="./osgEarthCatalogs/land_cover_dictionary.xml" />
<!-- CDB raster material used as landcover –
            osgEarth max_data_level 17 represents CDB LOD 8 -->
<LandCover name="LandCover" cacheid="cdb-landcover-campPed">
    <max_data_level>17</max_data_level>
    <xi:include href="./osgEarthCatalogs/land_cover_CDB.xml"/>
        <image name="cdb_rm" driver="cdbraster"  cdb_dataset="5"
        cdb_component_selector1="1" cdb_component_selector2="1"
                cacheid="camppend3-cdb-rastermaterial"
                cache_enabled="false">
        <cache_policy usage="none"/>
        <cdb_root>../../../../../../../Presagis/CDB/
                    CDB_Camp_Pendleton_3_0</cdb_root>
        </image>
</LandCover>
```

Table 7 on page 105 describes the attributes for the `image` and `LandCover` elements. Image is the same as for the elevation attribute, except that `cdb_dataset` must be 5. Landcover must include the image element.



You can only use `cdb_dataset="5" cdb_component_selector1="1" cdb_component_selector2="1"` in the `<image>` section for the raster material. The CDB vantage/vrf plugin will automatically gather all the needed information for the datasets below.

Table 10 on the facing page lists the codes for the Raster Material sub-dataset.

Table 10. Raster Material Descriptor sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
5	1	1-255	Corresponds to a 2D grid of material codes. The Material Layer is draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. This Dataset component follows the center grid conventions. Component selector 2 is the Material Layer number and ranges from 001 to 255.
5	1	1-254	Corresponds to a 2D grid of mixture values. The grid of mixture values is draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. This Dataset component follows the center grid conventions. Component selector 2 is the Material Mixture number and ranges from 001 to 254. Since all mixture values must add to 100%, it is possible to represent "n" Material Layers with a set of "n-1" Mixture Layers. The last Mixture Layer is implicit, and it is set to (100% - Sum of areas from previous layers). In the case where there is a single material layer, there is no need to store the (implicit) Mixture Layer since all pixel values are implicitly 100%. When there are two or more Material Layers, the Mixture Layer(s) must be generated.
6	1	1	Contains a Terrain Composite Material Table (TCMT) that provides a list of the Composite Materials shared by the Material Layers of the Raster Material dataset. There is one TCMT for each CDB tile. The Composite Material Table follows an XML notation that describes each Composite Material into its primary substrate, surface and secondary substrate components. Composite Materials are explained in further detail in Section 2.5.2, Composite Materials.

How VR-Vantage and VR-Forces Interpret the CDB Raster Material

CDB raster material is a complex dataset. It is a tile base (like most of the CDB datasets), and each pixel of the tile can be represented by multiple layers of composite material, each with a certain percentage. Also, each composite material can contain: 1 surface substrate, 1 primary substrate, and multiple secondary substrates. Each substrate can be composed of one or more CDB base materials. To simplify this,

VR-Vantage or VR-Forces read only the most important (based on percentage) CDB Base Material of the primary substrate of the raster material first layer. Once this is complete, VR-Vantage or VR-Forces map the CDB base material to a value. This mapping occurs in *./appData/cdb/CDBMaterials_ToProceduralClass.xml*. For example, BM_ASPHALT will have a value of 3. The complex workflow then assigns this value to an osgEarth class string value. This mapping is done in *./SharedData/16/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/land_cover_CDB.xml*. In that file. you can see that the value 3 maps to CDB_AsphaltOrOtherHardSurface. If you want to change the mapping, this is the field you need to edit (do not change the value and desc fields of that file). The list of CDB classes that you can use are found in *./SharedData/16/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/land_cover_dictionary.xml*.

The final, important mapping occurs between the class string of the *land_cover_dictionary.xml* in the *./appData/settings/vrfSim/landCoverDataSurfChar.map*. This is where CDB_AsphaltOrOtherHardSurface is mapped to a VR-Forces soil type: asphaltorotherhardsurface.

5.2.4 Configuring the GeoSpecific (GS) Dataset (100)

Geospecific data replicates the real world data. The following example shows how to configure a geospecific dataset:

```
<model driver="cdb"
            name="GS Man Made"
            cdb_dataset="100"
            cdb_component_selector1="1"
            cdb_component_selector2="1"
            cacheid="camppend3-cdb-gsmodels">
    <cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
    <sigsize_file>../userData/terrains/
        Sample_SignificantSize_SwitchDistance.xml</sigsize_file>
    <pagingdistance_file>../userData/terrains/
        Sample_Model_PagingDistance.xml</pagingdistance_file>
    <lightdefaultvalues_file>../appData/cdb/
        Lights_DefaultValues_3_0.xml</lightdefaultvalues_file>
</model>
```

Table 11 describes the attributes for the geospecific dataset.

Table 11. Attributes for geospecific dataset

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be **cdb**).
cdb_dataset	Specifies the CDB dataset to read. For geospecific always use “100”.

Table 11. Attributes for geospecific dataset (continued)

Attribute Name	Description
cdb_component_ selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 12. Default: 1.
cdb_component_ selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cacheid	Specifies where to save the cache files if caching is enabled. If Accelerated File Loading is enabled, the CDB geospecific models are converted to MEDF and the textures are converted to MEIF on the first load. They will be used the next time the CDB database is loaded.
cdb_root	Specifies the top folder of the CDB database.
elevation_cache	(Optional) Creates an elevation binary cache file for clamp models. Default: true.
texture_mipmap_ offset	(Optional) Specifies an offset to load lower mipmap textures from all geotypical textures. (Geospecific models can have geotypical textures.) Allowed values: 0-9. 0 means no offset. 1-9 means load a lower numbered LOD by the amount of the offset. For example, if a model is using a texture LOD 9, setting the offset to 2 causes it to load LOD 7.
debug	(Optional) Enables (true) or disables (false) debug information on CDB tiles.
debug_modelcolor	(Optional) If set to true (debug also must be set to true), color the models with the same color as the tiles.
range_factor	(Optional) Multiplier of the tile paging values found in *Model_ PagingDistance.xml*.
sigsize_file	(Optional) Replace the default significant size file (*./appData/cdb/SignificantSize_SwitchDistance.xml*) with another version.
pagingdistance_file	(Optional) Replace the default paging distance file (*./appData/cdb/default Model_PagingDistance.xml*) with another version.
lightdefaultvalues_ file	(Optional) Replace the default light configuration file (*Lights_ DefaultValues_3_2.xml*) with another one. Use this parameter only for Component Selector 1 (4 and 5, that is, Airport Light and Environment Light).

Table 12 lists geospecific sub-dataset codes.

Table 12. Geospecific sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
100	1	1	Man-made point features.
100	1	3	Man-made linear features (not supported).
100	1	5	Man-made areal features (not supported).
100	2	1	Natural point features.
100	2	2	Natural linear features (not supported).
100	2	5	Natural areal features (not supported).
100	3	1	Trees point features.
100	4	1	Airport light point features.
100	4	3	Airport linear features (not supported).
100	4	5	Airport areal features (not supported).
100	5	1	Environmental light point features.

Configuring the Geospecific Airport Lightpoint Sub-dataset (100-004)

The following example code shows how to configure a geospecific airport lightpoint dataset:

```
<model driver="cdb"
                name="GS Airport Light"
                cdb_dataset="100"
                cdb_component_selector1="4"
                cdb_component_selector2="1"
                cacheid="camppend3-cdb-gsairportlights">

    <cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
</model>
```

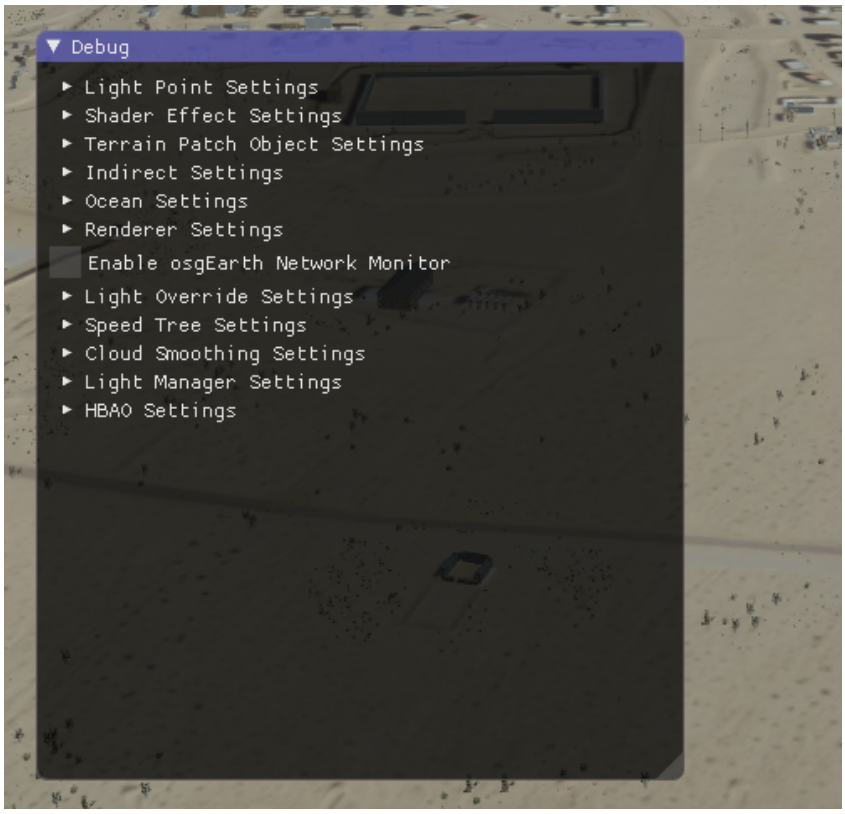
You can test the proper functioning of the lightpoints by turning them on and off using debugging tools:

➤ **To turn airport light points on or off using the debug window:**

1. Choose **Settings** > **Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 37 on page 163).

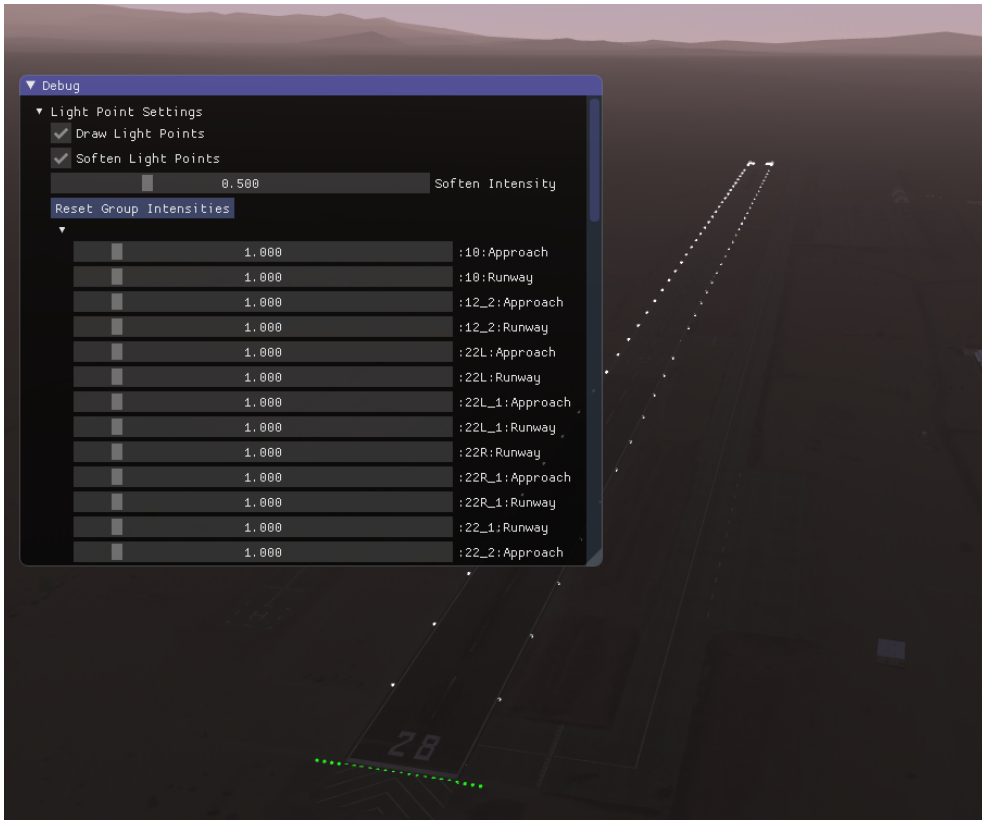
3. Select the Show Developer Debug Overlay check box. The Debug window opens (Figure 28).

Figure 28. Debug window



4. Expand Light Point Settings.
5. Toggle the slider of the runway you would like to change. For example, in the Camp Pendleton terrain, if you want to close the runway KNFG: 21, set the slider to 0.

Figure 29. Camp Pendleton airport with debug window



Configuring the Geospecific Environmental Lightpoint Sub-dataset (100-005)

The following example shows how to configure geospecific environmental lightpoints:

```
<model driver="cdb"
                    name="GS Environment Light"
                    cdb_dataset="100"
                    cdb_component_selector1="5"
                    cdb_component_selector2="1"
                    cacheid="camppend3-cdb-gsenvlights">
    <cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
</model>
```

Configuring Light Point Properties

You can change the behaviors of a type of light point. Light types are configured in *./appData/importConfig/Lights_DefaultValues_3_2.csv*. To change a light point type, you can edit the file in a text editor or spreadsheet. Each light type has the following attributes:

- Code: Light type code (do not edit).
- Description: Light type description (do not edit).
- Intensity: From 0.0 to 1.0.

- Red-Green-Blue: RGB value of the color (from 0.0 - 1.0 for each channel).
- Directionality: Directional, Omnidirectional, Bidirectional.
- Horizontal Lobe: When a light type is not native to the CDB specification, the Horizontal field represents the light point's half-intensity horizontal lobe width for the client-device (range from 0.0 to 360.0). When the light type is native to the CDB specification, the Horizontal field is used as a floating-point modifier that multiplies the horizontal lobe width calculated by the client-device. It applies only to Directional and Bidirectional light types. Default: 1.0.
- Vertical Lobe: When a light type is not native to the CDB specification, the Vertical field represents the light point's half-intensity vertical lobe width for the client-device (range from 0.0 to 360.0). When the light type is native to the CDB specification, the Vertical field is used as a floating-point modifier that multiplies the vertical lobe width calculated by the client-device. It applies only to Directional and Bidirectional light types. Default: 1.0.
- Residual Intensity: When a light type is not native to the CDB specification, Residual Intensity represents the residual intensity of the light. Residual Intensity is the intensity of the light (range normalized from 0.0 to 1.0) outside of the lobe defined by Lobe Width:Horizontal and Lobe Width:Vertical fields. When the light entry is native to the CDB specification, Residual Intensity is used as a floating-point modifier that multiplies the residual intensity calculated by the client-device. It applies only to Directional and Bidirectional light types. Default: 1.0.
- Frequency: A floating-point value greater than or equal to 0.0 that sets the blink or rotating frequency of the light in Hertz (cycles per second). A value of 0.0 disables all blinking and rotating properties. Default: 0.0.
- Duty Cycle: A floating-point value ranging from 0.0 to 1.0 that sets the duty cycle of the light. Duty cycle is defined as the percentage of time the light is turned on over a complete cycle. A value of 0.0 permanently turns the light off. A value of 1.0 turns it on. The value is ignored if Frequency = 0.0. Default: 0.5.
- Min Pixel Size: Minimum number of pixels that represent the light point when you are far from it. (This attribute is specific to VR-Vantage applications.)
- Max Pixel Size: Maximum number of pixels that represent the light point when you are close to it. (This attribute is specific to VR-Vantage applications.)
- Max Visible Distance: Distance in meters when the light point will disappear. (This attribute is specific to VR-Vantage applications.)

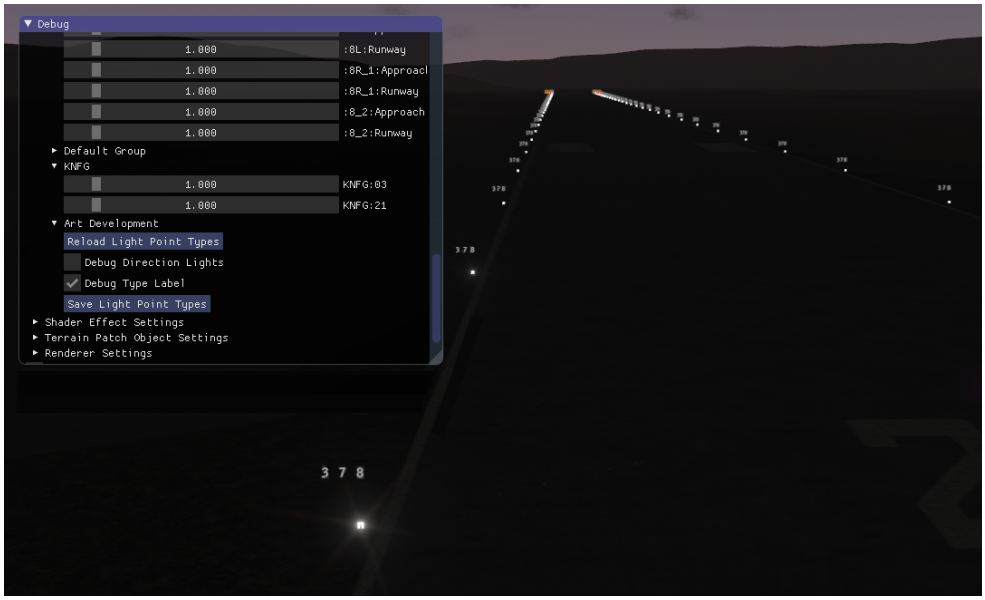
To change the behaviors of a light point type, you need to know what light type is being used.

➤ **To identify the type of light point:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 37 on page 163).
3. Select the Developer Debug Overlay check box. The Debug window opens (Figure 28).
4. Expand Light Point Settings.

5. Select the Debug Type Label check box.
6. Zoom in on the light point that you want to edit. It is labeled with a number (Figure 30).

Figure 30. Lightpoint type labels



5.2.5 Configuring the GeoTypical (GT) Dataset (101)

Geotypical data is data that is typical for a particular type of object but does not necessarily match the object in the real world.

The following example shows how to add a geotypical dataset to an earth file:

```
<model driver="cdb"
            name="GT Trees"
            cdb_dataset="101"
            cdb_component_selector1="2"
            cdb_component_selector2="1"
            cacheid="camppend3-cdb-gtmodels2">
    <cdb_root>../../../../../../../Presagis/CDB/
        CDB_Camp_Pendleton_3_0</cdb_root>
    <sigsize_file>../userData/terrains/
        Sample_SignificantSize_SwitchDistance.xml</sigsize_file>
    <pagingdistance_file>../userData/terrains/
        Sample_Model_PagingDistance.xml</pagingdistance_file>
    <speedtree_catalog>Camp_Pendleton_3_0_SpeedTrees.xml
        </speedtree_catalog
    <ogrfeatures name="Vegetation"/>
    <styles>
        <style type="text/css">
        Vegetation {
            marker-scale:        1;
            marker-placement: vertex;
            altitude-clamping:   terrain;
```

```
            altitude-resolution: 0.00008583068847656250;
            altitude-offset:     0;
        }</style>
    </styles>
</model>
```

Table 13 describes the attributes for a geotypical dataset.

Table 13. Attributes for geotypical dataset

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be **cdb**).
cdb_dataset	(Optional) Specifies the CDB dataset to read. For geotypical data always use “101”.
cdb_component_selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 14. Default: 1.
cdb_component_selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cacheid	Specifies where to save the cache files if caching is enabled. If Accelerated File Loading is enabled, the CDB geotypical models are converted to medf and the textures are converted to meif on the first load. They will be used the next time the CDB database is loaded.
cdb_root	Specifies the top folder of the CDB database.
elevation_cache	(Optional) Creates an elevation binary cache file for clamp models. Default: true.
texture_mipmap_offset	(Optional) Specifies an offset to load lower mipmap textures from all geotypical textures. Allowed values: 0-9. 0 means no offset. 1-9 means load a lower numbered LOD by the amount of the offset. For example, if a model is using a texture LOD 9, setting the offset to 2 causes it to load LOD 7.
cache_policy	(Optional) Specifies whether or not to cache models.
debug	(Optional) Enables debug information on CDB tiles.
debug_modelcolor	(Optional) If set to true, color the models with the same color as the tiles. The debug attribute must also be set to true.
range_factor	(Optional) Multiplier of the tile paging values found in *Model_PagingDistance.xml*.
sigsize_file	(Optional) Replace the default significant size file (*./appData/cdb/SignificantSize_SwitchDistance.xml*) with another version.

Table 13. Attributes for geotypical dataset (continued)

Attribute Name	Description
pagingdistance_ file	(Optional) Replace the default paging distance file (*./appData/cdb/default Model_PagingDistance.xml*) with another version.
rendering_ method	Valid values are OSG_RENDERING or INDIRECT_RENDERING. Default: INDIRECT_RENDERING. Using the indirect rendering value sets this model layer to use the optimized indirect rendering system, which allows the models to be rendered much faster than the OSG rendering method.
speedtree_catalog	(Optional) Specifies a file that lists the CDB geotypical SpeedTree replacement. (For more information, please see"5.3. CDB Configuration Files" on page 120 The **ogrfeatures** and **styles** elements must be present as in the example for SpeedTrees to work with CDB.

Table 14 lists the codes for the geotypical sub-dataset.

Table 14. Codes for geotypical sub-datasets

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
101	1	1	Man-made point features.
101	1	3	Man-made linear features (not supported).
101	1	5	Man-made areal features (not supported).
101	2	1	Tree point features.
101	2	2	Tree linear features (not supported).
101	2	5	Tree areal features (not supported).
101	3	1	Moving model location features (not supported).

5.2.6 Configuring the Extended Imagery Dataset (900)

The following example shows how to add the extended imagery dataset to an earth file.



This dataset needs the CAE API pro license and the configuration file *.\appData\cdb\900_ExtImagery.xml*. These extensions are not part of the CDB specification, but can permit vendors to extend CDB to their needs.

```
<image name="cdb_cartography"
           driver="cdbraster"
```

```
            cdb_dataset="900"
            cdb_component_selector1="1"
            cdb_component_selector2="1"
            cdb_cache_file="25"
            cacheid="yemen-cdb-cartography"
    <cdb_root>../../../../../../../Presagis/CDB/CDB_Yemen_3_2</cdb_root>
</image>
```

Table 15 describes the attributes for configuring extended imagery.

Table 15. Extended imagery attributes

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset (must be **cdbraster**).
cdb_dataset	(Optional) Specifies the CDB dataset to read. For imagery data always use “900”. Default: 1.
cdb_ component_ selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, please see Table 16. Default: 1.
cdb_ component_ selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cdb_cache_ file	Specifies how many CDB tiles to keep in memory while building an OSG tile. Default: 20.
cacheid	Specifies where to save the cache files if caching is enabled.
cdb_root	Specifies the top folder of the CDB database.
cdb_max_lod	(Optional) Specifies the maximum LOD of this dataset. Set this value if you have a CDB database that is larger than 500 geocells. For example, if you have the CDB World Wide database you can set that value to 0. This will cause the CDB driver to pre-parse all of the LOD folder. If the value is not set (default) it reads the CDB to find the maximum LOD of the imagery dataset.

Table 16 lists the sub-dataset code for extended imagery.

Table 16. Extended imagery sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
900	1	1	Cartography imagery.

5.3. CDB Configuration Files

MAK ONE applications use the following files to configure CDB terrain databases.

- *Datasets.xml*. This file lists the CDB 3.2 Datasets code and folder name. Do not edit this file until a new CDB specification is released or MAK creates an extended CDB dataset. This list is read by the MAK CDB API when the CDB DLLs are loaded.
- *Feature_Data_Dictionary.xml*. This file contains all the FACC, FSC code (Facc sub code) definitions from the CDB 3.2 specification. Do not edit this file.
- *SignificantSize_SwitchDistance.xml*. CDB uses the concept of Significant Size to switch between LODs. Since this concept is not supported currently by makOSG, the makOSG Openflight plug-in was modified to support FLT CDB models. To support the CDB model Significant Size, the plug-in was modified to convert SigSize into normal LOD switch in/out distance. The Switch distance is supported by makOSG. The plug-in contains default values that can be overwritten by this file. As a guideline, each CDB LOD in the file (also associated to a significant size) should have a different switch in distance greater than the next LOD (this will prevent abnormal behavior at runtime).

  Inside this file, switch distance can be defined by CDB datasets.

  You can override this file for each dataset by adding the following code to your *.earth* file:

  `<sigsize_file>.\yourSigSizeFile.xml</sigsize_file>`
- *Model_PagingDistance.xml*. Configures the distance for the view point when each LOD tile is getting loaded. This can be configured by CDB datasets.

  This file can be overwritten in the earth file for each dataset by using:

  `<pagingdistance_file>./AnotherModelPagingDistanceFile.xml</pagingdistance_file>`
- *Light_DefaultValues_3_2.xml*. This file contains all the CDB 3.2 light point default values defined in *CDB Specification - Volume 2.pdf*, Appendix E. By default, VR-Vantage uses *./appData/importConfig/Light_DefaultValues_3_2.csv* to configure light properties for CDB databases. You can override it by adding the **<lightdefaultvalues_file>** element to your earth file and specifying *Light_DefaultValues_3_2.xml* (or *Light_DefaultValues_3_0.xml*) as the source of light properties.

  If it is specified, the Model_CDB plug-in reads this file at load time.

  These values complete the information found in your CDB database under *./Metadata/Lights.xml*. Do not edit this file unless an error is found.

  You can specify a light properties file for each light dataset by using:

  `<lightdefaultvalues_file>./AnotherLightDefaultValuesFile.xml</lightdefaultvalues_file>`

  For example, if you want to use the CDB 3.0 default values you could add this to the airport lights section:

```
<lightdefaultvalues_file>./appData/cdb/Light_DefaultValues_3_
0.xml</lightdefaultvalues_file>
```

- *Light_DefaultValues_3_0.xml*. This file contains all the CDB 3.0 light point default values defined in *CDB Specification - Volume 2.pdf*, Appendix E. Do not edit this file unless you find an error in it except for the following attributes:
  - **<Radius>12.0</Radius>**. Radius in meters of the OSG light point.
  - **<MinPixelSize>2.0</MinPixelSize>**. Minimum pixel size of the OSG light point node.
  - **<MaxPixelSize>4.0</MaxPixelSize>**. Maximum pixel size of the OSG light point node.
  - **<MaxVisibleDistance>12000</MaxVisibleDistance**>. Maximum visible distance in meters.
- *Lights_xxx.xml*. This file shows how to do specific light tuning. Do not change the name of this file. The content must be updated with your device specific tuning and you must put the file in your CDB database's */Metadata* directory.
- *900_ExtImagery.xml*. This file is needed to extend the API to support the 900_ ExtImagery dataset in the Yemen CDB database. (You must have a CAE API pro license to support this.)
- *XXX_SpeedTrees.xml*. This file is used to map speedTrees to the CDB geotypical model. For example, you can create a file called *Camp_Pendleton_3_0_ SpeedTrees.xml* that will be referenced in the Geotypical tree section of your earth file. (For details, see "5.2. Configuring CDB earth Files" on page 104. The file will usually be in the same folder as the earth file.

5.4. Disabling Intersections in Model Layers

When users place simulation objects on a terrain, the intersections can be quite slow. If you do not need to check intersections for certain model layers, you can disable them:

```
<model
  ...
  <enable_intersections>false</enable_intersections>
</model>
```

This applies to geotypical as well as geospecific layers.

5.5. Caching CDB Data (cdbTool)

When caching is enabled, MAK ONE applications automatically cache terrain data as you load it and stores the data for future use.

MAK ONE applications have multiple levels of caching for CDB data:

- Models and textures. Set the **cacheid** field in your earth file. We recommend that you give it an appropriate name, such as **camppend3-cdb-gtmanmademodels**.

- Model clamping elevation. This is done automatically the first time CDB tiles are paged in if it is not disabled in the earth file.
- Height map and normal map. This is done automatically the first time CDB tiles are paged in.
- Elevation and imagery. If you enable osg caching in the earth file, this provides some load performance gain, but may require a lot of disk space.

When you cache data as it loads, initial load times can be lengthy. You can use the cdbTool to compress CDB models (flt) and textures (rgb) offline to speed up initial load times and runtime performance. (You can also run the tool on CDB databases that have been partially cached by automatic caching. In that case the tool only compresses the models and textures that have not yet been compressed.)

The syntax for the cdbTool is as follows:

```
./bin64/cdbTool.exe -f earth_file [-p threads -c cache_path -- -v -h]
```

where:

(-p | --threads) *threads* is the number of threads to use. Too many threads will be slow and run out of memory.

(-c | --CachePath) *cache_path* is the directory to use to save the information. Default: *./appData/cache*.

(-f | --EarthFile) *earth_file* is the input earth file name and path containing the CDB datasets information.

(-- | --ignore_rest) causes MAK ONE applications to ignore the rest of the arguments following this flag.

(-v | --version) displays version information and exits.

(-h | --help) displays usage information and exits.

The following example caches *CampPendleton3-CDB.earth* in the default directory using four threads:

```
cdbtool -f "..\userData\terrains\CampPendleton3-CDB.earth" -p 4
```



6. Processing MetaFlight Files

To display MetaFlight files in MAK ONE applications, you must first process them. This chapter explains how to do so.

6.1. Building Efficient MetaFlight Terrains 124
6.2. Processing MetaFlight Files for Use in MAK ONE Applications 124
6.3. Splitting Datasets into Individual MetaFlight Files 125
6.4. Converting Virtual Texture Datasets to Tiled Textures 125
6.4.1 The mftTool 125
6.4.2 Using the mftTool to Process MetaFlight Files 127
6.4.3 Convert Geometry Grid Datasets with Tiled Textures 128
6.5. Convert Source Data into MEDF Format 128
6.6. Create an MTF File for the MetaFlight Terrain 129

6.1. Building Efficient MetaFlight Terrains

MAK ONE applications can only load MetaFlight terrains built with certain tools. All MetaFlight files must be preprocessed before they can be loaded. The following are recommendations for constructing MetaFlight terrains that can be processed and loaded into a MAK ONE application:

- The MetaFlight terrain can contain one or more Geometry Grid Datasets (GGDS).
- The MetaFlight terrain can contain one or more Virtual Texture Datasets.
- The MetaFlight terrain can contain one or more Cultural Feature Datasets.
- The MetaFlight terrain can be built in the Geodetic, Geocentric, UTM, or Flat Earth coordinate systems.
- When using a geocentric coordinate system, the FLT file must include a transform at the top of the file and an empty degree of freedom (DOF) node under it. The DOF maintains the transform when you convert it to MEDF format and the FLT is optimized.
- For the best conversion of Virtual Textures into a tiled texture GGDS, match every Virtual Texture layer with a geometry layer.
- For optimal performance, the GGDS should be set up in a Pyramid structure (Level 1: 1x1, Level 2: 2x2, Level 3: 4x4, and so on).
- For optimal performance, levels 1-4 should be made of FILE LOD GEOM tiles only. Do not put any children LOD GEOM tiles in these levels.
- For optimal performance, the terrain should be balanced so that any single FILE LOD GEOM is not too large and can be paged in real-time (based on machine speed and video card). In other words, keep any single file small.
- For optimal performance, any single FILE LOD GEOM should not contain too many LOD GEOM nodes as children.
- Geometry Grid Tile size (in meters) should match Virtual Texture Tile coverage (in meters).
- For optimal performance, limit the number of external references. This reduces cull times. (For example, instead of having 10 individual buildings, externally reference one city block with all 10 buildings.)

6.2. Processing MetaFlight Files for Use in MAK ONE Applications

Before you can load a MetaFlight file in a VR-Vantage application, you must process it as described in this set of topicschapter. If you try to load an unprocessed MetaFlight file, the application displays an error message. If you run the MAK processing tools on a MetaFlight database that does not conform, an error message is sent to the console window.

To process a MetaFlight file, you must:

1. Convert source data into MEDF and MEIF format.
2. Split each dataset into its own MetaFlight file.
3. Convert geometry from using virtual texture datasets into using tiled textures. (If the terrain does not use virtual textures, you can skip this step.)
4. Use the terrain composition process to combine the MetaFlight terrain files and save them in MTF format.



Geometry files referenced by a MetaFlight file must be in OpenFlight format, and not in Vega Prime's binary format (*.vsb*). MAK's terrain libraries do not support VSB files.

6.3. Splitting Datasets into Individual MetaFlight Files

MAK tools work on one Geometry Grid Dataset (GGDS) at a time. Some MetaFlight files contain multiple GGDSs, so each one has to be moved into its own file. It is your responsibility to understand the syntax of the MetaFlight file and to know whether or not you need to split up datasets.

➤ **To split datasets:**

1. Create a copy of the original file for each dataset. Give each a name that identifies the dataset it contains.
2. Open each copy of the original file in a text editor.
3. In each file, delete all of the <GeometryGridDataset> XML element blocks except the one for the dataset you want to have in this file.

When you are finished, each MetaFlight file will have only one GGDS and possibly one or more Virtual Texture Datasets (VTDS).

6.4. Converting Virtual Texture Datasets to Tiled Textures

The display engine does not handle virtual textures at runtime. Therefore you must convert the virtual texture datasets into tiled textures.

6.4.1 The mftTool

The mftTool is a command-line tool for converting virtual texture datasets into tiled textures so that they can be loaded by MAK ONE applications. It creates the highest resolution textures possible from the virtual texture dataset and binds them to individual tiles.

The syntax for mftTool is as follows:

```
mftTool -f filename [-c index -w index -l index -m string
  -r path -x size -s directory -t filename -A -g -e -u
  -p threads --processEmats --extendLevelZero distance -v -h --]
```

Table 17 describes the command-line options.

Table 17. mftTool command-line options

Option	Description
(-- | --ignore_rest)	Ignore all arguments after this one.
(-A | --regenerateAll)	Forces the regeneration of all tiled textures. This can be much slower and is needed less frequently if the virtual texture dataset does not change.
(-c | --col) *index*	The column to convert, using a zero-based index.
(-e | --encryptImages)	The tiled textures are generated in the MEIF file format. If not specified, the file format will be DDS.
(-z | --extendLevelZero *distance*	Specifies the page-in range (in meters) of grid level zero in the MEDF and MetaFlight files that the mftTool creates. Use this option if the terrain disappears when you zoom out, particularly in Plan View mode. Increasing the range causes the terrain to page in at a greater distance.
(-f | --file) *filename*	The name of the MetaFlight file that contains the GGDS.
(-g | --generateFiles)	Generate tiled textures from the virtual texture and generate tiles in MEDF format from the FLT tiles that use the new textures. An MFT file is published. The output name will be equal to the MetaFlight file specified with the -f option with the string _medf appended to the end before the extension. For example, converting a file named *yourTerrain.mft* creates a new MetaFlight file named *yourTerrain_medf.mft* in the same directory as *yourTerrain.mft*. If the generated tiled texture already exists, the file is not regenerated. The -u and -g arguments are mutually exclusive.
(-h | --help)	Displays usage information.
(-l | --level) *index*	The level to convert, using a zero-based index.
(-m | --missing) *string*	The missing texture placeholder.
(-p | --threads) *threads*	Specifies the number of threads to run. Specifying zero results in the number of threads equaling the number of logical processors the host machine has.
--processEmats	Find materially classified (emat) versions of the virtual textures and generate new emat textures.
(-r | --searchDir) *directory*	Adds a directory to the search path when resolving external references. During processing any external references not found are removed from the generated files.

Table 17. mftTool command-line options (continued)

Option	Description
(-s | --subDir) *directory*	The path to the substitution texture specified by the VTDS. The arguments -s and -t are mutually exclusive.
(-t | --vtFile) *file*	The name of the MetaFlight file that contains the VTDS. The arguments -s and -t are mutually exclusive.
(-u | --publishMftOnly)	Publish a new MetaFlight file that can be loaded by VR-Vantage applications. The output name will be equal to the MetaFlight file specified with the -f option with the string _medf appended to the end before the extension. Converting a file named *yourTerrain.mft* creates a new MetaFlight file named *yourTerrain_medf.mft* in the same directory as *yourTerrain.mft*. The -u and -g arguments are mutually exclusive.
(-v | --version)	Display version information and exit.
(-w | --row) *index*	The row to convert, using a zero-based index.
(-x | --textureSize) *size*	Specifies the maximum texture size to generate in pixels per dimension. Larger textures use more memory and are slower to page in, but allow for greater virtual texture resolution.

If neither the generate (-g) or the publish only (-u) argument is specified, mftTool prints out information regarding the MetaFlight terrain.

6.4.2 Using the mftTool to Process MetaFlight Files

To process a terrain with a single MetaFlight terrain file that has one GGDS and one VTDS, run the mftTool as follows:

```
mftTool -f C:\yourTerrain\yourTerrain.mft
  -s C:\yourTerrain\ds200-vt-otw-virtual-texture -g -A -e
```

The following example shows how to process a terrain with two MetaFlight terrain files, one with the GGDS and one with the VTDS. In this case, the tiles also contain external references to other OpenFlight models.

```
mftTool
  -f C:\yourTerrain\ds100-flight-terrain\
  ds100-flight.mft
  -t C:\yourTerrain\ds200-vt-otw-virtual-texture\
  ds200-vt.mft
  -r C:\yourTerrain\textures
  -r C:\yourTerrain\objects
  -m C:\yourTerrain\MissingTexture.png -x 4096 -g
```

This creates a new MetaFlight file that references the new geometry grid dataset with virtual textures bound to it.

6.4.3 Convert Geometry Grid Datasets with Tiled Textures

The mftTool only needs to publish the MFT when the GGDS uses tiled textures. Cultural feature datasets are often built this way. These can be converted using the medfTool like all other files. The command syntax for the medfTool is described in "18.16. Compressing Model Files" on page 289.

To only publish a new MetaFlight for the cultural dataset, do the following:

```
mftTool -f C:\yourTerrain\flt\ds7_culture_db5_building_area_new-hier_
cult\ds7_culture_db5_building.mft -u -p 0
```

To convert a MetaFlight with tiled textures into a compressed format, use the medfTool, as follows:

```
medfTool -s C:\yourTerrain\flt\textures
  -s C:\yourTerrain\flt\objects
  --directory C:\yourTerrain\flt\
  ds7_culture_db5_building_area_new-hier_cult -x flt
  -m rgb -m rgba -m jpg -m bmp -p 0 -z 1
```



The medfTool should never be run on a directory that contains GGDS tiles that use Virtual Textures.

6.5. Convert Source Data into MEDF Format

Many file formats, including OpenFlight files, are often slow to load, so it is best to convert them to the fast loading MEDF (MAK Encrypted Data Format) format. Do this for terrains that do not have virtual textures (it is done automatically when creating geometry grid datasets from virtual texture datasets) and on cultural feature datasets. This should be done on all data files being loaded, but is required for all external references used by MetaFlight terrains.



You must convert the files to MEDF and MEIF before you run the mftTool.

To convert external references in separate directories:

```
  medfTool -s C:\yourTerrain\flt\textures --directory
C:\yourTerrain\flt\objects[/br]&amp;nbsp;&amp;nbsp;&amp;nbsp; -x flt -m rgb -m rgba
-m jpg -m bmp -p 0 -z 1
```

To convert external reference textures that are in the same directory:

```
  medfTool --directory C:\yourTerrain\flt\textures -m rgb
  -m rgba -m jpg -m bmp -p 0 -z 1
```

6.6. Create an MTF File for the MetaFlight Terrain

Once you have processed all of the MetaFlight files, create an MTF file for the terrain.

> Any missing models or white objects usually indicate missing external references. Fixing any missing search paths passed to the tools should resolve most problems.

➤ **To create an MTF file for the MetaFlight terrain:**

1. Start a VR-Vantage-based application.
2. Add each MetaFlight terrain as a terrain patch.
3. Save the terrain as an MFT terrain.





7. 2D View Layers

Terrains can include multiple layers to display in the 2D projection (plan view). There is a terrain, *./SharedData/16/latest/TerrainData/TerrainConfiguration/Test-2DViewLayers.mtf*, included in the terrains data package for MAK ONE applications, that shows a simple example of layers in different map groups.

7.1. 2D View Layers Overview 132
7.2. Defining 2D View Layers in earth Files 132

7.1. 2D View Layers Overview

The 2D View Layers feature allows you to define alternate maps in an *.earth* file to use in MAK ONE applications when using the 2D projection (plan view). These maps can help you to better understand your scenario or plan your exercise, while still allowing you to see a realistic terrain in the 3D projection (stealth). Alternate maps could include annotations, contour maps, feature models, geodetic graticule, MGRS graticule, aerodromes, and so on.

The osgEarth VisibleLayers can be placed into groups and then displayed using the 2D View Layer Settings toolbar or a saved observer view. To learn about saving and using observer views, see:

- "Saving and Recalling Views" in the *VR-Vantage Users Guide*
- "Saving and Recalling Views" in the *VR-Forces Users Guide*



MAK ONE applications track which layers you are rendering and, after the length of time specified in the General Application Settings, unloads the unused layers. Visually, you do not need to page all the layers of 3D rendering to work with the 2D layers. If you need the layers to load—for example, intersections for buildings—you must set MAK ONE applications to load those layers.

7.2. Defining 2D View Layers in earth Files

2D View Layers use two annotations in osgEarth: mapgroup and mapgroup_isect.

- The `mapgroup` annotation indicates the groups where you want a layer to be visible. If a `mapgroup` is defined for a layer and does not include the default group, it is visible in 2D (plan view) but not in 3D (stealth). The assumption is that the map groups are used only in plan views. If a layer does not have any `mapgroup` annotation, then it is included only in the default map group. The layers that are part of the default map group are visible in both 2D and 3D. Most layers are part of the default group.
- The `mapgroup_isect` annotation indicates that a layer should participate in intersections, such as intervisibility lines, even when a different group is selected.
- The `enabled` annotation indicates whether a layer should open by default.

The following example indicates that the Ala Moana image is part of one map group: air. When you load the earth file, the air group option is listed in the 2D View Layers list. When you select it, the image is visible in the plan view but not the stealth view.

```
  <GDALImage name="Alamona Imagery2" driver="gdal"  lod_
blending="true">
    <cache_policy usage="none"/>
    <mapgroup>air</mapgroup>
    <enabled>false</enabled>

<url>../../TerrainData/Terrain/HawaiiTifExample/otw/AlaMoanaNewRetouche
d.tif</url>
  </GDALImage>
```

In the next example, the Oahu Skyscrapers layer is included in multiple map groups: air, naval, and the default. Because it is available in the default group, it is visible in both the plan and stealth views. Additionally, because `mapgroup_isect` is set to true, the layer is included in intersections when a different map group (such as ground) is selected. The buildings are not visible when any group other than the two specified map groups are selected.

```
  <FeatureModel name="Oahu_Skyscrapers" driver="feature_geom"
enabled="true" vrfsim:enabled="true" cacheid="oahu_skyscapers">
    ...
    <mapgroup>naval,default</mapgroup>
    <enabled>false</enabled>
    ...
  </FeatureModel>
```

If you look at a terrain that included these osgEarth examples, selecting the air layer from the 2D View Layers list would show you the imagery from the first example and the buildings from the second example.





8. Dynamic Terrain in VR-Forces

This chapter describes the dynamic terrain feature in VR-Forces.

8.1. Introduction to Dynamic Terrain 136
8.2. Manually Changing the Terrain 136
8.3. Resetting Dynamic Terrain Changes 137
8.4. Removing Dynamic Terrain Damage 137
8.5. Resetting All Changes in Dynamic Terrain 137
8.6. Creating Dynamic Terrain Areas 138
8.7. Dynamic Terrain Switch Types 138
8.8. Craters 138

8.1. Introduction to Dynamic Terrain

VR-Forces supports the following types of dynamic terrain:

- Damage to buildings as the result of munition impact.
- The ability to open and close doors, windows, and other features.
- Deformable terrain (craters) in response to munition impact.

Changes to buildings depend on the presence of switch nodes and appropriate model states in the models that can be damaged or changed.

If certain types of terrain damage take place in a navigation area, VR-Forces can automatically regenerate navigation data for the terrain. This lets simulation objects take the damage into account as they move through the terrain. The priority of regeneration and the types of terrain change that trigger it are configured in *./appData/settings/vrfSim/navigationProfiles.mtl*. By default, navigation data is regenerated for damage, and damage_structure switch types. For information about navigation areas and navigation profiles, see the *Generating Navigation Data* chapter in the *VR-Forces Users Guide*. For information about switch types, see "8.7. Dynamic Terrain Switch Types" on page 138



The speed of navigation data regeneration depends on different factors (most notably sector size, number of affected sectors, and the complexity of the terrain), but in all cases it is not instantaneous. It takes at least several seconds, and possibly considerably longer. It is not abnormal for regeneration to take up to 20-30 seconds even in a well sectorized terrain. Therefore it is normal for entities to continue navigating using the original navigation data for a brief period until navigation data is updated.

For details about how terrain is damaged, please see "Configuring Damage for Dynamic Terrain" in the *VR-Forces Users Guide*.

8.2. Manually Changing the Terrain

Typically, terrain changes occur as the result of munition damage or actions of players in a scenario. For example, a character might open a door so that it can enter a building. However you can manually apply damage states or make other changes.

➤ **To manually change the state of a terrain object:**

1. Choose **Simulation > Dynamic Terrain > Add Changes**. The Add Dynamic Terrain Changes dialog box opens. The cursor changes to a green circle that represents the area within which changes can be made.
2. Optionally, change the radius.
3. Click on the terrain or enter the location where you want to make changes. If there are any dynamic terrain models in that area they are listed in the list window.

4. For each option listed that you want to change, select the state you want from the list.
5. Click OK.

8.3. Resetting Dynamic Terrain Changes

If you have made changes to the terrain by hand, you can reset those changes. Reset Changes also lets you remove damage created by a munition.

➤ **To reset dynamic terrain changes:**

1. Choose **Simulation > Dynamic Terrain > Reset Changes**. The Reset Dynamic Terrain Changes dialog box opens. The cursor changes to a green circle that represents the area within which changes can be made.
2. Optionally, change the radius.
3. Click on the terrain or enter the location where you want to reset changes. If there are any dynamic terrain models in that area they are listed in the list window.
4. For each option listed that you want to change, select Reset in the list.
5. Click OK.

8.4. Removing Dynamic Terrain Damage

You can remove terrain damage caused by munitions or by hand. Remove Damage does not affect changes to doors and windows.

➤ **To remove dynamic terrain damage:**

1. Choose **Simulation > Dynamic Terrain > Remove Damage**. The Remove Dynamic Terrain Damage dialog box opens. The cursor changes to a green circle that represents the area within which damage will be removed.
2. Click on the terrain or enter a location in the dialog box.
3. Optionally, specify the radius around the location within which to remove damage.
4. Click OK.

8.5. Resetting All Changes in Dynamic Terrain

In a dynamic terrain, you can reset all changes — that is, remove damage like craters and ruined buildings, as well as changes like persistent tracks or doors or windows that have been opened or closed. The terrain returns to its original state, not the state of the last scenario load.

➤ **To return a scenario's terrain to its original condition:**

1. Choose **Simulation > Dynamic Terrain > Reset All Changes**.
2. Click OK.

8.6. Creating Dynamic Terrain Areas

If you disable the global dynamic terrain processor when you create a scenario, you can still enable dynamic terrain within dynamic terrain areas. (Global dynamic terrain processor is an option on the Advanced tab of the New Scenario dialog box. For details, please see "Creating a Scenario" in the *VR-Forces Users Guide*.)

Dynamic terrain areas are tactical graphics that are available on the Tactical Graphics Palette. You create them just like any other areal tactical graphic.

8.7. Dynamic Terrain Switch Types

Table 18 lists the dynamic terrain switches supported.

Table 18. Damage types and values

Type	Allowable Values	Description
damage	none, slight, moderate, destroyed	General terrain damage.
damage_ structure	none, slight, moderate, destroyed	Damage specific to houses and buildings.
door	open, closed	The state of a door.
window	open, closed	The state of a window.

You can choose to tag switches used for dynamic terrain with a position (@dis switchposition x,y,z) to use as the position of the switch (in model coordinates) if it exists, rather than using the bounding sphere center generated by OSG.

8.8. Craters

VR-Forces can add craters to the terrain in response to munition impact. The size of the craters can vary depending on the soil type and munition type. This type of dynamic terrain only works on osgEarth terrains in parts of the terrain defined by **heightfield** elements. It does not work in the high fidelity inset areas.

When a crater is created in a navigation area, the navigation data regenerates. Entities that try to move through the crater will follow its contours. If an entity was on top of the area where the crater formed, it falls into the hole.

The ability to create craters is configured in the *dynamic-terrain.sysdef* file and the *detonationDeformationParams.mtl* file. Both are part of the base SMS. You enable or disable craters in *dynamic-terrain.sysdef* by setting the **craters-enabled** parameter to True or False. You can specify the maximum number of craters that can exist concurrently in a scenario by setting the **crater-max-limit** parameter in *dynamic-terrain.sysdef*. *detonationDeformationParams.mtl* configures the crater size and image based on soil type and munition type.



9. Configuring Feature Data in VR-Forces

This chapter explains how to update the feature data configuration file, *featureconfig.txt*, so that the simulation engine can understand feature data. It is specific to VR-Forces.

9.1. Introduction 140
9.2. Specifying Feature Names for the Front-End and Back-End 140
9.3. Named Queries 141
9.4. Feature Configuration Attributes 141
9.5. Attribute Transformations 142
9.5.1 Transformations as Aliases 142
9.5.2 Transformations as Derived Attributes 142
9.5.3 Transformations as New Attributes 142
9.5.4 Writing Multiple Transforms to Account for Possible Failures 143
9.6. Source-Specific Configuration 143
9.6.1 Loading Source-Specific Configuration Files 143
9.6.2 Redefining Queries 143
9.7. Extracting Feature Data from GDB to Shapefiles 144
9.8. Converting S-57 Feature Data to Shapefiles 144
9.9. Configuring Aggregate-Level Movement Restrictions 145
9.10. Dynamic Features 146

9.1. Introduction

VR-Forces can import feature data, such as shapefiles, that provide information about roads, buildings, waterways, and other terrain features. The front-end and back-end handle feature data differently. The front-end must render the data visually. The back-end must understand how features are to be used as it simulates objects on the terrain.

To use feature data correctly for simulation, the VR-Forces simulation engine must know some things about the feature data. For example, when an areal feature is loaded from a feature source, VR-Forces needs to know how to determine whether that feature is a building footprint, lake boundary, forest boundary, or something else entirely.

The simulation engine determines how features should be used by sending queries to the terrain database. The queries that VR-Forces uses are defined in *./appdata/settings/featureconfig.txt*. Feature data is defined using attributes. Attributes have names and values. If the feature data that you are using uses the attributes that VR-Forces knows about, it can process your terrain correctly. If not, you must edit *featureconfig.txt* so that VR-Forces can understand your data. The configuration file is commented. This chapter provides additional documentation about querying feature data.

9.2. Specifying Feature Names for the Front-End and Back-End

As described in "3.6. Adding Feature Layers to an earth File" on page 70, feature layers are added to an earth file using the **model** element. The front-end uses the **name** attribute of the **model** element to determine what kind of feature layer is being added. In many cases, the back-end can use the same name to determine how to handle the feature data. However, there are some cases where the front-end needs to handle feature data differently from the back-end and needs to use a name that would cause the back-end to process the data incorrectly.

The back-end determines the name of the feature layer based on the following criteria:

- If features come from the server:
  - If the **vrfsim:layer** element is included in the model, the back-end uses the name specified by **vrfsim:layer**.
  - If **vrfsim:layer** is not included, the back-end uses the value in the **name** attribute.
- If the features come from a local file (that is, they use the “ogr” driver):
  - If the **vrfsim:layer** element is included in the model, the back-end uses the name specified by **vrfsim:layer**.
  - If **vrfsim:layer** is not included, the ogr library picks the name.

The supported feature names are specified in *./appdata/settings/featureconfig.txt*.

The following earth file snippet specifies the name “buildings”, which both the front-end and back-end would use:

```
<model name="buildings" driver="some_driver">
   <features> ... </features>
</model>
```

The following earth file snippet specifies the name “special_buildings” and the **vrfsim:layer** “buildings”:

```
<model name="special_buildings" driver="some_driver">
   <features> ... </features>
   <vrfsim:layer>buildings</vrfsim:layer>
</model>
```

In this case, the front-end loads feature data based on the name “special_buildings”. The back-end loads feature data using the name “buildings”.

The following earth file snippet specifies the ogr driver. The driver would choose the feature names.

```
<model name="special_buildings" driver="ogr">
   <features> ... </features>
</model>
```

For more information about the **vrfsim:** syntax, please see "3.17. Differential Processing of Elements and Attributes" on page 87.

9.3. Named Queries

Named queries are used throughout the VR-Forces code and configuration to determine what features are. For example, the following named query is evaluated on a feature to determine whether it should be treated as a building:

```
MAK_BUILDING:  (mak_layer="buildings") OR (mak_layer="BuildingsP") OR
(DTDESCRIPTION = "Building") OR (DTFIDCODE = 181)
```

In this query definition, **mak_layer**, **DTDESCRIPTION**, and **DTFIDCODE** are attribute names. If a feature has one of these attributes with the specified value, VR-Forces knows to treat the feature as a building.

Please see *./appdata/settings/featureconfig.txt* for information about query syntax.



Queries that begin with “MAK_” are referred to either in a simulation model set configuration, or in the VR-Forces code. Do not change the names of these queries.

9.4. Feature Configuration Attributes

When VR-Forces loads a feature in the simulation engine, it adds attributes that indicate where the feature came from. VR-Forces may add one or more of the following features:

- MAK_LAYER. This attribute represents the source of the data — either the name of the file for features loaded from a local file, or the name of the feature layer in an earth file. In the example in "9.3. Named Queries" above, MAK_LAYER

would match a file named *BuildingsP.shp*, for example, or an earth file feature layer named “buildings”.

- MAK_EARTH_FILE. If the feature is loaded from an earth file, the file it was loaded from.
- MAK_SOURCE_FILE. The file from which you loaded the terrain. This could be an MTF file, an earth file, or another format that VR-Forces can load directly.

9.5. Attribute Transformations

Attribute transformations allow you to configure attributes on features matching a specific query. These are used in VR-Forces to:

- Create common names for attribute concepts, such as “width”, which may be named differently depending on the source.
- Create new attributes whose value is derived from other attributes, which can be modified through configuration instead of in the code.
- Add entirely new attributes to features that match the given query.

9.5.1 Transformations as Aliases

Different sources of feature data often use different names for the same information. For example, a height attribute could be named “height”, “hgt”, “h”, or some other name. You can use attribute transformations to create an “alias” for the different names so that a single attribute name can be used in code and configuration.

For example, the following code defines the attribute **MAK_WIDTH**:

```
Transform ALL: double MAK_WIDTH = [ width | w | wdt | gdb_width | dtwidth | wid2d ]
```

This adds, to each feature matching the named query **ALL**, a new double attribute named **MAK_WIDTH**. The value of **MAK_WIDTH** will be the value of the first attribute found in the bracketed list.

9.5.2 Transformations as Derived Attributes

You can use transformations to create derived attributes – that is, attributes whose value depends on the value of another attribute. For example, the following line defines **MAK_NUM_LANES**, which depends on the width of a road:

```
Transform MAK_ROAD: int MAK_NUM_LANES = MAK_WIDTH / 4
```

This adds a new int attribute named **MAK_NUM_LANES** to each feature matching the named query **MAK_ROAD**. The value of this attribute is based on the value of **MAK_WIDTH**.

9.5.3 Transformations as New Attributes

You can use transformations to add entirely new attributes – for example, to differentiate one way roads from two way roads. In this example, a query for one way road is created, and a new attribute **MAK_ONEWAY** is added to each feature that matches the query:

```
MAK_ONEWAYROAD: MAK_ROAD AND (ONEWAY = "yes" OR ONEWAY = 2)
Transform MAK_ONEWAYROAD: int MAK_ONEWAY = 2
```

By doing this, we can query the attribute **MAK_ONEWAYROAD** on any feature, and its existence will mean that the feature represents a one way road.

9.5.4 Writing Multiple Transforms to Account for Possible Failures

You can write multiple transforms for a particular feature if you anticipate that the preferred transform might fail. A transform might fail if it refers to a feature attribute that is not present in the feature data being processed. If you include multiple transforms in the file, they are processed in order until one is successful.

For example consider the following transforms:

```
Transform MAK_S57_DEPTH_AREAS: double MAK_MIN_DEPTH = DRVAL1
Transform MAK_S57_DEPTH_AREAS: double MAK_MIN_DEPTH = 0
```

This transform configures S-57 depth areas. The first transform uses the value of DRVAL1. If the data does not have the DRVAL1 attribute, it sets the depth to 0.

9.6. Source-Specific Configuration

Since different sources of feature data can have different, even contradictory, definitions of the same data, you can create additional configuration files that exist alongside specific data sources.

9.6.1 Loading Source-Specific Configuration Files

Whenever a feature source is loaded, VR-Forces looks for a configuration file that is specific to that source. The filename that it looks for depends on the source of the data. See *featureconfig.txt* for the details of source-specific configuration file loading by source type.

9.6.2 Redefining Queries

Source-specific configuration files can redefine queries in the main configuration file, so the main configuration file can be kept as simple as possible. In the following example, the terrain loaded points to two different *.shp* files, each with a different concept of a main road.

Test.mtf points to two different Shape files:

- *./SharedData/16/latest/Terrain/Test/Features/RoadL.shp*. Any feature with the attribute "HIGHWAY" set to 1 is a main road.
- *./SharedData/16/latest/Terrain/Test/Features/MainRoads.shp*. All features are main roads.

Here is the definition of a main road in the main configuration file:

```
MAINROAD: MAK_ROAD AND MAK_NUM_LANES >= 4
```

To specify a different definition of **MAINROAD** for each source, two new configuration files can be created:

- *RoadL.mfc* contains the following definition:

  ```
  MAINROAD: HIGHWAY = 1
  ```

- *MainRoads.mfc* contains the following definition:

  ```
  MAINROAD: ALL
  ```

At runtime, these queries are combined, using the **mak_layer** attribute, to create a single definition of the **MAINROAD** query:

```
OR
  AND
      MAK_ROAD
      MAK_NUM_LANES >= 4
  AND
      MAK_LAYER = "RoadL"
      HIGHWAY = 1
  MAK_LAYER = "MainRoads"
```



In this example, it is possible to create this query entirely within *featureconfig.txt*. Here the source-specific configuration is used for readability.

9.7. Extracting Feature Data from GDB to Shapefiles

When you load feature data from a shapefile, it is used more efficiently than if you load it from a GDB file. (For details about loading feature data into a feature layer, see "1.4. Feature Layers" on page 39.) If you have GDB terrains that contain feature data, you can extract that data to shapefiles with the vectorNetworkToShapeFiles utility. The syntax for the utility is as follows:

`vectorNetworkToShapeFiles.exe terrain_file [`*`output_directory`*`]`

where:

- **terrain_file** is the GDB file to read.
- *output_directory* is the directory in which to create the shapefiles. If *output_directory* is not specified then output files are named based on the name of the input file.

9.8. Converting S-57 Feature Data to Shapefiles

S-57 feature data does not have a spatial index, so it loads slowly. By contrast, shapefiles have a spatial index, so they load quickly. Therefore, we recommend that you convert S-57 data to shapefiles. VR-Forces includes a command-line tool, ogr2ogr, that you can use to convert your S-57 data. (OGR is a sublibrary of GDAL, which VR-Forces uses to load feature data. Complete documentation is available at http://www.gdal.org/ogr/drv_s57.html and http://www.gdal.org/ogr2ogr.html.)

S-57 files have many layers of data. A shapefile only contains one layer of data. So, to convert your S-57 data, you must create a shapefile for each layer of data that you want to convert. Since VR-Forces does not use all of the data in an S-57 file, this is a simpler process than it might sound.

Before running ogr2ogr, set the following environment variables:

```
S57_CSV = <full_path_to>/appData/gdal_data
```

(The environment variable should specify a full path. If you use a relative path, it is relative to the path from which you run ogr2ogr, not from the VR-Forces root. If you run ogr2ogr from *./appData/gdal_data*, you do not have to set the variable.)

```
OGR_S57_OPTIONS = "RETURN_PRIMITIVES=ON,RETURN_LINKAGES=ON,LNAM_
REFS=ON"
```

This setting works for the layers that VR-Forces needs. If you want to convert other layers, you will have to research the proper environment variable settings.

The command syntax is:

```
ogr2ogr -skipfailures [-nlt type] output_fileinput_filelayer
```

where:

- **-nlt***type* is the geometry type of the output layer. *type* can be NONE, GEOMETRY, POINT, LINESTRING, POLYGON, GEOMETRYCOLLECTION, MULTIPOINT, MULTIPOLYGON or MULTILINESTRING. Add "25D" to the end to make these 3D points.
- *input_file* is the input data source (the S-57 file).
- *output_file* is the name of the output file (the layer name with *.shp* at the end). All output files should be placed in the same directory. This simplifies configuring the earth file that references them.
- *layer* is the name of the layer from the source to convert.

The following example commands convert the layers that VR-Forces needs:

```
ogr2ogr -skipfailures SOUNDG.shp S57FILE SOUNDG
ogr2ogr -skipfailures -nlt POLYGON LNDARE.shp S57FILE LNDARE
ogr2ogr -skipfailures DEPARE.shp S57FILE DEPARE
ogr2ogr -skipfailures M_COVR.shp S57FILE M_COVR
```

There will be some output errors for each of these because they have some attributes that cannot be represented in shapefiles, such as lists of numbers and strings.

9.9. Configuring Aggregate-Level Movement Restrictions

In aggregate-level scenarios, terrain features can affect movement of simulation objects. This is configured by the relationship of entries in *featureconfig.txt* and a simulation object's movement system.

In *featureconfig.txt*, the terrain mobility configuration section specifies terrain restriction levels, as follows:

```
-- Terrain Mobility Configuration
--
MAK_UNRESTRICTED_TERRAIN:  MAK_ROAD
MAK_RESTRICTED_L1_TERRAIN: NONE
--MAK_RESTRICTED_L2_TERRAIN: mak_layer = "forest"
MAK_RESTRICTED_L2_TERRAIN: NONE
--MAK_IMPASSABLE_TERRAIN:    MAK_OBSTACLE
MAK_IMPASSABLE_TERRAIN:    NONE

MAK_INFANTRY_UNRESTRICTED_TERRAIN:  MAK_UNRESTRICTED_TERRAIN
MAK_INFANTRY_RESTRICTED_L1_TERRAIN: MAK_RESTRICTED_L1_TERRAIN
MAK_INFANTRY_RESTRICTED_L2_TERRAIN: MAK_RESTRICTED_L2_TERRAIN
MAK_INFANTRY_IMPASSABLE_TERRAIN:    MAK_IMPASSABLE_TERRAIN
```

The restriction levels are referenced in the terrain-mobility block in a system definition file. The following code is from *ground-aggregated-movement.sysdef*:

```
(terrain-mobility
  (entry
      (query-name "MAK_UNRESTRICTED_TERRAIN")
      (speed-factor 1)
      (priority 1)
  )
  (entry
      (query-name "MAK_RESTRICTED_L1_TERRAIN")
      (speed-factor 0.75)
      (priority 2)
  )
  (entry
      (query-name "MAK_RESTRICTED_L2_TERRAIN")
      (speed-factor 0.4)
      (priority 3)
  )
  (entry
      (query-name "MAK_IMPASSABLE_TERRAIN")
      (speed factor 0)
      (priority 4)
  )
)
```

9.10. Dynamic Features

VR-Forces uses feature data for path planning, obstacle avoidance, and other interactions between simulation objects and the terrain. In addition to the features that you can import into a terrain using shapefiles, VR-Forces can treat simulation objects and objects such as cultural features, as features and insert them into a feature layer.

For example, in aggregate-level scenarios, users and simulation objects can create combat engineering objects (CEOs), such as minefields, fortified lines, and bridges, that can impede or facilitate unit movement. VR-Forces implements this capability by dynamically treating combat engineering objects as if they were terrain features.

VR-Forces maps entities to features in *./appData/settings/featureconfig.txt*. The following is an example of some mappings:

```
  -- Dynamic Features
  --

  MAK_ROUTE:            {"1:17:0:0:2:-1:-1:-1"}
  MAK_AREA:             {"1:18:0:0:1:-1:-1:-1"}
  MAK_DYNAMIC_OBSTACLE: {"1:18:0:0:2:-1:-1:-1"}

  MAK_VRF_Roads:                     {"1:17:1:0:62:11:0:0"}
  MAK_VRF_Bridges:                   {"1:17:1:0:62:12:0:0"}

  MAK_VRF_Dynamic_Paths: (MAK_VRF_Roads OR MAK_VRF_Bridges) AND
Percent_Complete >= *Minimum_Completion_Percentage_For_Effect

  MAK_VRF_Engineering_Object_Points: {"1:16:1:-1:62:-1:-1:-1"}
  MAK_VRF_Booby_Trap:                {"1:16:1:0:62:1:0:0"}
  MAK_VRF_Unexploded_Ordnance:       {"1:16:1:0:62:2:0:0"}
  MAK_VRF_Engineering_Object_Lines:  {"1:17:1:-1:62:-1:-1:-1"}
  MAK_VRF_Abatis:                    {"1:17:1:0:62:1:0:0"}
  MAK_VRF_Anti_Tank_Ditch:           {"1:17:1:0:62:2:0:0"}
  MAK_VRF_Barbed_Wire:               {"1:17:1:0:62:3:0:0"}
```



10. Configuring Airport Settings in VR-Forces

Aerodrome and runway objects allow you to give aircraft realistic places to take off and land. You can create aerodromes based on information in the terrain file or using tactical graphic objects. You can then control the airport and aerodrome settings either manually or as part of a plan. This chapter describes the settings that are available.

10.1. Overview of Airports 150
10.2. Airport Settings Panel 150
10.2.1 Centering on an Airport 150
10.2.2 Setting Aerodrome Lights 150
10.2.3 Setting Runway Conditions 152
10.3. Setting Runway Lights in a Plan 152

10.1. Overview of Airports

You can control the settings of airports, either as part of a plan or manually, with the Airport Settings Panel. The changes you make are reflected in any simulation connected to the exercise.

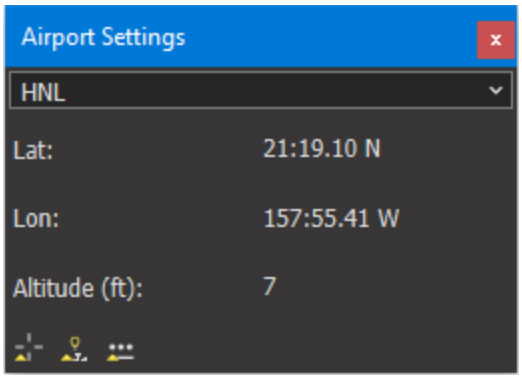
Airports are a layer in MAK earth and streaming terrains. Airport features are part of the vector files that are distributed with VR-Forces. These files are originally from X-Plane and, while they are good for many of the airports in the world, they may not be entirely accurate. You can use GIS tools to edit the files and change, add, or remove airport features as needed.

10.2. Airport Settings Panel

You can locate airports and control the settings from the Airport Settings Panel.

➤ To open the Airport Settings Panel, choose **View > Airport Settings Panel**. The Airport Settings Panel opens (Figure 31).

Figure 31. Airport Settings Panel



10.2.1 Centering on an Airport

You can select and center on an airport from the Airport Settings Panel Figure 31 above.

➤ **To center on an airport:**

1. In the Airport list, select the airport that you want to center on. You can filter the list by entering part of the name or code.
2. Click Center on Airport ( ). Your window focuses on the selected airport.

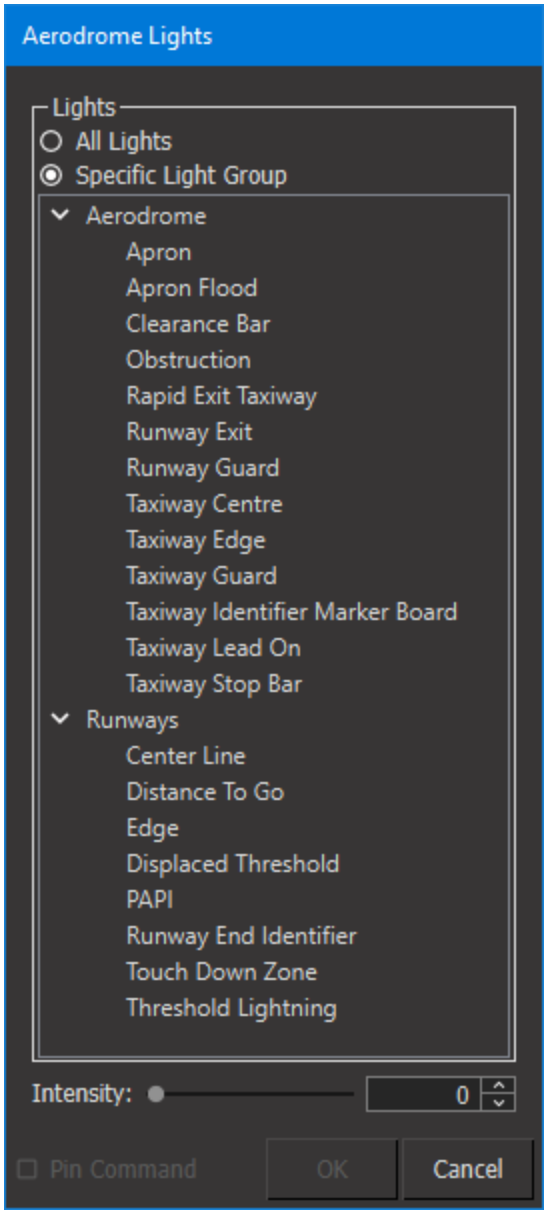
10.2.2 Setting Aerodrome Lights

You can adjust the to adjust the lighting for all aerodrome or runway lights at the specified aerodrome. The lighting changes are not visible in any views of the exercise, but the changes are published to the network and set the state on the published objects. You can also see the light settings in the Information dialog.

➤ **To change an airport's runway light settings from the Airport Settings Panel:**

1. In the Airport Settings Panel, specify the airport.
2. Click Runway Light Settings ( ). The Set Aerodrome Lights dialog box opens (Figure 32).

Figure 32. Aerodrome Lights dialog box



3. Choose whether you want to set the intensity for all of the lights in the aerodrome or a specific light group.
4. If you choose to control a specific light group, select the group that you want to set.

5. Set the intensity using the slider or by entering a value in the box. The intensity is a percentage, from 0 (off) to 100%.
6. Click OK.

10.2.3 Setting Runway Conditions

You can specify the conditions of a runway. The runway conditions are not visible in any views of the exercise, but the changes are published to the network and set the state on the published objects. You can also see the runway condition in the Information dialog box.

➤ **To set runway conditions from the Airport Settings Panel:**

1. In the Airport Settings Panel, specify the airport.
2. Click Runway Condition (). The Runway Condition dialog box opens.
3. Select a condition.
4. Click OK.

10.3. Setting Runway Lights in a Plan

As part of a plan, you can change the lighting for one runway or all runways at an airport in procedural terrain. The lighting will be visible in any 3D views of the exercise.

➤ **To change an airport's runway light settings in a global or local plan:**

1. Open a global or an individual plan.
2. Choose **Commands > Set Runway Lights**. The Set Runway Lights dialog box opens (Figure 32 on the previous page).
3. In the Airport list, select the airport for which you want to set the lights. You can filter the list by entering part of the name or code.
4. In the Runway list, select a particular or runway or ALL to set the lights for all runways at the airport.
5. In the Intensity list, select the intensity of the lights. You can choose from five levels, or you can turn the lights off.
6. Click OK. The command is added to the plan.



11. Terrain Performance and Configuration

This chapter covers miscellaneous issues that affect terrain performance and functioning.

11.1. Configuring File Caching 154
11.1.1 Specifying the Location to Cache Terrain Server Data 155
11.1.2 Caching earth File Terrain Data 155
11.1.3 Enabling Texture Compression 156
11.1.4 Clearing the File Cache 157
11.2. Configuring Paged and Streaming Terrains 157
11.3. Debugging earth Files 157
11.3.1 Viewing a Terrain in Debug Mode 158
11.4. Using Shader-based Effect Maps 159
11.4.1 Composite Effects Maps 161
11.4.2 Debugging Shaders 162
11.4.3 Reloading Shaders 163
11.5. Indirect Rendering of Terrain 164
11.5.1 Scaling the LOD Distance for Indirect Rendering 164
11.5.2 Setting the LOD Transition Percentage 165
11.5.3 Disabling Indirect Rendering for Terrain 165
11.6. Specifying the Maximum Number of Paged LODs 166
11.7. Indirect Rendering for osgEarth Layers 166
11.8. Displaying Terrain Intersection Lines 166
11.9. Displaying DDS Textures Correctly 167
11.9.1 Flipping DDS Textures Globally 168
11.10. Enabling Occlusion Queries 168
11.11. Converting OpenFlight File Texture Data 169
11.12. Drawing Bounding Volumes for Scene Graph Nodes 170
11.13. Masking the Ocean 171
11.14. Debugging the VR-Forces Simulation Terrain 172
11.14.1 Starting the Simulation Terrain Debugging Tool 172

11.14.2 Refreshing the Debug Display 173
11.14.3 Viewing Terrain Skin, Normals, and Collision Objects 173
11.14.4 Viewing Simulation Terrain Features 176
11.14.5 Displaying the NavMesh 178
11.14.6 Displaying the Query Area 179
11.14.7 Displaying Elevation Tiles 180
11.14.8 Displaying Recent Intersections 181
11.14.9 Displaying Page-in and Page-out Nodes 182
11.14.10 Enabling Auto Refresh 183
11.14.11 Simulation Terrain Debugging from the Back-End Console 183

11.1. Configuring File Caching

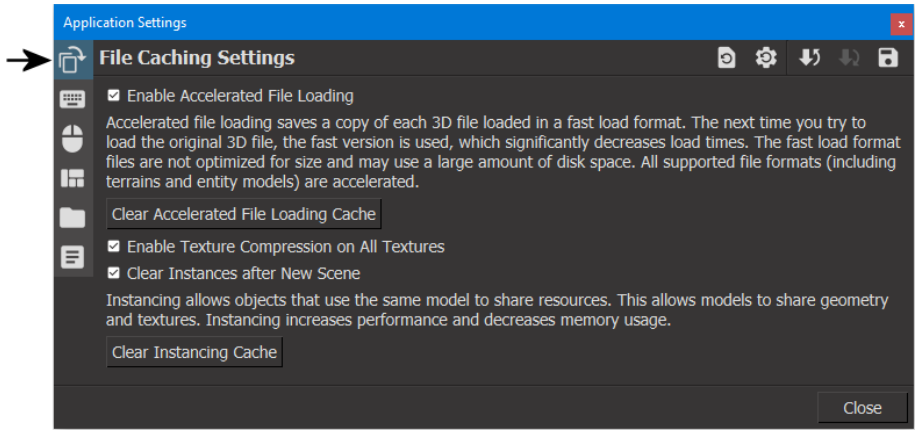
When file caching is enabled, MAK ONE applications store information about the terrain and models in a format that enables it to be loaded quickly. This improves loading time the next time the cached objects are loaded. If caching is disabled, terrains and models must be loaded from their native formats,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

When an application processes a request to load a file, such as an external reference in a terrain, it checks the cache and the directory of the requested file for an MEDF version of the file. If it does not exist, it loads the file in the original format.

➤ **To enable or disable file caching:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 33).

Figure 33. File Caching Settings page



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

11.1.1 Specifying the Location to Cache Terrain Server Data

In VR-Vantage and VR-Forces, when file caching is enabled, the application caches data from terrain servers in the default locations. For both applications, you can override this by specifying an alternative caching location using the OSGEARTH_CACHE_PATH environment variable. In VR-Forces, you can also use the VRFSIM_OSGEARTH_CACHE_PATH environment variable. In VR-Vantage, the OSGEARTH_CACHE_PATH setting overrides the **cache** element in an earth file. In VR-Forces, these environment variables interact as follows:

- If you only set OSGEARTH_CACHE_PATH, then the front-end and back-end both use it. It overrides the **cache** element in the earth file.
- If you set OSGEARTH_CACHE_PATH and VRFSIM_OSGEARTH_CACHE_PATH then the front-end uses OSGEARTH_CACHE_PATH and the back-end uses VRFSIM_OSGEARTH_CACHE_PATH.
- If you only set VRFSIM_OSGEARTH_CACHE_PATH, the back-end uses that value and the front-end uses the default cache location or the location specified in the earth file.

11.1.2 Caching earth File Terrain Data

You can cache the terrain data in earth files to speed up loading terrain and to allow you to load terrains without connecting to a terrain server, either locally or over the internet. You can generate cached data through the GUI. You can cache imagery and elevation. (Feature data is not cached.)

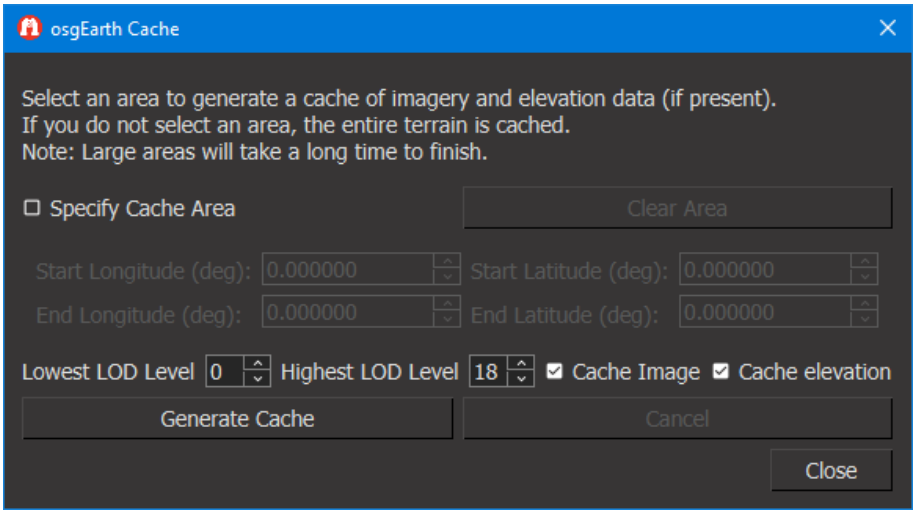


It can take a very long time to generate the cache for large areas.

➤ **To generate cached data:**

1. Load a terrain that uses an earth file.
2. Choose **View > Terrain Editor Panel**. The Terrain Editor opens.
3. Select the Tools page.
4. Click the Generate Cache button. The osgEarth Cache dialog box opens (Figure 34).

Figure 34. osgEarth Cache dialog box



5. If you want to generate cached data for the entire terrain, make sure that the Specify Cache Area check box is cleared.
6. If you want to generate cached data for a portion of the terrain:
   a. Select the Specify Cache Area check box.
   b. Specify the latitude and longitude of a rectangle that defines the area you want to cache. You can do this by adjusting the sliders or entering values for the four values or you can draw a rectangle on the terrain.

      To draw a rectangle on the terrain, click the left mouse button and release it to specify the upper left corner of the rectangle. Drag the mouse to define the cache area. Left-click again to specify the lower right corner.
7. Optionally specify the lowest LOD level and the highest LOD level.
8. Click Generate Cache.

11.1.3 Enabling Texture Compression

When texture compression is enabled, textures are converted internally to a compressed format (DXT). Compressing textures decreases memory use and increases rendering performance.

> The textures provided with MAK ONE applications are already compressed. This setting is useful if you are adding your own data.

➤ **To enable or disable texture compression:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 33 on page 154).
3. Select or clear the Enable Texture Compression On All Textures check box.

11.1.4 Clearing the File Cache

You can clear the file cache if you need to free disk space.

The cache is normally configured to be in *./appData/cache*. You can delete files from this directory if you want to. The most likely reason to do this is if you load a terrain and realize that it should have had its DDS textures flipped, but has now been saved with them oriented the wrong way. You could delete the cache, then reload the terrain and toggle DDS Textures to the opposite setting.

➤ **To clear the file cache:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 33 on page 154).
3. Click Clear Accelerated File Loading Cache.

11.2. Configuring Paged and Streaming Terrains

You can improve performance for some terrains by configuring the following environment variables:

- OSG_MAX_PAGEDLOD. OSG_MAX_PAGEDLOD specifies the target maximum number of paged LODs to maintain. Default: 200. Set it to a lower value if you need to reduce memory consumption when using paged terrains. (This variable only affects the front-end.)
- GDAL_CACHEMAX. GDAL_CACHEMAX specifies the maximum cache size for streaming terrain. The default value is 200 MB, which is often adequate. However, if you are using a highly detailed terrain, increasing the cache size to a larger value, such as 2000, can significantly improve performance.

11.3. Debugging earth Files

Earth files can contain many layers of elevation, imagery, and feature data. When you are debugging performance problems for a terrain that uses an earth file, it can be helpful to turn off individual layers and see how that affects performance. You can do that in the Terrain Editor. For details about how to enable and disable layers, see "4.3.1 Enabling and Disabling Layers" on page 101.

In addition to seeing how enabling and disabling a layer affects performance, you can test the effect of changing other properties. The properties you can change for all layers are as follows:

- Visible. Enable or disable display of the layer.
- Opacity. The opacity of the layer. The range of values is 0.0 through 1.0.
- Minimum Visible Range. The minimum camera range at which this layer is visible.
- Maximum Visible Range. The maximum camera range at which this layer is visible.

- Lighting. Enable or disable OpenGL lighting on the model. This setting only applies to model layers.

For ground cover layers, the properties described in the previous list apply. You can also set the following properties for each type of ground cover in a layer:

- Wind Factor. Affects the extent to which some types of ground cover respond to wind. For example, trees that sway in the wind.
- Density. Specifies the number of land cover instances that will render in a given area. Range 0.0 through 1.0.
- Fill. Percentage of land that this layer's instances cover. Lower values result in a patchy appearance.
- Maximum Distance. The maximum viewing distance from the camera.
- Brightness. The level of brightness to apply. Default 1.0. (Some terrains provided with MAK ONE applications override the default.)
- Contrast. The level of contrast to apply. Default: 0.5.

You can only disable lighting for model layers (features area specified using the **model** element).

The properties described in the previous paragraphs are displayed in the Runtime Options window on the Terrain Editor. For details about viewing and changing properties, see "4.3.2 Editing the Properties of a Layer" on page 102.



When you enable or disable layers, the changes are not saved to the earth file. To make changes permanent, you must edit the earth file.

In VR-Forces, if you want to filter out earth file layers and have the changes persist, use the Earth Layers page. For details, see "Filtering Earth File Layers" in the *VR-Forces Users Guide*.

11.3.1 Viewing a Terrain in Debug Mode

You can view an osgEarth terrain in debug mode. In debug mode, terrain tile boundaries are drawn. The Properties window of the Terrain Editor displays additional information.

Figure 35. Debug mode terrain tiles



This information is more easily viewed in Plan View mode than Stealth mode. As you zoom in and out, the terrain tile boundaries redraw.

➤ **To view a terrain in debug mode:**

1. Load the osgEarth terrain that you want to view.
2. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.
3. In the Terrain Content window, select the line immediately below the topmost line in the window. This line is probably the name of the terrain.
4. In the Properties window, select the Debug Mode check box. The window displays terrain tile boundaries and the Properties redisplay.
5. To exit debug mode:
    a. Select the name of the terrain in the Terrain Content window. The Properties window changes to show the Debug Mode check box.
    b. Clear the Debug Mode check box.

11.4. Using Shader-based Effect Maps

MAK ONE applications support several types of shader-based effect texture maps. Shaders are computer programs that run on the graphics processing unit (GPU). MAK ONE applications make extensive use of them for lighting effects.

Effect maps are raster images that apply highly realistic textures to the terrain and models. By applying different types of effect maps to terrain and models, you can improve the visual qualities of your simulation without the overhead of high polygon counts.

MAK ONE applications support the following types of shader-based effect maps:

- Normal, or bump, maps. 3 Channel. Normal maps give terrains the appearance of relief, such as a rocky landscape.
- Specular maps. 3 Channel. Specular maps affect the highlight color of objects.
- Ambient occlusion maps. 1 Channel. Ambient occlusion maps model areas that do not receive direct light, such as cracks and crevices and shaded areas of terrain and models. These areas are lit only by ambient light.
- Reflection maps. 4 Channel. Reflection maps affect the reflectivity of surfaces, such as windows. Reflection maps reflect objects in the sky, not the terrain.
- Emissive maps. 4 Channel. Emissive maps control the emissivity of whatever they are applied to based on the current ambient light values. The alpha channel is an ID that is set from 0-255. All the pixels that should light together (like a window, and perhaps a surrounding sill) should have the same ID. When the ambient light level drops below a given ID level, those pixels are emissive. Channel 2 (green) is used to indicate the intensity of the emissivity. These textures need to use a lossless compression scheme, such as PNG. *Little Pond (online).mtf* has sample emissivity textures on many of the houses.
- Nightmaps maps. 3 Channel. Used when ambient light is low. Used in conjunction with emissive maps to control the brightness of pixels.
- Emissive Nightmap maps. 3 Channel. Used when ambient light is low. Makes all pixels glow 100%. An emissive map is not needed, but you cannot control how bright or dark the pixels are.
- Lightmap maps. 3 Channels. Lightmaps store precomputed lighting for surfaces in a scene.
- Sensor Map. 3 Channel. Texture for adding an intensity to all three color channels based on CamerFX's sensor mode. The red channel is added when the sensor mode is an NVG sensor type. The green channel is added when the sensor type IR and the power plant is turned off for a particular entity. The blue channel is added when the sensor type is IR and the power plant is turned on for a particular entity.
- Gloss maps. 1 Channel. Gloss maps control how wide or narrow the specular highlight appears.
- Reflection maps. 4 Channel. Reflects the environment onto an object.
- Flow maps. 3 Channel. Flow maps define direction-based distortion, such as water flow.
- Snow Mask maps. 1 Channel. Controls how much and how fast snow accumulates. If the value is 0, then the snow accumulates at the full road rate. If it is 1, then the snow never accumulates.
- Rain Mask effect maps. 1 Channel. Controls whether an area will have puddles and the amount of accumulation required before puddles appear. If the value is less than one, then puddles will display. If the texture has 0 for a particular texel, then puddle display at the lowest accumulation. If the value is closer to 1, the greater the accumulation must be before puddles display.

You can include effect maps as part of an OpenFlight or FBX model or terrain or as raster maps (on the Raster Maps page of the Terrain Settings dialog box). MAK ONE applications use a naming convention for effect map files that lets it process the files properly without any additional configuration. To specify the type of map file, use the following naming conventions:

- Normal map: *filename_NML.png*
- Ambient Occlusion map: *filename_AO.png*
- Specular map: *filename_SPC.png*
- Reflection map: *filename_RFL.png*
- Emissive map: *filename_EMM.png*
- Emissive Nightmap: *filename_NGT.png*
- Lightmap: *filename_LMM.png*
- Sensor map: *filename_SNR.png*
- Gloss maps: *filename_GLS.png*
- Reflection map: *filename_RFL.png*
- Nightmap: *filename_NTE.png*
- Flow map: *filename_FLO.png*
- Snow Mask map: *filename_SNO.png*
- Rain Mask map: *filename_RAI.png*



You can use up to seven effect maps per texture. If you need more than that, you must use a composite effects map.

The map code must be uppercase on Linux.

To see the effect of effect maps, you must enable lighting. For details, please see the chapter:

- *Lighting Effects* in the *VR-Vantage Users Guide*
- *Lighting Effects* in the *VR-Forces Users Guide*

11.4.1 Composite Effects Maps

Effects maps can be composited into a single file using channels. The name of the file represents each effects map in its respective channel, Red, Green or Blue (for example, House_GLS_SPR_AO_REF.png - Glossy, Specular, Ambient Occlusion and Reflection composite map). Only one of the textures can be a composite texture, but all files have to use the Composite Effects Maps naming convention.

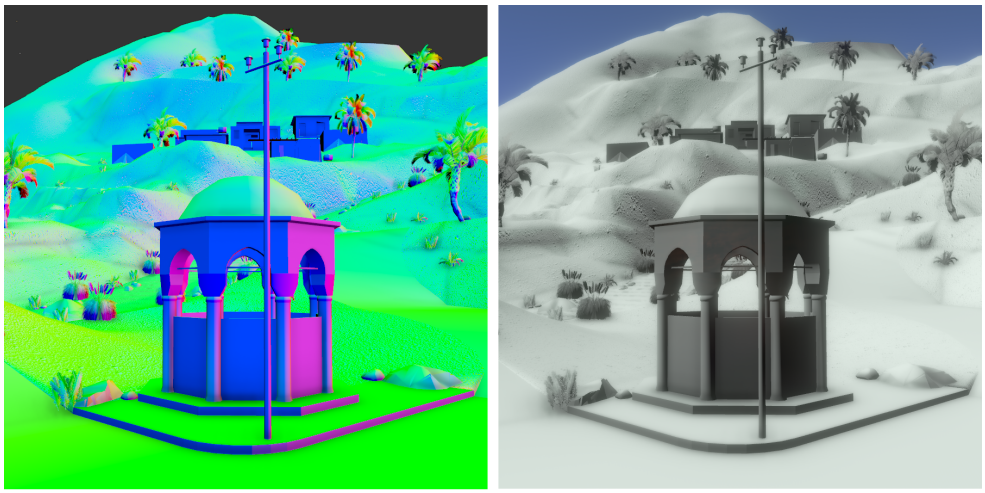
- NML: 3 Channel Normal map
- SPR: 1 Channel Specular map
- EMS: 1 Channel Emissive map
- LMM: 3 Channel Lightmap (supported for CDB)

- GLS: 1 Channel Gloss map
- AO: 1 Channel Ambient Occlusion map
- REF: 1 Channel Reflection map
- NGT: 3 Channel Emissive Nightmap map
- NTE: 3 Channel Nightmap map
- FLO: 1 Channel Flow map

11.4.2 Debugging Shaders

VR-Vantage and VR-Forces provide some shader debugging capabilities within its lighting shaders. The shaders are instrumented such that setting different flags (through uniforms) changes the behavior of the shader. Typically it changes fragment colors based on some parameter or parameters. Figure 36 shows a view of VR-Village with two different shader debug settings.

Figure 36. Shader component visualization

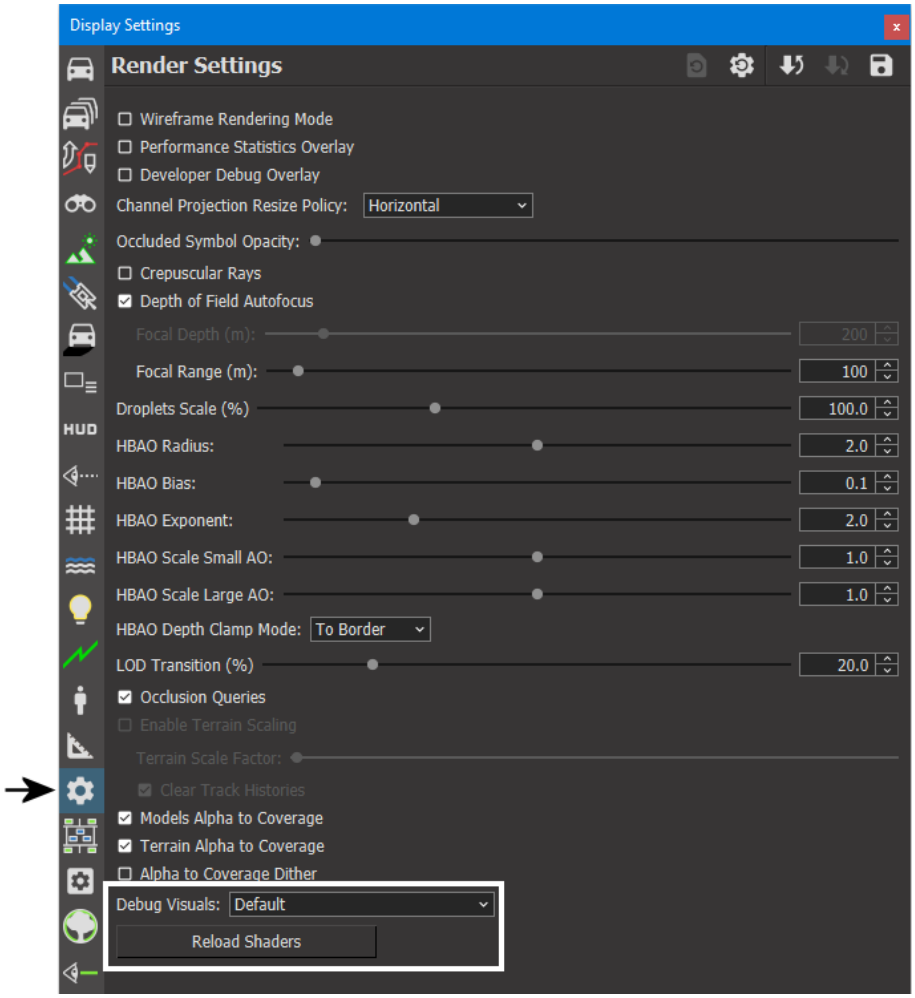


Bumpmap Normals | Diffuse Light

➤ **To debug shaders:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 37).

Figure 37. Render Settings page



3. Select an option from the Debug Visuals list. (If you have SensorFX installed, the list includes options for visualizing materially classified textures.)

11.4.3 Reloading Shaders

Developers who are fine tuning shaders for terrain and object models do so by editing text files that configure the shaders. They can load their new shaders without shutting down the application. Compile failures are reported in the console. We recommend that you display it while debugging.

➤ **To dynamically reload shaders:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 37 above).
3. Click Reload Shaders.

11.5. Indirect Rendering of Terrain

Indirect rendering is supported for simple OpenFlight terrains loaded as a terrain patch. Indirect rendering improves rendering performance by combining geometry into fewer draw calls. It can combine geometry better than instancing or clustering because it does not require textures to be combined in an atlas or the geometry to have the same vertex data. It is enabled by default for the terrains that use indirect supported features. For details about enabling, disabling, and configuring indirect rendering, see:

- "1.2. Adding Elevation Data to a Terrain" on page 37
- "11.5.1 Scaling the LOD Distance for Indirect Rendering" below
- "18.15. Disabling Indirect Rendering for an OpenFlight Model" on page 289
- "3.5. Processing Streamed Data" on page 69
- "2.1. Connecting to Terrain Servers" on page 58

Indirect rendering does not effect the rendering of:

- Extracted external references
- Speedtrees
- Triton dynamic ocean
- Silver-lining sky
- osgEarth terrain skin
- Tactical graphics
- TerraPage terrains.

11.5.1 Scaling the LOD Distance for Indirect Rendering

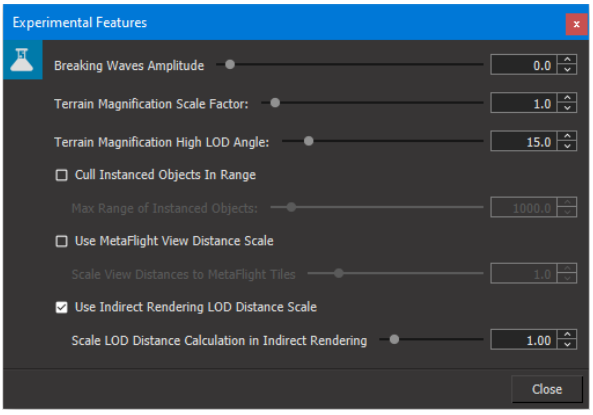
When you use indirect rendering, you can dynamically adjust the distance at which LODs get paged in. If a terrain contains high LODs for objects at distances from the eyepoint at which the details are not visible, you can pay the performance price for the LODs without the visual benefit of high resolution. By scaling the LOD distance, you can cause VR-Vantage to display objects at lower levels of detail without needing to make any changes to the terrain database.

This feature can also help terrain developers to quickly assess the LOD settings for the terrains that they are developing.

➤ **To scale the LOD distance for indirectly rendered objects:**

1. Choose **Settings > Experimental Features.** The Experimental Features dialog box opens (Figure 38).

Figure 38. Experimental Features



2. Select the Use Indirect Rendering LOD Distance Scale check box. The Scale LOD Distance Calculation in Indirect Rendering slider becomes active.
3. Adjust the slider as desired or specify a value in the spin box.
4. Click Close.

11.5.2 Setting the LOD Transition Percentage

When VR-Vantage transitions between LODs, it blends them so that the change in resolution is not visually jarring. You can set the LOD transition percentage to get the most pleasing effects for your terrain. A setting of zero means there is no transition blending.

➤ **To set the LOD transition percentage for indirect rendering:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 37 on page 163).
3. Adjust the Indirect LOD transition percentage slider.

11.5.3 Disabling Indirect Rendering for Terrain

In some situations, you may need to disable indirect rendering. For example, you cannot extract props or water textures while indirect rendering is enabled. You can disable indirect rendering for your current session using ImGui.

➤ **To disable indirect rendering:**

1. Press **F4** to open the ImGui.
2. Expand Terrain Patch Object Settings.
3. Select the Override Terrain Patch Object Settings check box.
4. Clear the Load Next Patch With Indirect check box.
5. Load the terrain (or reload it, if the terrain was already loaded). Indirect rendering is now disabled.

11.6. Specifying the Maximum Number of Paged LODs

You can specify the maximum number of paged LODs that the database pager can keep in memory at any one time. This option only applies to osgEarth terrains.

➤ **To specify the maximum number of paged LODs:**

1. Load the osgEarth terrain that you want to configure.
2. Choose **View > Terrain Editor Panel**. The Terrain Editor panel opens.
3. In the Terrain Content window select the topmost line in the window (Terrain). The properties for this line are displayed in the Properties window.
4. Edit the value for the Max Paged LOD property.
5. Save the terrain.

11.7. Indirect Rendering for osgEarth Layers

Indirect rendering is enabled by default for osgEarth layers. If you want to disable indirect rendering or explicitly set it, add the `<indirect_rendering>false</indirect_rendering>` element to the feature layer. Light point layers are the only layers for which indirect rendering is set to false by default.

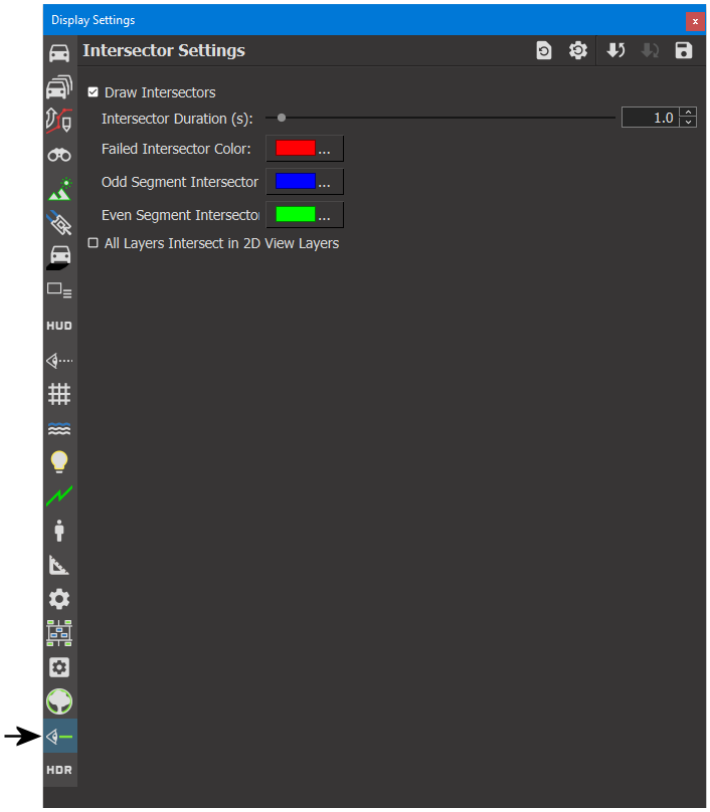
11.8. Displaying Terrain Intersection Lines

You can display terrain intersector lines to help you debug problems with terrain performance.

➤ **To display terrain intersection Lines**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Intersector Settings page (Figure 39).

Figure 39. Intersector Settings page



3. Select the Draw Intersectors check box.
4. Optionally, set the duration of the intersector lines.
5. Optionally change the colors used for intersector lines.

11.9. 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, MAK ONE applications assume that images have the OpenGL orientation. However, sometimes images are not oriented properly and therefore you may have to flip them.

You can change the DDS textures setting as follows:

- On a per-model basis. (For details, please see "18.12. Flipping DDS Textures for a Model" on page 284.)
- When you add a terrain patch. (For details, please see "1.2. Adding Elevation Data to a Terrain" on page 37.)
- As a global setting. (For details, please see "11.9.1 Flipping DDS Textures Globally" on the next page.)

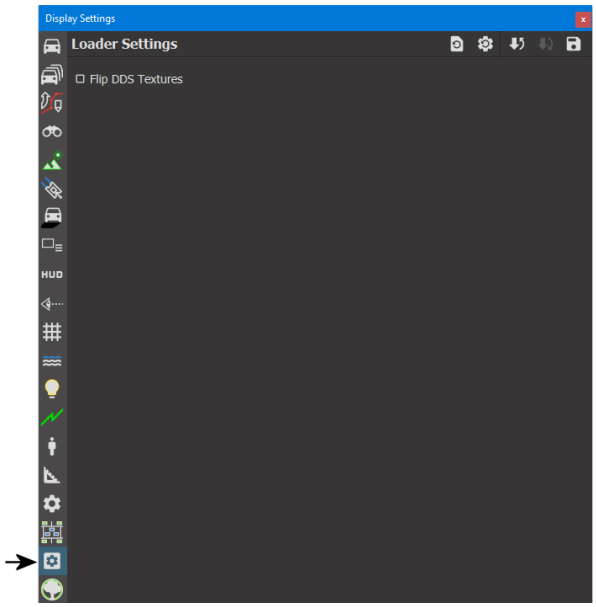
11.9.1 Flipping DDS Textures Globally

You can specify that the DDS textures setting apply to all terrains. This setting is primarily beneficial for paged terrains because you cannot know ahead of time if textures need to be flipped, as you could for a model or terrain patch. If a paged terrain has DDS textures that are upside down relative to the application'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 40).

Figure 40. Loader Settings page



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

11.10. Enabling Occlusion Queries

If you are simulating on a terrain with many buildings, such as in a city, enabling occlusion queries can improve the frame rate. When they are enabled, MAK ONE applications draw bounding boxes around some objects and checks to see whether they are being rendered or are blocked visually (Figure 41). You can view the bounding boxes by enabling occlusion nodes and OC Draw Debug in the Developer Debug Overlay Terrain Patch Object Settings. (For information about Developer Debug Overlay, see "Using the Developer Debug Overlay" in *VR-Vantage Users Guide* or "Using the Developer Debug Overlay" in *VR-Forces Users Guide*.)

Figure 41. Occlusion query bounding boxes



➤ **To enable or disable occlusion queries:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 37 on page 163).
3. Select or clear the Occlusion Queries check box.

11.11. Converting OpenFlight File Texture Data



OpenFlight files do not contain information about soil characteristics, which VR-Forces models need to calculate entity mobility.

To provide soil characteristic data for VR-Forces, when it loads an OpenFlight terrain database, it tries to deduce soil type information from the names of the texture files that are usually associated with OpenFlight files.

By default, VR-Forces tries to determine the soil types by looking for keywords, such as road, street, grass, water, and so on, in the name of the texture file. If this method does not adequately capture soil type information in the file you are importing, you can map the texture files to soil types in the *surfChar.map* file. This file is specific to each terrain. For an example, see

./SharedData/16/latest/TerrainData/Terrain/Makland/Geometry/OpenFlight/surfChar.map.

Table 19 lists the MAK soil types to which you can map the OpenFlight texture files.

Table 19. MAK soil type

asphalt	dryGround	orchards	shallowRiver
boulder	forest	pavedRoad	shallowStream
building	grass	rock	softSoil
cultivatedFields	gravelRoad	sand	swamp
deepLake	mud	shallowLake	tree
deepRiver	ocean	shallowPond	USRailroad
dirtRoad			

The *surfChar.map* file supports two methods of matching textures to soil types, substring match and regular expression. For a substring match, use the **Match** keyword and provide a string to match, as in:

```
Match       HL_water                ocean
```

VR-Forces matches textures with the string **HL_water** to the ocean soil type.

To use a regular expression, use the **RegExprMatch** keyword and provide a regular expression, as in:

```
RegExprMatch ^water.*               ocean
```

VR-Forces matches textures that start with the string **water** to the ocean soil type.

The following is an example *surfChar.map* file:

```
SurfaceCharacteristicMap
{
   Match      HL_water                        ocean
   Match      Water_Ocean_Deep_Blue           ocean
   Match      Tree_Evergreen                  tree
   Match      Tree_Canopy_Evergreen           forest
   Match      Road_Highway_Two-Lane_Asphalt pavedRoad
   Match      Road_Street_Single-Lane         pavedRoad
}
```

The center column in each entry is an OpenFlight texture. The right column is a MAK soil type.



The list of supported soil types is fixed. Therefore, you cannot add new soil types.

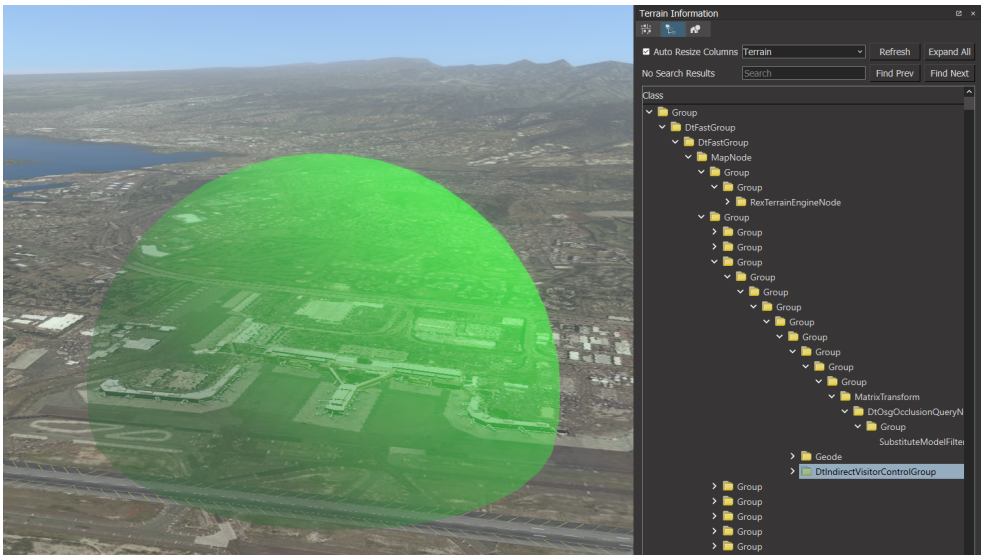
11.12. Drawing Bounding Volumes for Scene Graph Nodes

You can draw bounding volumes for the nodes in the scene graph. This lets a terrain artist or developer visualize the size of the bounding volume for a portion of the terrain so that they know when it will be culled from the scene. It is also useful for finding out if there is something wrong with the size of the bounding volume.

For any given node, the bounding volume drawn is a superset of all the bounding volumes of the children of the node.

Figure 42 shows a bounding volume for the terminal area of the airport in Honolulu, Hawaii, USA in the MAK Earth (online) terrain.

Figure 42. Terrain bounding volume



➤ **To display terrain bounding volumes:**

1. Choose **View > Terrain Information Panel**. The Terrain Information Panel opens.
2. Select the Scene Graph tab.
3. In the drop-down list at the top of the tab, select Terrain.
4. Click Refresh.
5. Expand the scene graph to display the node whose bounding volume you want to display.
6. Right-click the node and, on the context menu, select **Draw Bounding Volume**.
7. Move the observer until you see the bounding volume.

11.13. Masking the Ocean

You can mask the ocean for certain areas to prevent it from rendering. This is useful in areas where the terrain is below sea level. For example, the Dead Sea in the Middle East and Death Valley in California are masked by default. Those areas are blacked out in the file that is used for the ocean mask texture.

The ocean mask texture is set in *./data/Environment/Ocean/Triton.config*. The default is **ocean-mask-texture = ocean-mask-texture.tif**. You can either edit this texture file to black out other areas where you do not want to render the ocean, or you can create a new file to use.

You can use any GeoTIFF texture for masking the ocean. The file must be saved in the same location as the default *ocean-mask-texture.tif* file (*./data/Environment/Ocean*), and it must be defined in the *Triton.config* file.

Higher resolution textures provide better pixel level control for masking the ocean. The limits on the texture size correspond to the limit of the maximum texture size that the GPU can handle. If you do not require a fine level of control over the area you want to mask, you do not need to be extremely accurate. For example, in the default *ocean-mask-texture.tif* file, a large area of southern California is blacked out around Death Valley.

> If the setting is not in the *Triton.config* file, or if the file specified in the setting is not located in the correct directory, the entire ocean is rendered with nothing masked.

11.14. Debugging the VR-Forces Simulation Terrain



VR-Forces has a tool that you can use to help debug problems in the simulation terrain. The front-end and back-end usually use the same terrain, but there may be particular aspects of the terrain configured for simulation purposes as opposed to visualization. You may need to understand why simulation objects are moving on the terrain in unexpected ways. You can examine the following aspects of the terrain:

- The terrain skin.
- Point, areal, and linear features.
- The navigation mesh.
- Terrain queries.

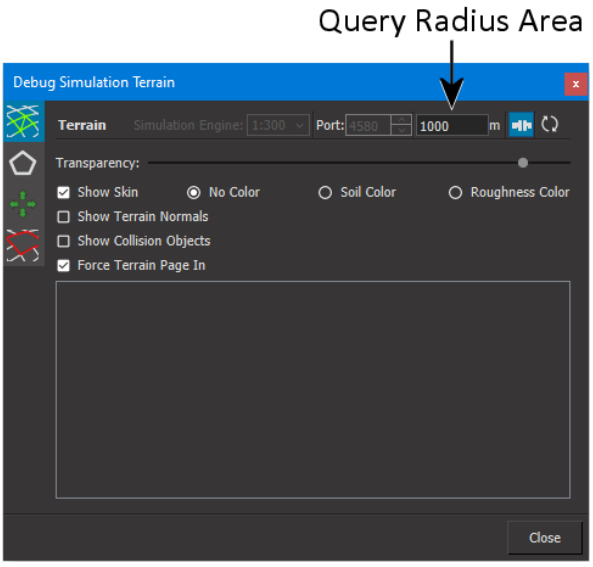
11.14.1 Starting the Simulation Terrain Debugging Tool

This section explains how to start the Debug Simulation Terrain Tool and set it up to debug a terrain.

➤ **To start the Simulation Terrain Debugging Tool:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43).

Figure 43. Simulation Terrain Debugging Tool



2. If you are running more than one back-end, in the Simulation Engine list, select the back-end that you want to debug.
3. Optionally change the Query Area Radius. This determines the area in which VR-Forces applies the debugging tests.
4. Click the Connect button ( ) to connect to the back-end.

11.14.2 Refreshing the Debug Display

The display of debug data does not automatically update as you move the observer. You can refresh the display manually. If you are displaying elevation tiles, you can set the display to refresh automatically. For details, please see "11.14.10 Enabling Auto Refresh" on page 183.

➤ To refresh the display, click the Refresh button ( ).

11.14.3 Viewing Terrain Skin, Normals, and Collision Objects

The Terrain page lets you view the terrain skin, terrain normals, and collision objects. When you view the terrain skin you can also associate soil type or roughness type with a color. The colors associated with soil types and roughness types are configured in *./appData/settings/vrfGui/debugDrawColors.csv*.

You have an option to force page-in of the terrain. Be default, the tool pages in the simulation terrain for all areas of the terrain visible in the window. If you disable this option, VR-Forces pages in terrain for all tiles that have a simulation object on them and wherever you click on the terrain.

➤ **To show the terrain skin:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 above).
2. Select the Terrain page.

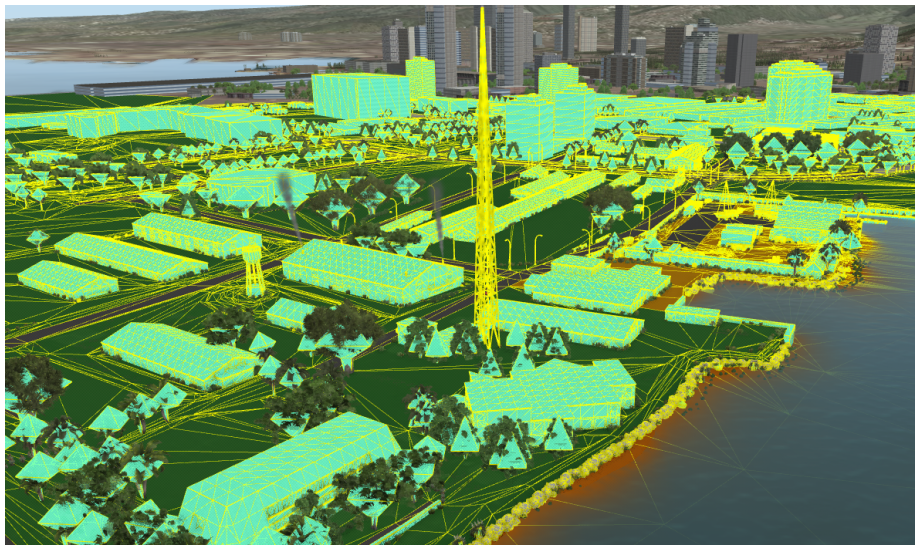
3. Select the Show Skin check box. The terrain skin displays (Figure 44).

Figure 44. Showing the terrain skin



4. Optionally select the Soil Color or Roughness Color option buttons. Figure 45 shows the terrain with soil color. (To get rid of the glow effect, disable high dynamic range (HDR).)

Figure 45. Terrain skin with soil color



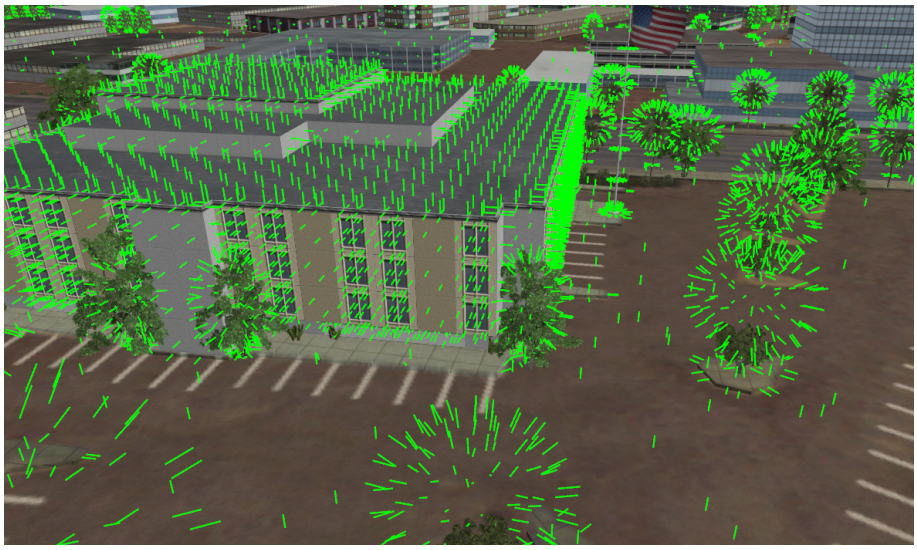
5. Optionally, select or clear the Force Terrain Page In check box.

➤ **To show terrain normals:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on the previous page).
2. Select the Terrain page.

3. Select the Show Terrain Normals check box. The terrain normals display (Figure 46).

Figure 46. Terrain normals



4. Optionally, select or clear the Force Terrain Page In check box.

➤ **To show collision objects:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the Terrain page.
3. Select the Show Collision Objects check box. The terrain normals are displayed (Figure 47).

Figure 47. Collision objects



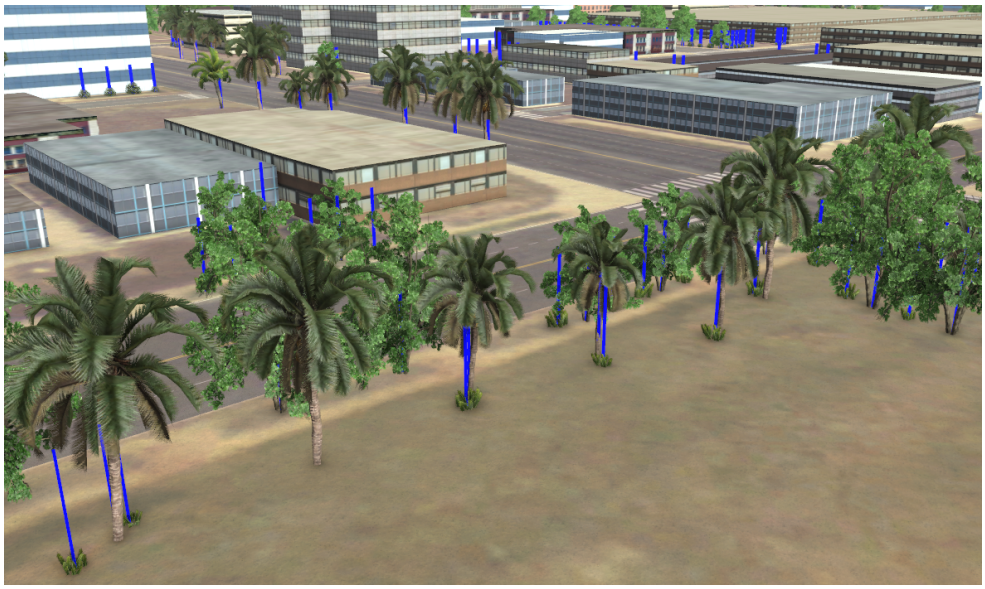
4. Optionally, select or clear the Force Terrain Page In check box.

11.14.4 Viewing Simulation Terrain Features

You can display point, areal, and linear features. Features are displayed based on a set of preconfigured queries that you can select. You can also type in your own queries. For information about how queries are defined, please see "9.3. Named Queries" on page 141.

Figure 48 shows how point features are displayed.

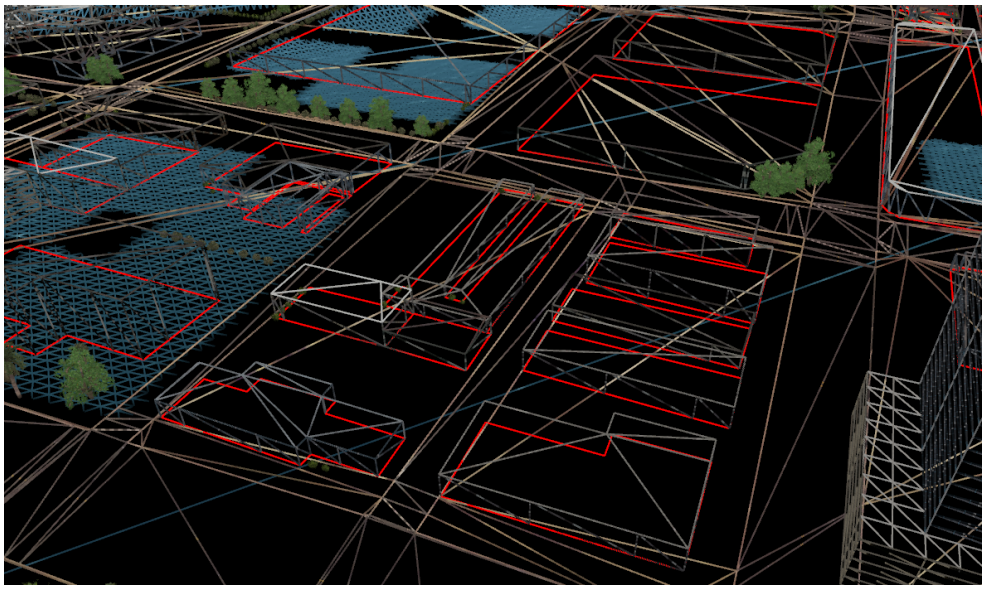
Figure 48. Point features



Areal features display extruded buildings. Changing to wireframe mode helps you see the areas.

Figure 49 shows how linear features display when wireframe is enabled.

Figure 49. Areal features



Each linear feature displays using a different color. There is no particular significance to the color choices.

Figure 50 shows how linear features are displayed.

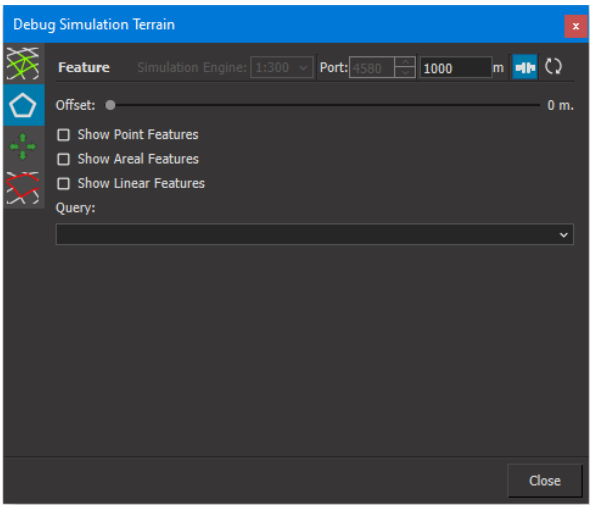
Figure 50. Linear features



➤ **To display simulation terrain features:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the Features page (Figure 51).

Figure 51. Features page



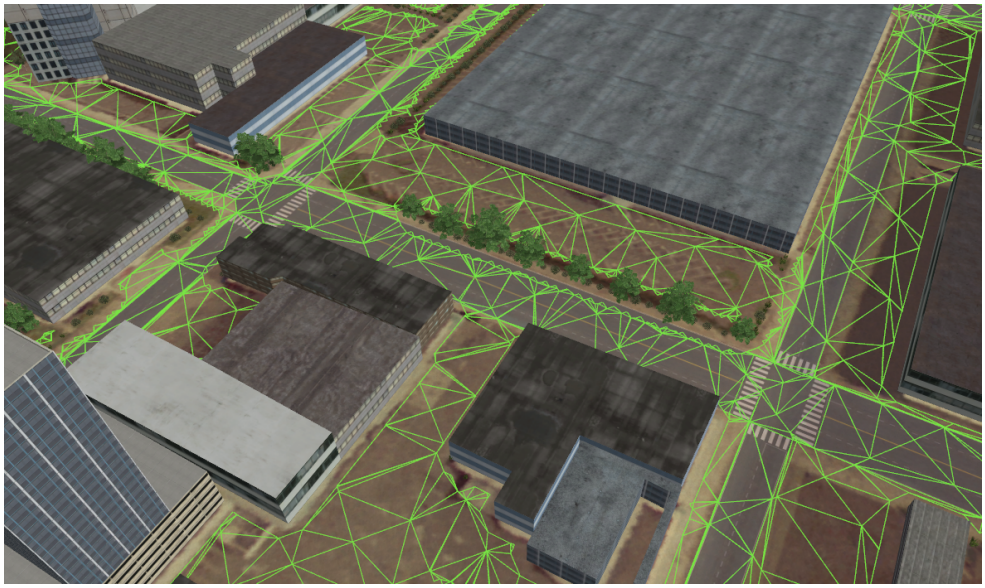
3. Select the check box for each feature type that you want to display.
4. In the list, select a query or type a query.

5. Optionally, adjust the Offset slider. It raises the feature display above the terrain, which can improve visibility.

11.14.5 Displaying the NavMesh

You can display the NavMesh for terrains that have one. You display the NavMesh by selecting one of the profiles. For complete information about the NavMesh, please see the chapter on *Generating Navigation Data* in the *VR-Forces Users Guide*. Figure 52 displays a navigation mesh.

Figure 52. NavMesh

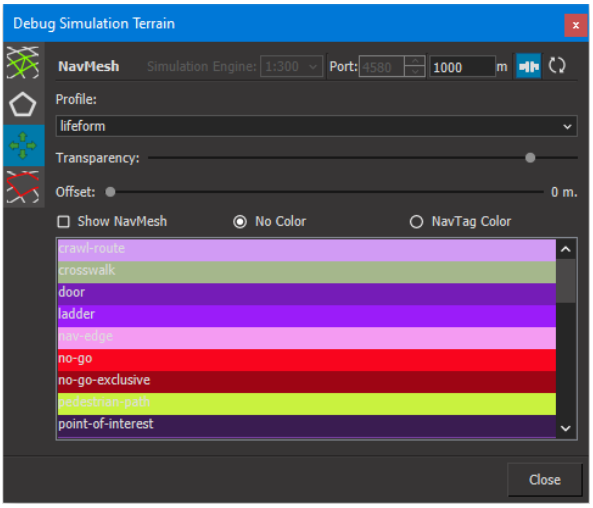


The NavMesh does not always align with the terrain.

➤ **To display the NavMesh:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the NavMesh page (Figure 53).

Figure 53. NavMesh page



3. In the Profile list, select the profile that you want to view.
4. Select the Show NavMesh check box.
5. Select the No Color or NavTag Color option.
6. Optionally, adjust the Offset slider. It raises the navmesh above the terrain, which may improve visibility.

11.14.6 Displaying the Query Area

You can display a box that shows the query area radius (Figure 54).

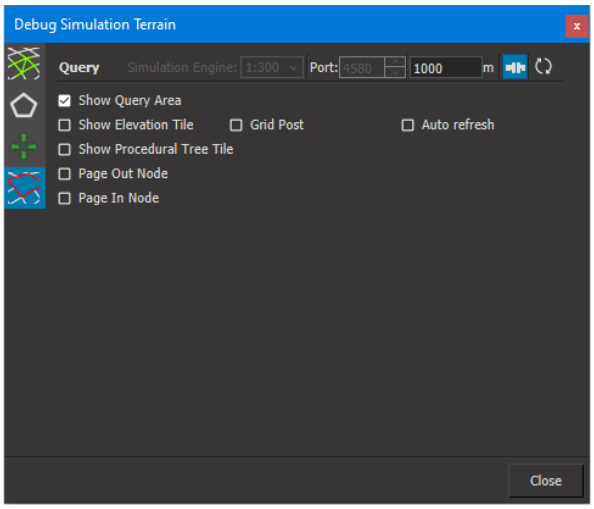
Figure 54. Query area



➤ **To display the query area:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the Query page (Figure 55).

Figure 55. Query page



3. Select the Show Query check box. A red box appears on the terrain.

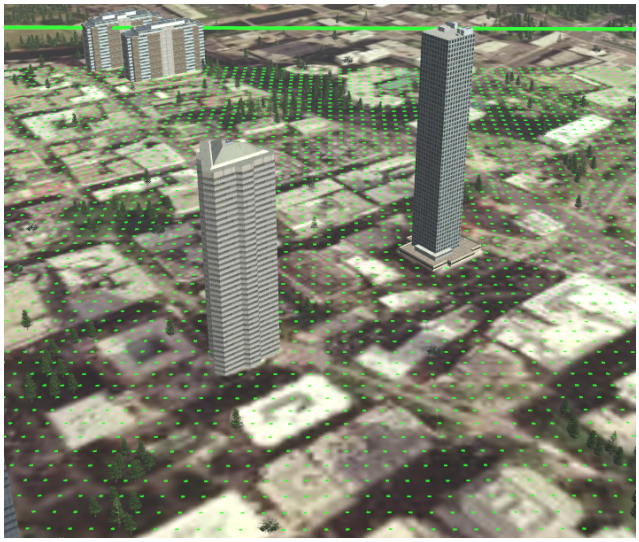
11.14.7 Displaying Elevation Tiles

As entities move through the terrain, VR-Forces pages in elevation tiles for the areas where the entities are located. You can display the areas covered by these tiles (Figure 56). You can also display the grid posts in those areas.

Figure 56. Elevation tiles



Figure 57. Elevation tile and Grid post



VR-Forces only shows elevation tiles that are within or overlap the boundary of the query radius area.

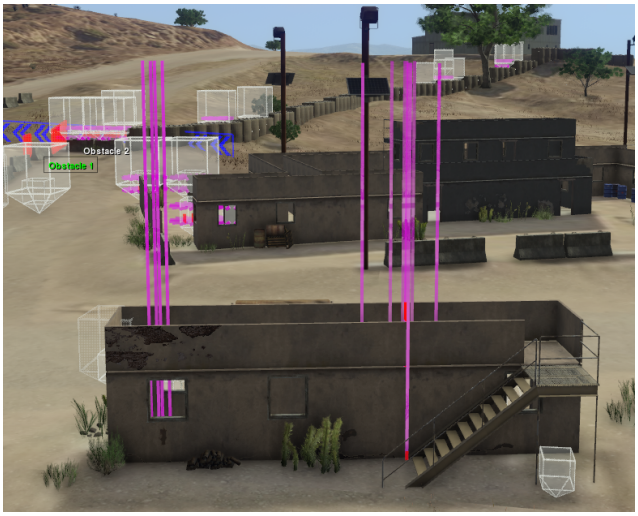
➤ **To display elevation tiles:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the Query page (Figure 55 on the previous page).
3. Select the Show Elevation Tile check box.
4. Optionally, select the Grid Post check box.

11.14.8 Displaying Recent Intersections

You can display the recent intersections, which is useful for model developers to debug and understand what the controller and actuator code is doing. It can also help you find places where intersections are greater than desired. The visualization displays the chords that were tested.

Figure 58. Intersections



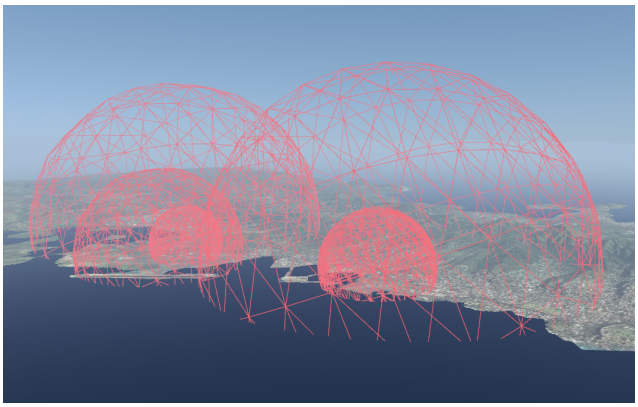
➤ **To display intersections:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the Query page (Figure 55 on page 180).
3. Select the Show Last Intersections check box.

11.14.9 Displaying Page-in and Page-out Nodes

You can display the feature tiles that are loaded (paged in) and you can display the tiles that are in the page-out queue. These are the tiles that will be paged out if higher priority tiles need to be paged in.

Figure 59. Page out nodes



➤ **To display page-in and page-out nodes:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the Query page (Figure 55 on page 180).

3. If you are in Stealth observer mode, zoom out.
4. Select or clear the Page Out Node or Page In Node check boxes.

11.14.10 Enabling Auto Refresh

You can set up the Simulation Terrain Debug Tool to automatically refresh the debug display.

➤ **To set up auto refresh:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 43 on page 173).
2. Select the Query page (Figure 55 on page 180).
3. Select the Auto Refresh check box.

11.14.11 Simulation Terrain Debugging from the Back-End Console

You can display additional terrain paging information by entering commands on the back-end console. You can also display information about feature sets if they are present. Table 20 lists the commands and their results.

Table 20. Terrain paging information commands

Command	Result
`features config`	Prints queries and transforms from *featureconfig.txt*. For details about feature configuration, please see "Chapter 9. Configuring Feature Data in VR-Forces" on page 139.
`features help`	Prints the syntax of feature commands.
`features info`	Prints generic information about the feature sets, such as the number of paged in features and the coordinate system.
`features list`	Lists all feature sets. The index of each set is used on the commands that take lists of feature sets.
`features pageout`	Pages out all unused feature tiles.
`features print` *`index`* [*`number`*]	Prints the feature sets as text. Optionally, specify the maximum number of features to print.
`terrain PSG`	Prints the contents of the scene graph to the console. You can display the same information in the front-end by opening the Scene Graph Nodes Panel.
`pageout`	Pages out all tiles. Tiles that are needed are then paged back in.
`terrain save` *`filename`*	Saves the paged-in tiles to an MTF file. This is useful for quickly and easily saving a portion of a large terrain to a smaller database. The default filename is *simTerrain.mtf*.

Table 20. Terrain paging information commands (continued)

Command	Result
`pbv`	Prints page bounding information to the console.
`terrain addlayer` *`layer`*	Displays information in the layer specified by *`layer`*. Developers can write plug-ins to create new layers and add keywords to display them. The timer layer, included with MAK ONE applications, causes the console to display how long it takes to perform terrain intersections. This information can be useful when diagnosing performance issues.

➤ **To execute a console command:**

1. Open the back-end console.
2. Type the command. You can type commands even if the console is displaying status messages. It remembers what you typed.
3. Press **Enter**.



12. Terrain Tutorials

This section explains how to create a local terrain server. It also shows how to extract props from a terrain.

12.1. Creating an earth File to Load Terrain Data 186
- 12.1.1 Add a Map Element 186
- 12.1.2 Add Elevation Data to the earth File 186
- 12.1.3 Add Imagery to the earth File 186
- 12.1.4 Add Feature Data to the earth File 187
- 12.1.5 Add Options to the earth File 189
- 12.1.6 Save the earth File 189
- 12.1.7 Load the earth File and Save it as an MTF File 190

12.2. Extracting Props from a Terrain 191
- 12.2.1 Create a New Terrain and Add a Terrain Patch 191
- 12.2.2 Extract the Props 192
- 12.2.3 Change the Model Definition for a Prop 194

12.1. Creating an earth File to Load Terrain Data

Earth files are XML files that use osgEarth syntax. All the information that a MAK ONE application needs to load elevation, imagery, and feature data is provided. The data can be stored locally, on a network, or the internet. Data is loaded locally or streamed over the network and paged in as needed.

MAK ONE applications come with a terrain *Little Pond (online).earth* that merges local Little Pond data with streaming terrain from VR-TheWorld Server. For this tutorial, we will ignore the rest of the world and focus on Little Pond.

One approach to creating an earth file is to start with one of the earth files provided, keep the basic structure, and add, edit, or replace elements as needed. For this tutorial, we will start from an empty file and build all the major elements.

12.1.1 Add a Map Element

The root element in an earth file is the **map** element. It encloses all other elements. Open a text editor and add the following code:

```
<map name="Little Pond" type="geocentric" version="2">

</map>
```

12.1.2 Add Elevation Data to the earth File

We want this terrain to load the elevation data in *./SharedData/16/latest/TerrainData/Terrain/LittlePond/Elevation/w072n042.dt1*.

Elevation data is configured using an **elevation** element. The location of data in an earth file is provided in a **url** element. We need to use the gdal driver to load elevation data. Add the following text after the opening of the **map** element:

```
<elevation name="Little Pond Elevation" driver="gdal">
   <url>../../TerrainData/Terrain/LittlePond/Elevation/w072n042.dt1</url>
   <tile_size>32</tile_size>
</elevation>
```

The file path is relative to the directory that contains the earth file.

12.1.3 Add Imagery to the earth File

The Little Pond terrain has two image files: a large low-resolution image and a smaller higher resolution image. Image data is configured using **image** elements. If we were to simply add two **image** elements to the earth file, only the last one listed would be loaded. We would only see one of the images.

To use more than one image, we place an **image** element for each image file inside an **image** element that uses a composite driver. List the images from bottommost to topmost. If you list them in the opposite order from the example code, *LittlePond.tif* would be underneath *W72N42.tif* and you would not be able to see it.

Add the following text after the **elevation** block:

```
<image driver="composite">
    <image name="Little Pond Imagery" driver="gdal">
        <url>../../TerrainData/Terrain/LittlePond/Imagery/W72N42.tif</url>
```

```
    </image>

    <image name="Little Pond Imagery" driver="gdal">

<url>../../TerrainData/Terrain/LittlePond/Imagery/LittlePond.tif</url>
    </image>
</image>
```

We use the gdal driver to load the images.

12.1.4 Add Feature Data to the earth File

In earth files, you add feature data, such as buildings and vegetation, using a **model** element. You can add as many **model** elements as you need. A **model** element specifies the location of a shapefile. Then it adds **style** elements for each feature type to add from the shapefile. In this tutorial, we will add one kind of house and all of the vegetation in the shapefile.

Add a Building to the earth File

Add the following code to your earth file after the **image** block. As in the other elements, the **url** element specifies the location of the shapefile. To load buildings, you need to use the feature_geom driver for the **model** element and the ogr driver for the **features** element.

```
<model name="Little Pond Buildings" driver="feature_geom">
    <features name="Buildings" driver="ogr">
        <url>../../TerrainData/Terrain/LittlePond/Features/BuildingsP.
shp</url>
    </features>
    <layout>
        <tile_size_factor>3</tile_size_factor>
    </layout>
    <styles>
        <style type="text/css">
        HousesColonialGarrison {
        model:              "../../ModelData/Structures/HousesColonial
GarrisonClapboardGreen.medf";
        model-heading:       0-[rotation];
        model-scale:        1;
        model-placement: vertex;
        altitude-clamping:   terrain;
        altitude-resolution: 0.00008583068847656250;
        altitude-offset:     0;
        }
        </style>

        <selector class="HousesColonialGarrison">
            <query>
                <expr>TYPE = 'HousesColonialGarrison'</expr>
            </query>
```

```
            </selector>
        </styles>
        <clustering>false</clustering>
        <lighting>true</lighting>
    </model>
```

The **styles** element lets you specify a **style** for each building you want to display. This example shows just one building. Take a look at *Little Pond (online).earth* to see the other buildings that can be configured from this shapefile.

Each style has a name, in this case HousesColonialGarrison. The **model** attribute specifies the model used to display the object. The **selector** element matches the style to an entry in the TYPE column in the database file that accompanies the shapefile (*BuildingsP.dbf*).

To better understand this relationship, open *BuildingsP.dbf* in a spreadsheet application (Figure 60). The TYPE column lists the different types of buildings in the shapefile. The **selector** element looks in the TYPE column.

Figure 60. *BuildingsP.shp*

	A	B	C	D	E
1	LENGTH	WIDTH	HEIGHT	ROTATION	TYPE
203	8.21000000000	11.02000000000	9.00000000000	345	HousesColonialGarrison
204	8.21000000000	11.02000000000	9.00000000000	35	HousesColonialGarrison
205	8.21000000000	11.02000000000	9.00000000000	320	HousesColonialGarrison
206	8.21000000000	11.02000000000	9.00000000000	230	HousesColonialGarrison
207	8.21000000000	11.02000000000	9.00000000000	230	HousesColonialGarrison
208	8.21000000000	11.02000000000	9.00000000000	250	HousesColonialGarrison
209	8.21000000000	11.02000000000	9.00000000000	155	HousesColonialGarrison
210	8.21000000000	11.02000000000	9.00000000000	325	HousesColonialGarrison
211	8.21000000000	11.02000000000	9.00000000000	250	HousesColonialGarrison
212	8.21000000000	11.02000000000	9.00000000000	140	HousesColonialGarrison
213	8.21000000000	11.02000000000	9.00000000000	335	HousesColonialGarrison
214	8.21000000000	11.02000000000	9.00000000000	65	HousesColonialGarrison
215	8.21000000000	11.02000000000	9.00000000000	210	HousesColonialGarrison
216	8.21000000000	11.02000000000	9.00000000000	125	HousesColonialGarrison
217	8.21000000000	11.02000000000	9.00000000000	335	HousesColonialGarrison
218	8.21000000000	11.02000000000	9.00000000000	30	HousesColonialGarrison
219	8.21000000000	11.02000000000	9.00000000000	335	HousesColonialGarrison

Add Vegetation to the earth File

The vegetation uses SpeedTree models. To load SpeedTrees from an earth file, you need to use the feature_custom driver in the **model** element. You use the ogr driver in the **features** element.

Similar to the buildings, the **model** block for vegetation specifies a shapefile and styles. However, unlike with the buildings, we do not specify individual models to load. The marker attribute indicates that the application should take every entry in the VEG column of *VegetationP.shp*, append ST to it, and look for a model of that name. For example, the column has several entries named TreesBigLeafMaple. The application adds ST to form TreesBigLeafMapleST, finds the model definition TreesBigLeafMapleST, and loads the model specified in the model definition.

To add vegetation to the terrain, add the following code to the earth file after the **model** block for the buildings:

```
<model name="Little Pond Vegetation" driver="feature_custom">
    <features name="Vegetation" driver="ogr">
        <url>../../TerrainData/Terrain/LittlePond/Features/VegetationP
.shp</url>
    </features>
    <layout>
        <tile_size_factor>3</tile_size_factor>
    </layout>
    <styles>
        <style type="text/css">
            Vegetation {
            marker:              [veg] + "ST" ;
            marker-scale:        1;
            marker-placement: vertex;
            altitude-clamping:   terrain;
            altitude-resolution: 0.00008583068847656250;
            altitude-offset:     0;
            }
        </style>
    </styles>
    <clustering>true</clustering>
    <lighting>true</lighting>
</model>
```

12.1.5 Add Options to the earth File

Finally, add some common options to the file. In particular, this section specifies a location for the cache. Add this block just before the closing of the **map** element:

```
<options>
   <lighting>true</lighting>
   <elevation_interpolation>triangulate</elevation_interpolation>
   <terrain>
       <min_tile_range_factor>3</min_tile_range_factor>
       <skirt_ratio>0.1</skirt_ratio>
   </terrain>
   <cache type="filesystem">
       <path>../../appData/cache</path>
   </cache>
</options>
```

12.1.6 Save the earth File

You should save earth files in the *./SharedData/16/latest/TerrainData/TerrainConfiguration* directory. Save the file as *myLittlePond.earth*.

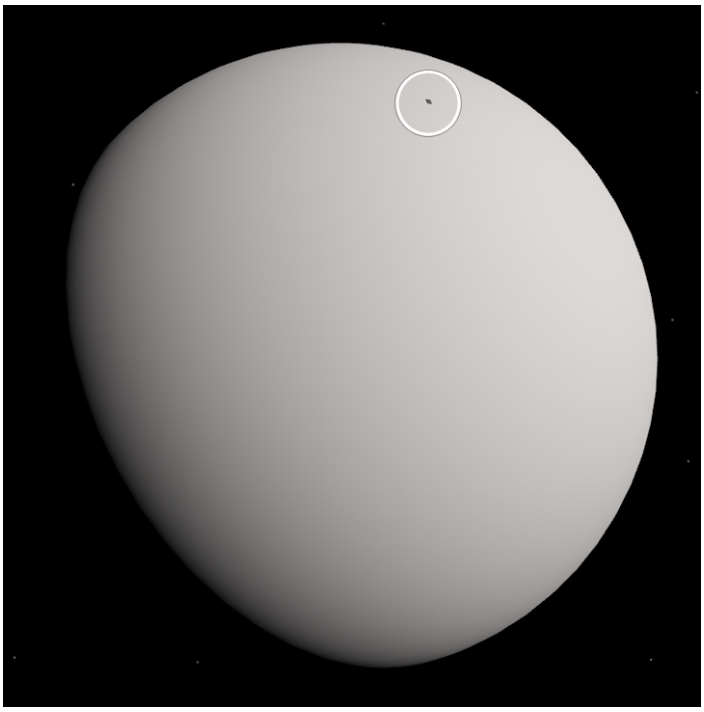
12.1.7 Load the earth File and Save it as an MTF File

Your earth file for Little Pond is complete. After you load it, you should save it as an MTF file.

➤ **To load the earth file:**

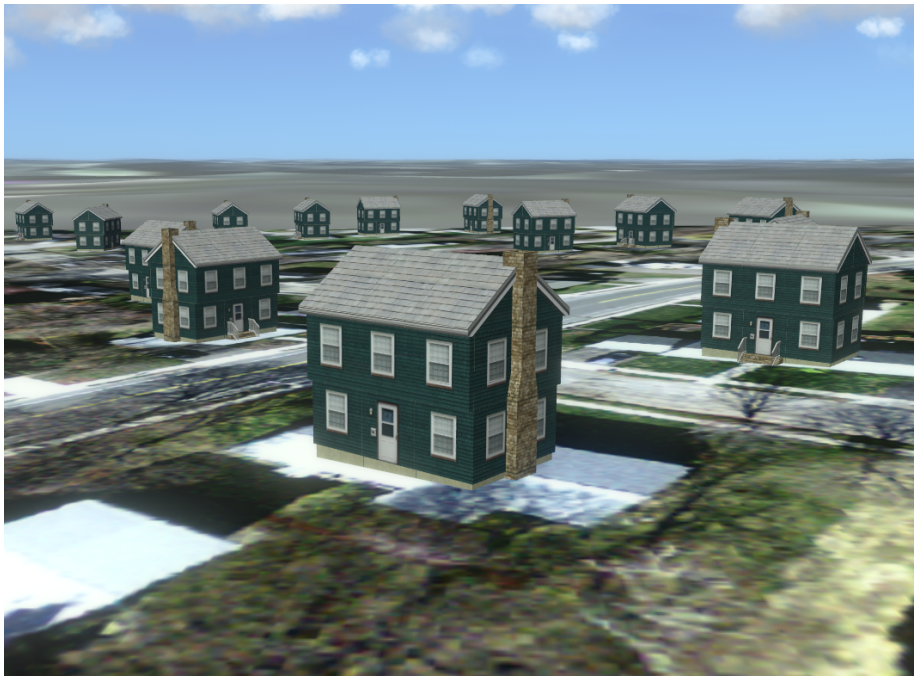
1. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.
2. Select the Add Data tab.
3. Click Terrain Patch. The Add Terrain Patch dialog box (a file browser) opens.
4. Select *./SharedData/16/latest/TerrainData/TerrainConfiguration/myLittlePond.earth*.
5. Click Connect. The terrain loads. It is visible as a small patch on the Earth's sphere (Figure 61).

Figure 61. Little Pond terrain



6. Zoom in on the terrain (Figure 62).

Figure 62. Little Pond terrain



7. Choose **File > Save Terrain**. The Save Terrain dialog box opens.
8. Give the terrain a name.

12.2. Extracting Props from a Terrain

In this tutorial, we show how to create a prop by extracting information from a terrain.

The general steps for extracting props from a terrain are:

1. Create a new terrain with a terrain patch ("12.2.1 Create a New Terrain and Add a Terrain Patch" below) or open a terrain that has external references.
2. Select the external references that you want to extract as props and extract them ("12.2.2 Extract the Props" on the next page).
3. Optionally, change the model definitions ("12.2.3 Change the Model Definition for a Prop" on page 194).

12.2.1 Create a New Terrain and Add a Terrain Patch

➤ **To create the new terrain and add a terrain database:**

1. Choose **File > New Terrain**. If any terrain data is loaded, it is removed.
2. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.
3. Select the Add Data tab.
4. Click Terrain Patch. The Add Terrain Patch file browser opens.

5. Navigate to the *./SharedData/16/latest/TerrainData/Terrain/VR-Village/Geometry/OpenFlight* directory.
6. Select *VR-Village.medf*.
7. Click Open. The Add Terrain Patch dialog box opens.
8. Clear the Use Indirect Rendering check box.
9. Click OK. The terrain loads.

12.2.2 Extract the Props

➤ **To extract the props:**

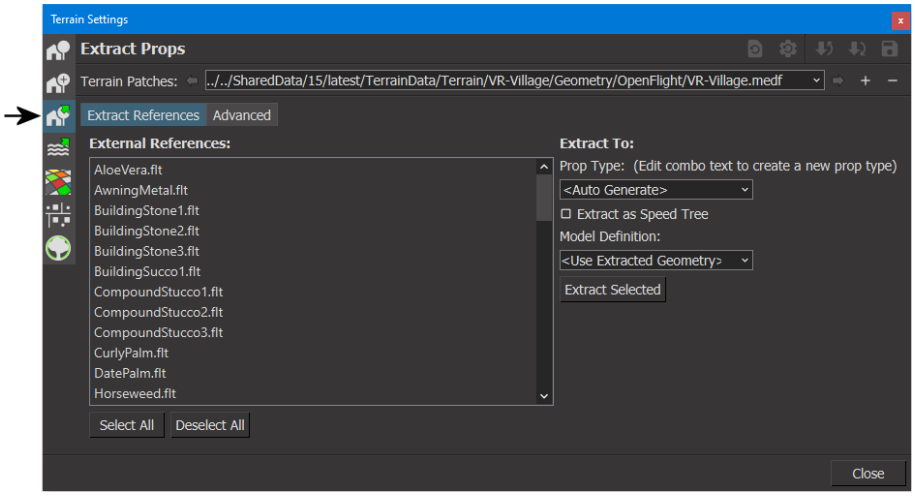
1. Move the observer so that you have a view of the village similar to Figure 63.

Figure 63. VR-Village



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 64). 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 64. Extract props page



5. Click Select All. All of the references are selected.
6. Under Extract To, 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 65). 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 65. Selected props



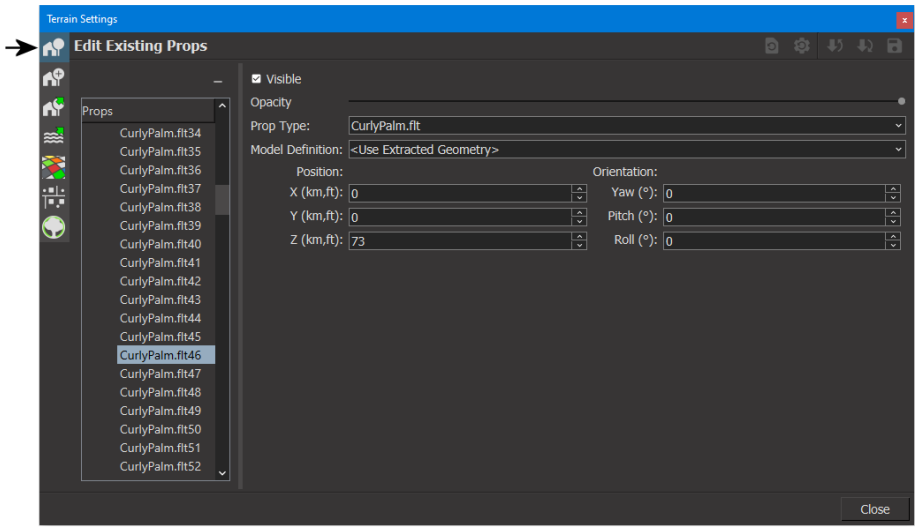
You may see a message at the top of the Extract Props page that reads "Extraction is disabled due to terrain patch object being rendered with indirect. Please disable indirect for the terrain patch to enable prop extraction." If you do, follow the instructions in "11.5.3 Disabling Indirect Rendering for Terrain" on page 165.

12.2.3 Change the Model Definition for a Prop

➤ **To change the model definition for a prop:**

1. Select one of the trees.
2. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
3. Select the Edit Existing Props page. The selected prop is highlighted in the list of props (Figure 66).

Figure 66. Edit Existing Props page



4. In the Model Definition box, select a model definition from the list, such as TreesGiantRedwoodST. The tree changes to the new model definition (Figure 67).

Figure 67. New model definition for tree



5. To restore the tree to its original model, on the Edit Existing Props page, change the model definition to Use Extracted Geometry.



III. Visual Models in MAK ONE Applications

The people, vehicles, and cultural features that you see in the display window of a MAK ONE application are rendered based on visual models specific to an object type. The movement and behavior of objects that can move are controlled by simulation models in the VR-Forces back-end or in a remote application. This section explains how MAK ONE applications manage visual models.

Chapter 13. Introduction to Visual Models

Chapter 14. Schemas

Chapter 15. Model Definitions

Chapter 16. Element Definitions

Chapter 17. Mapping Models and Effects

Chapter 18. Wakes, Cockpits, 2D Icons, and Other Models

Chapter 19. The WRM Specification (DIS Notes)

Chapter 20. The MAK FBX Specification

Chapter 21. Special Effects (Particle Systems)



13. Introduction to Visual Models

This chapter introduces basic concepts for how MAK ONE applications manage visual models.

13.1. Introduction to Object Modeling 200
13.1.1 Model Sets 202
13.2. Model Formats for 3D Objects, and Effects 202
13.3. OpenFlight and FBX Models 203
13.4. Managing Model and Element Definitions 203
13.5. Visual Model Composability 204
13.6. Indirect Rendering for Entities 208

13.1. Introduction to Object Modeling

The objects that MAK ONE applications can display that are not part of the terrain, such as simulation objects, tactical graphics, detonations, and cockpit displays, are generically called scene elements. Scene elements are defined in element definitions. When a MAK ONE application joins an exercise, it receives information about objects over the network. When the application runs a scenario, it publishes information about the objects it is simulating. It may also receive object information over the network from other simulation participants. When VR-Vantage joins an exercise, it receives information about objects over the network. When VR-Forces runs a scenario, it publishes information about the objects it is simulating. It may also receive object information over the network from other simulation participants. Similarly, VR-Engage receives object information from the scenarios that it hosts or connects to. The objects are identified using object type enumerations. MAK ONE applications map these object types to element definitions. (For more information about object type enumerations and mapping them to element definitions, please see "17.1. Introduction to Object Type Mapping" on page 248.)

An element definition is a set of visual definitions that tell MAK ONE applications how to render (display) the object. Visual definitions specify the 3D model, 2D icon, or graphic, and the visual effects that the object supports, such as trailing effects, height-above-terrain lines, track histories, and so on. A scene element can have up to three different element definitions – one for each model set (3D, 3D Colorized, 2D Icon). MAK ONE applications choose the correct element definition to use based on the model set being used by the observer. (For more information about model sets, please see "13.1.1 Model Sets" on page 202.)

Element definitions are created and edited in the Visual Model Editors dialog box. In addition to letting you manage element definitions, this dialog box lets you create and manage model definitions and schemas. Model definitions specify values for some basic object parameters. Most importantly for most MAK ONE applications users, model definitions specify the 3D model to use for element definitions in the 3D Models model set. What this means is that if you want to add a new model to MAK ONE applications, you will specify the model file, such as an OpenFlight file, in a model definition and then specify that model definition in the element definition for the simulation object you want to add.

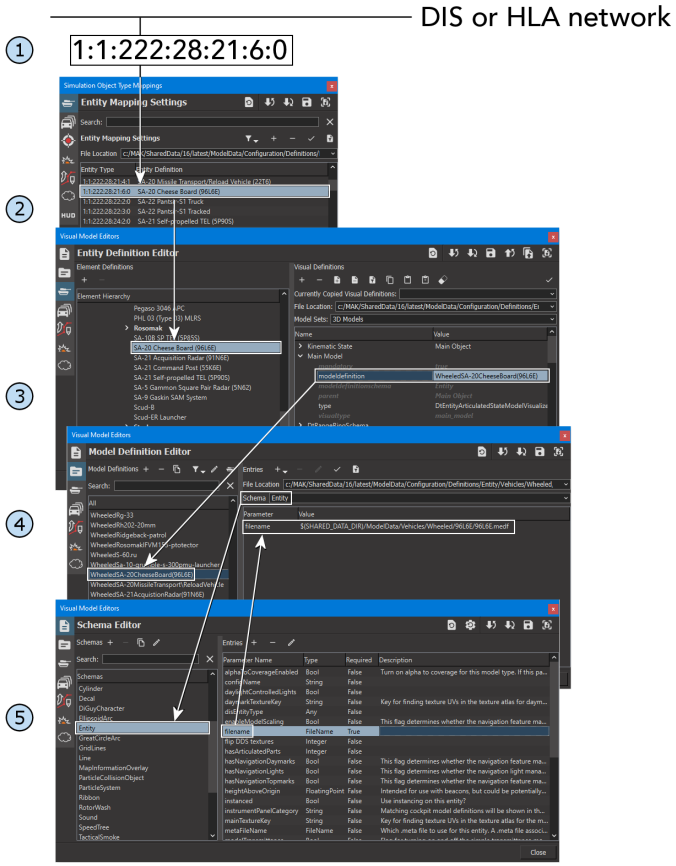


The chapters in this manual that discuss how to configure visual models do so in the context of using the Visual Models Editor dialog box. We strongly recommend that whenever possible VR-Forces users use the Simulation Object Editor to add and configure visual models. The only time that you should have to use the Visual Models Editor dialog box is if you need to add a new schema or edit model attributes that are not supported by the Simulation Object Editor. For details about how to use the Simulation Object Editor, please see Section XII, *Creating and Editing Simulation Models*, in the *VR-Forces Users Guide* and the chapters in that section.

Schemas define the most basic scene elements, such as an entity, SpeedTree tree, or line. Most MAK ONE application users will never have to create or edit a schema. However, if you create a new model definition, you must make sure it uses the correct schema and that it includes all required parameters.

Figure 68 illustrates these relationships using an M1A2 entity. A MAK ONE application receives the object type 1:1:225:1:1:3:-1 over the network (callout 1). In the Simulation Object Type Mappings dialog box, this object type is mapped to the M1A2 Abrams entity definition (2). (An entity definition is a specific type of element definition.) The M1A2 Abrams entity definition (in the Visual Model Editors dialog box) specifies all of the visual definitions for this entity (3). The Main Model visual definition has a **modeldefinition** parameter. The value for this parameter is TrackedM1A2Abrams. The TrackedM1A2Abrams model definition has a **filename** parameter that specifies *M1A2.medf* as the model to use for entities that use this model definition (4). The TrackedM1A2Abrams model definition has a **filename** parameter because it uses the Entity schema and the Entity schema requires a **filename** (5).

Figure 68. Element definitions and simulation object type mapping



When you add simulation objects and other scene elements in the Simulation Object Editor, it creates model definitions and handles all of the mappings described in the previous paragraphs and the illustration. Although this manual explains all of the details of the Visual Model Editors dialog box and Simulation Object Type Mappings dialog box, using the Simulation Object Editor to add simulation objects is the preferred method.

13.1.1 Model Sets

The model or icon that is used for an object is determined by its element definition. For convenience, element definitions are organized into model sets. For example, all of the element definitions that specify 3D models are in the 3D Models model set, and so on. As a result, when you choose the 3D Models model set for an observer mode, all of the entities use a 3D model. (The model set is an observer-specific setting. VR-Vantage allows you to change model sets.)

Each MAK ONE application includes one or more of the following types of object models and icons:

- 3D Models. Realistic 3D models. This is the default model set for the Stealth observer mode.
- 3D Colorized Models. 3D colorized models are cartoon-like models that create an exaggerated notional view of the world. This is the default model set for XR observer mode.
- 2D Icons. 2D MIL-STD 2525B font-based icons. This is the default model set for the Plan View observer mode. You can also configure MAK ONE applications to use 2525a images or any arbitrary images for 2D icons.

The model sets available in MAK ONE applications are fixed. You cannot add model sets.

For information about changing model sets, see :

- "Changing the Model Set" in the *VR-Vantage Users Guide*
- "Changing the Model Set" in the *VR-Forces Users Guide*

13.2. Model Formats for 3D Objects, and Effects

MAK ONE applications include 3D models for vehicles, lifeforms, environmental objects, and effects. The models are in a compressed, encrypted format (extension *.medf*).

MAK ONE applications include a utility, the MEDF Tool, that lets you encrypt files in MEDF format. You cannot use it to decrypt encrypted files. For more information, please see "18.16. Compressing Model Files" on page 289.

You can provide your own models if they are in OpenFlight or FBX format.

13.3. OpenFlight and FBX Models

The vehicle models provided with MAK ONE applications were originally created in either OpenFlight (FLT) or FBX formats. OpenFlight is a runtime format that is used primarily within the modeling and simulation industry. FBX is an interchange format used throughout the gaming industry and is supported by the model development tools, such as 3DS MAX and Maya, commonly used by that industry.

The distinction between runtime format and interchange format is important. Vehicle models typically consist of:

- Geometry - the physical dimensions of the object.
- Textures - the skin of the object.
- Articulated parts - moving parts such as turrets, wing flaps, or cargo doors. (Some models do not have articulated parts.)
- Animations - used to show movement of wheels, propellers, rotors, and so on. Animations provide realism without the overhead of actually physically modeling moving parts. OpenFlight models support flipbook animations. FBX models supports flipbook and key frame animation.

FBX models provide only the 3D model. They do not specify any moving parts. OpenFlight models can include comments that specify the behavior of articulated parts. They also include comments that control levels of detail (LODs). These comments are specified in the WRM Specification. (For details about the WRM Specification, please see "Chapter 19. The WRM Specification (DIS Notes)" on page 293.)

When a MAK ONE application loads an OpenFlight model (whether in native format or encrypted format (MEDF)), it reads the comments and automatically knows how to visualize the articulated parts because they conform to the WRM Specification. When a MAK ONE application loads an FBX model, it does not necessarily know how to represent anything other than the geometry and textures. That is because the FBX format does not have a standard way to represent moving parts. However, since the nodes in an FBX model have names, if LOD nodes and articulated part nodes are named intelligently and consistently, it is possible to map them to DIS articulated part names and MAK ONE applications can then visualize them properly. MAK has implemented a way to do this using configuration files. For details, please see "Chapter 20. The MAK FBX Specification" on page 329.

13.4. Managing Model and Element Definitions

When a MAK visualization application discovers a simulation object (entity or unit) on the network, it must decide how to represent it in the GUI. This is accomplished by mapping 3D models and 2D icons to object type enumerations.

The chapters in this section explain how to use the Visual Model Editors dialog box to create and edit schemas, model definitions, and element definitions. This process can be rather complex. If you are customizing the visual characteristics of the simulation objects provided with MAK ONE applications or adding new object types that are significantly different from those provided with MAK ONE applications, you may need to create new model definitions or edit the existing ones.

If you are using VR-Forces and if you just need to add new object types that are similar to existing ones, it is strongly recommended that you do so in the Simulation Object Editor.

When you create a simulation object in the Simulation Object Editor, it adds new model definitions and object type mappings automatically. You will not need to do anything in the Visual Model Editors to configure the new simulation object.

Model definitions and element definitions (visual definition files) are SMS-specific. The visual definition files for the SMSs provided with MAK ONE applications are stored in subdirectories of *./SharedData/16/latest/ModelData/Configuration/Definitions*. If you create a new SMS, the visual definition files are stored as part of the SMS, in *./data/simulationModelSets/<sms>/gui/visuals*. You can copy the visual definition files for the provided SMSs into the SMS hierarchy if you want to make it easier to copy an SMS to another VR-Forces installation.



VR-Vantage does not use simulation model sets. However, due to the integration between VR-Vantage and VR-Forces, VR-Vantage uses the same set of visual definition files as VR-Forces.

When you open the Visual Model Editors, it loads the visual model definitions for the SMS used by the loaded scenario. If no scenario is loaded, it defaults to *EntityLevel.sms*. For details about how to edit visual models in the Simulation Object Editor, see "Editing a Simulation Object's Visual Model" in the *VR-Forces Users Guide*.

13.5. Visual Model Composability

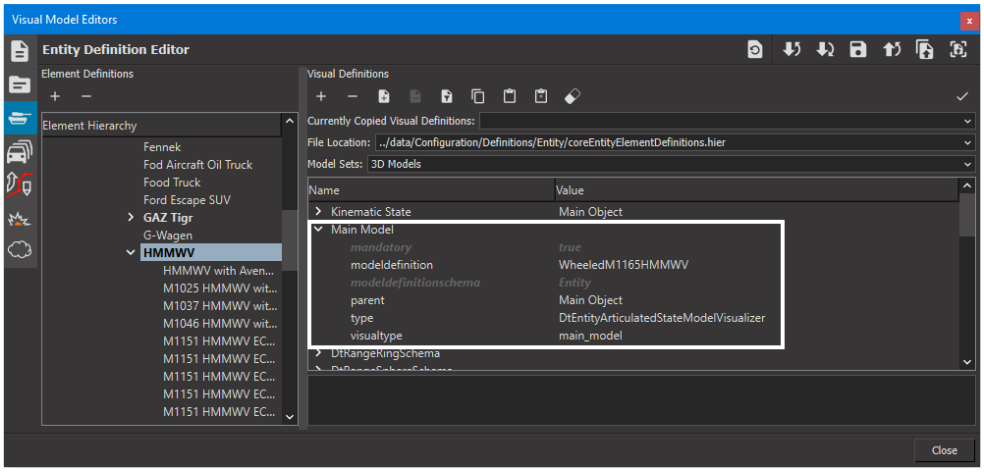
MAK ONE applications compose models for rendering performance, configuration optimization, and asset management efficiency.

- Each model that has its own element definition is an instanced model, which supports optimal rendering performance.
- Using the element definition hierarchy, models inherit all parameters from the parent definitions. Only the element-specific attributes need to be changed.
- Similar models share a base model. This saves on the number of models that need to be built and reduces the effort when modifications need to be made.

Visual models are composed in the Visual Model Editors dialog box. Each base vehicle model, weapon, or vehicle variant has its own model definition that specifies its visual model file.

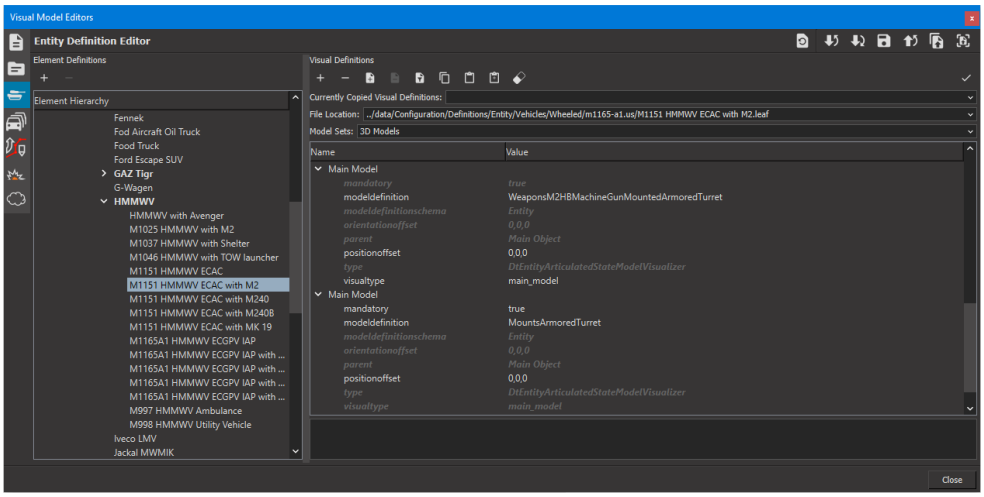
In the Visual Model Editors dialog box, each element definition has a Main Model visual definition, which specifies the model definition that it uses, and hence, the visual model that will be displayed (Figure 69).

Figure 69. Model definition for Main Model



The element definition can have multiple additional Main Model visual definitions, each with its own model definition and offset, which can be composed on top of the base Main Model (Figure 70).

Figure 70. Multiple Main Model definitions



In the HMMWV element definition, we have multiple entities that use the same base model, but different weapon configurations (Figure 71 and Figure 72).

Figure 71. Multiple weapon options

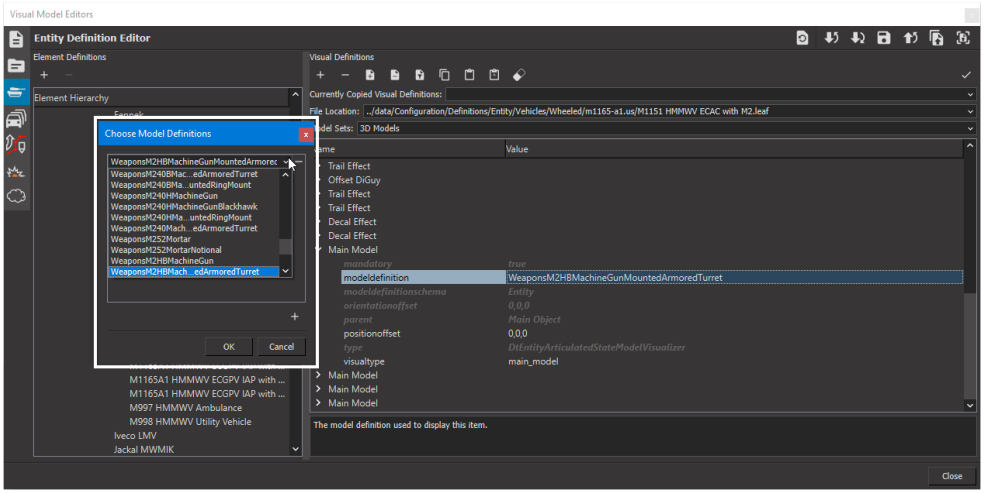
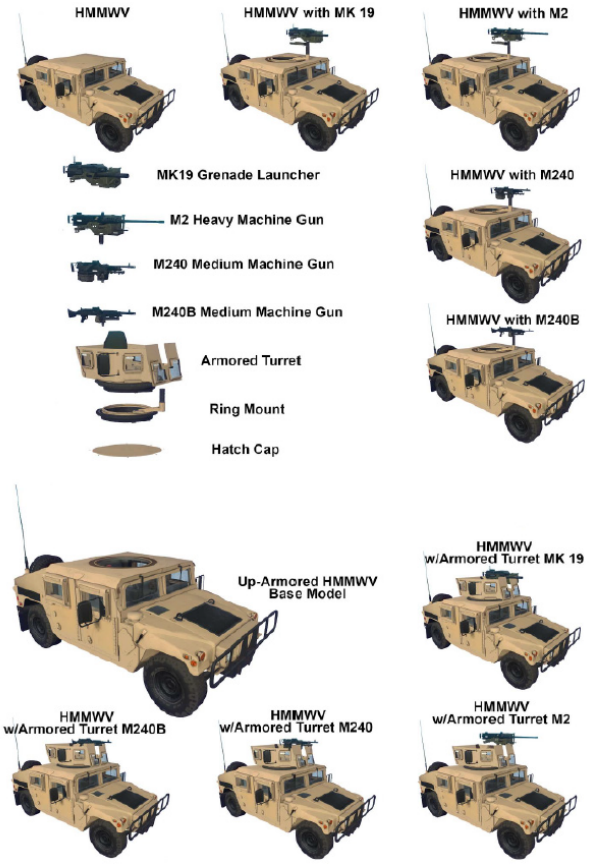


Figure 72. HMWWV models



The weapon configuration is also used on other entities. Figure 73 shows an older M1025 HMMWV using the same M2 with Ring Mount configuration designed for the Up-Armored HMMWV, with just an offset applied in configuration.

Figure 73. Weapon with offset



The M1 Variations are composed in the same fashion to create the M1A2 SEP and SEP V2 (Figure 74).

Figure 74. M1 variations



This modular approach also works on other configurations besides weapons. Vehicles that use the same chassis, like the HEMTT, can be configured to use the same base model (Figure 75 and Figure 76).

Figure 75. HEMTT element definition

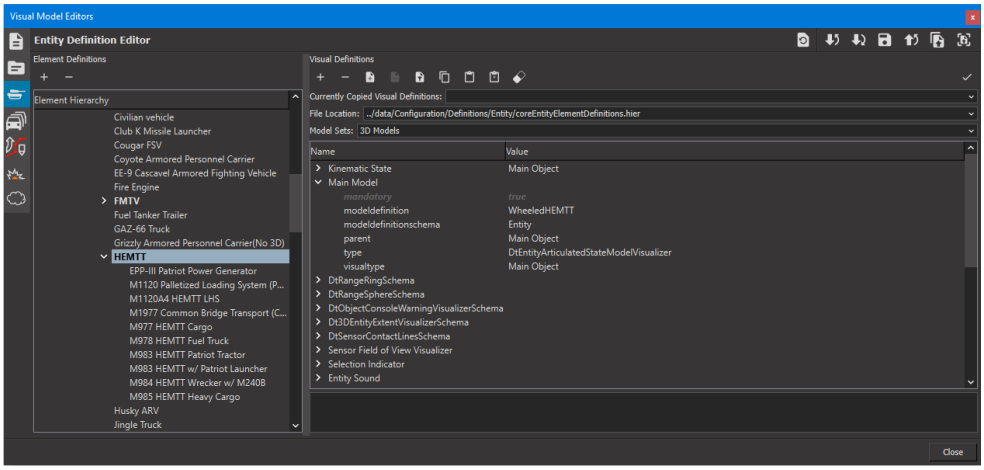
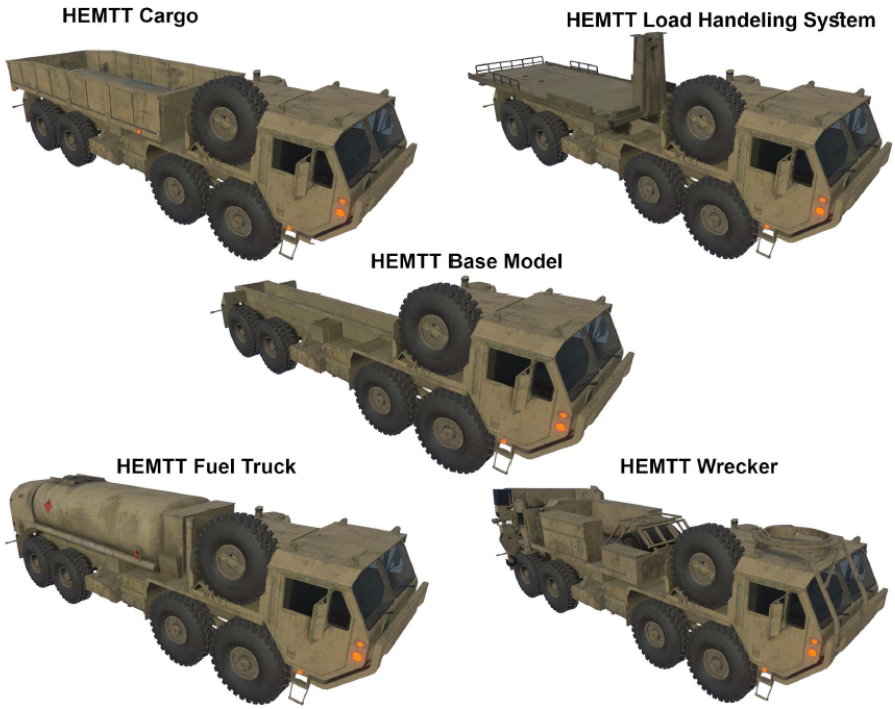


Figure 76. HEMTT models



13.6. Indirect Rendering for Entities

MAK ONE applications use indirect rendering for entities (except for DI-Guy characters). Indirect rendering is enabled by default. It improves rendering performance by combining geometry into fewer draw calls. For more information, please see "11.5. Indirect Rendering of Terrain" on page 164. The sections on scaling LOD distance and transition percentage also apply to entities.

If you do not want to use indirect rendering, you can disable it with the `--disableIndirectEntities` command-line argument.





14. Schemas

In "Section III. Visual Models in MAK ONE Applications" on page 197 we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit model definitions and model schemas.

14.1. Creating and Editing Schemas 210
14.2. Creating Schemas 211
14.3. Saving Schemas 212
    14.3.1 Exporting Selected Schemas 212
14.4. Deleting Schemas 213
14.5. Copying Schemas 213
14.6. Adding a Parameter to a Schema 214
14.7. Editing a Schema Parameter 215
14.8. Deleting a Parameter from a Schema 215

14.1. 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. MAK ONE applications use 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, see "Managing VR-Vantage Settings" in the *VR-Vantage Users Guide* or "Managing VR-Forces Settings" in the *VR-Forces Users Guide*.

Model definition schemas define the elements that can be displayed in a scene. The model definition schemas provided with MAK ONE applications are:

- 3DText
- Animation
- Channel Compass
- Cockpit Display
- ControlLine
- ControlPoint
- ControlPrism
- Cuboid
- Cylinder
- Decal
- DiGuyCharacter
- EllipsoidArc
- Entity
- GreatCircleArc
- GridLines
- Line
- ParticleCollisionObject
- ParticleSystem
- Ribbon
- RotorWash
- Sound
- SpeedTree
- TacticalSmoke
- Terrain

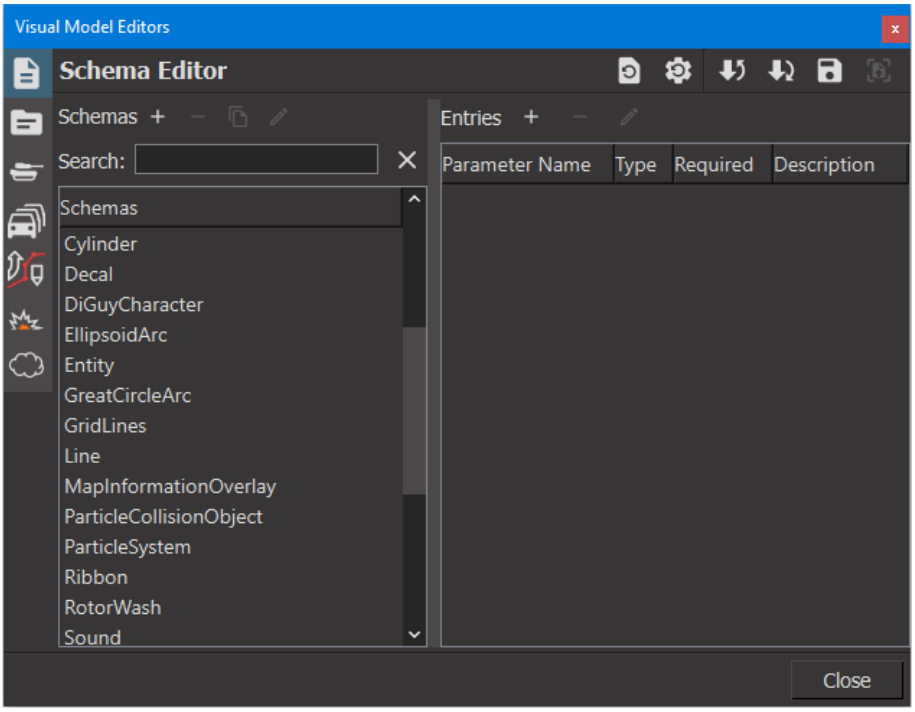
- TidalStreamWake
- Tree
- Wake
- Watercraft
- WaterImpact
- WidgetTheme

14.2. Creating Schemas

➤ **To create a schema:**

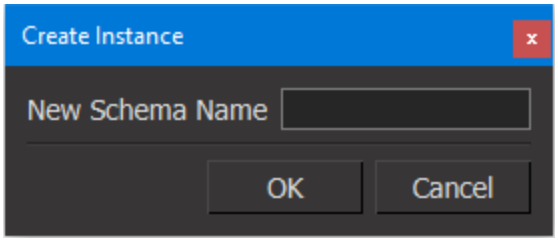
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 77).

Figure 77. Schema Editor page



3. Click the Add button ( ) next to the Schemas label. The Create Instance dialog box opens (Figure 78).

Figure 78. Create Instance dialog box



4. Type a name for the new schema.
5. Click OK.
6. Add parameters as described in "14.6. Adding a Parameter to a Schema" on page 214.

14.3. Saving Schemas

You can save all schemas and you can save selected schemas to their own file.

➤ **To save all schemas**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 77 on the previous page).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. Click the Save button ( ). A confirmation prompt opens.
5. Click Yes. The Export Schema dialog box opens.
6. Select the file to save to or type the name of a new schema file.
7. Click Save.

14.3.1 Exporting Selected Schemas

➤ **To export selected schemas:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 77 on the previous page).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. Select the schema that you want to save.
5. Click the Export Selected button ( ). The Export Schema dialog box opens.
6. Select the file to save to or type the name of a new schema file.
7. Click Save.

14.4. Deleting Schemas

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

➤ **To delete a schema:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 77 on page 211).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema you want to delete.
5. Click the Delete button ( ) next to the Schemas label. You are prompted to confirm the deletion.
6. Click Yes. The schema is removed from the list.
7. Save the schemas.

14.5. 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:**

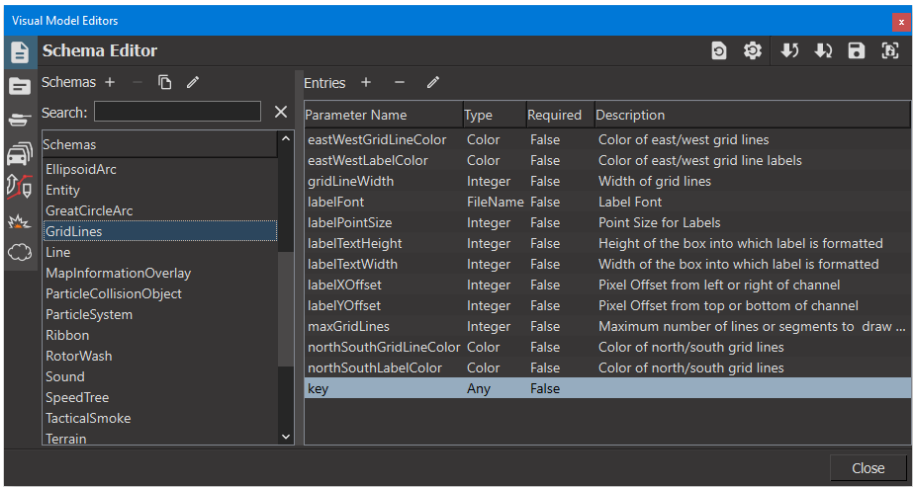
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 77 on page 211).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. Select the schema that you want to copy.
5. Click the Duplicate button ( ). The Duplicate Schema dialog box opens.
6. Type a name for the copied version of the schema.
7. Click OK.
8. The new schema is added to the list.
9. Edit the parameters as described in "14.7. Editing a Schema Parameter" on page 215.
10. Save the schemas.

14.6. Adding a Parameter to a Schema

➤ **To add a parameter to a schema:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page.
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema to which you want to add a parameter.
5. Click the Add button (+) next to the Entries label. A parameter called “key” is added to the parameter list (Figure 79).

Figure 79. New schema parameter



6. Double-click “key”. The Rename Parameter dialog box opens.
7. Type a name for the parameter.
8. If the parameter has a specific type, click the value in the Type column and edit it.
9. If the parameter is required, click the value in the Required column and change it to True.
10. Optionally, add a description. The description provides context sensitive help on the element definition pages. To add a description:
    a. Click the empty field in the Description column. The Edit Text dialog box opens.
    b. Type the descriptive text.
    c. Click OK.

14.7. Editing a Schema Parameter

➤ **To edit a parameter for a schema:**

1. Choose **Settings > Visual Model Editors.** The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 77 on page 211).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema for which you want to edit a parameter.
5. Click the parameter attribute that you want to edit. The attribute becomes editable.
6. Type the new attribute value.

14.8. 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 77 on page 211).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. In the Schemas list, select the schema from which you want to delete a parameter.
5. Select the parameter that you want to delete.
6. Click the Delete button ( ) next to the Entries label.



15. Model Definitions

In "Section III. Visual Models in MAK ONE Applications" on page 197 we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit model definitions.

15.1. Creating and Editing Model Definitions 218
15.2. Work Flow Issues for Model Definitions 219
15.3. Model Definition Files 220
15.4. Creating a Model Definition 221
15.5. Deleting a Model Definition 222
15.6. Copying a Model Definition 222
15.7. Filtering the List of Model Definitions 223
15.8. Editing a Model Definition 223
    15.8.1 Adding Parameters to a Model Definition 224
    15.8.2 Editing a Model Definition Parameter 225
    15.8.3 Deleting Parameters from a Model Definition 225
15.9. Saving Model Definitions 226
    15.9.1 Exporting Selected Model Definitions 226
15.10. Importing Model Definitions 226
    15.10.1 Importing Model Definitions from the Command Line 227

15.1. Creating and Editing Model Definitions

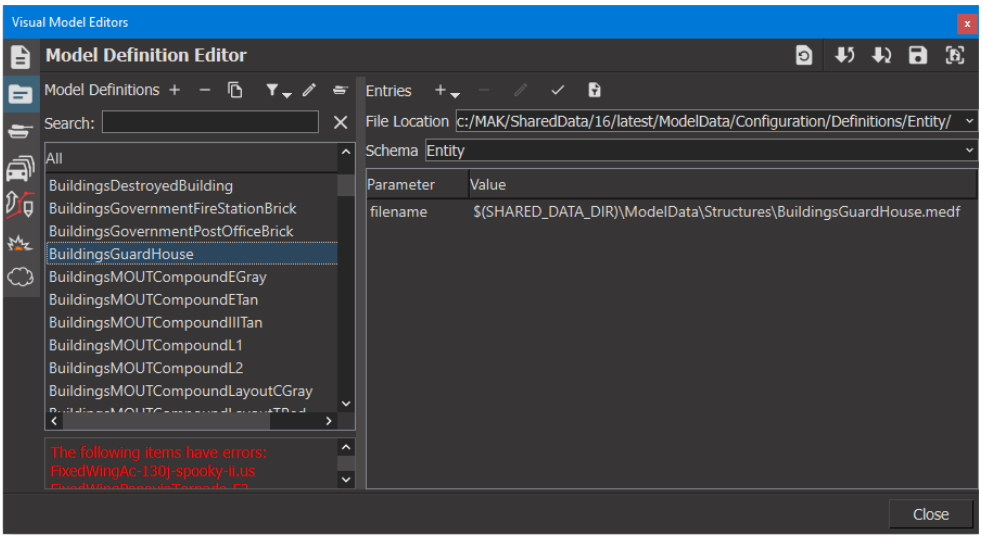
Many of the visual definitions in an element definition have a model definition parameter. A model definition specifies a set of parameters that are appropriate to the object type. (The parameters are determined by the model definition's schema. For information about model schemas, please see "14.1. Creating and Editing Schemas" on page 210.) For most MAK ONE application users, the most important thing to know about model definitions is that for 3D models, the model definition specifies the 3D model, such as an OpenFlight file, that is used to display an entity. If you want to add new models to MAK ONE applications, you probably need to create a new model definition for each one.

> If you are using VR-Forces, iIt is strongly recommended that you use the Simulation Object Editor to edit visual models, which includes the model definition.

MAK ONE applications come with an extensive set of model definitions that are mapped to the included models. You can edit or delete these model definitions and you can create new ones.

Model definitions are created and edited on the Model Definition Editor page of the Visual Model Editors dialog box (Figure 80).

Figure 80. Model Definition Editor page



The model definitions provided with MAK ONE applications are stored in *./SharedData/16/latest/ModelData/Configuration/Definitions*. When you open the Visual Model Editors dialog box, it reads the files in this directory in alphabetical order. If files contain duplicate entries, the entry in the last file loaded takes precedence. You can use command-line options to change the location of the model definitions that get loaded at startup. For details, please see "15.10.1 Importing Model Definitions from the Command Line" on page 227.

- Any model definition that uses the Entity, Tree, or SpeedTree schema is automatically added to the Props Palette in VR-Forces.
- The model definition for an entity includes the 3D model used to represent it. You can change the model in the Visual Model Editors dialog box. However, you can also quickly and easily change the 3D model for an entity in the Simulation Object Editor. For details, see "Editing a Simulation Object's Visual Model" in the *VR-Forces Users Guide*.
- Changes to element definitions do not take effect until you restart the MAK ONE application or reload the visual models. For information about reloading visual models, see "Reloading Visual Models in the VR-Forces GUI" in the *VR-Forces Users Guide*.

15.2. Work Flow Issues for Model Definitions

A default MAK ONE application installation has multiple model definition files. When you start a MAK ONE application, they are loaded. When you open the Visual Model Editors dialog box, all of the model definitions are listed. However, the editor does not keep track of which model definitions come from which model definition file. When you create or edit model definitions, the MAK ONE application does not automatically save them. Therefore, you are responsible for saving new and edited model definitions to the correct location.

The default model definition files are:

- *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity/AggregateLevelModelDefinitions.ommx*. Model definitions specific to aggregate-level scenarios in VR-Forces.
- *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity/baseModelDefinitions.ommx*. Model definitions used by both entity- and aggregate-level scenarios in VR-Forces.
- *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity/coreModelDefinitions.ommx*. Model definitions for objects that do not have .entity files in VR-Forces.
- *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity/EntityLevelModelDefinitions.ommx*. Model definitions for simulation objects in entity-level scenarios in VR-Forces.

Before you add, delete, or edit a model definition, consider which of the following work flows best meets your need:

- If you want to create new model definitions, the preferred work flow is to export them to a new file. This is the easiest way to avoid overwriting other model definition files and makes them easily portable to newer releases of MAK

ONE applications.

If you want to add the new model definitions to an existing model definition file:

1. Replace the model definitions in the editor r by using the Load and Replace option to load the model definition file you want to edit. (For details, see "15.10. Importing Model Definitions" on page 226.)
2. Add the new model definitions.
3. Export all changes back to the original model definition file.

- If you have edited existing model definitions, you can export them to their original file or to a new file:
  - If you export to a new file, make sure that the file name is alphabetically later in alphabetical order than the original model definition file. This ensures that when the new model definition is loaded it will overwrite the original one.
  - If you want to export to the original model definition file:

1. Replace the model definitions in the editor by using the Load and Replace option to load the model definition file you want to edit. (For details, see "15.10. Importing Model Definitions" on page 226.)
2. Make your changes.
3. Export all changes back to the original model definition file.

- If you want to delete a model definition, you must delete it from the source model definition file. Follow the work flow for editing an original model definition file.

For more information about how to export model definitions, see "15.9.1 Exporting Selected Model Definitions" on page 226.



Do not use the Export All option if you are editing the combined model definition files that are loaded by default unless you are very certain about how you want to use the resultant file. If you export to a file in the *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity*directory, it will conflict with the other files in that directory.

15.3. Model Definition Files

A default MAK ONE application installation has multiple model definition files. (This is to accommodate the simulation model set organization in VR-Forces.) When you start a MAK ONE application, they are all loaded. When you open the Visual Model Editors dialog box, it lists all of the model definitions.

The default model definition files, found in the *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity* directory, are:

- *AggregateLevelModel Definitions.ommx*. Model definitions specific to aggregate-level scenarios in VR-Forces.
- *baseModel Definitions.ommx*. Model definitions used by both entity and aggregate-level scenarios in VR-Forces.
- *coreModel Definitions.ommx*. Model definitions for objects that do not have *.entity* files in VR-Forces.
- *EntityLevelModel Definitions.ommx*. Model definitions for simulation objects in entity-level scenarios in VR-Forces.

As of VR-Vantage 3.0 and VR-Forces 5.0, the model definition configuration file location changed from *./data/Configuration/Definitions/Entity* to *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity*. If you are upgrading from a previous version of your MAK ONE application, you can copy your *.ommx* files into this new location and the application will pick them up automatically.

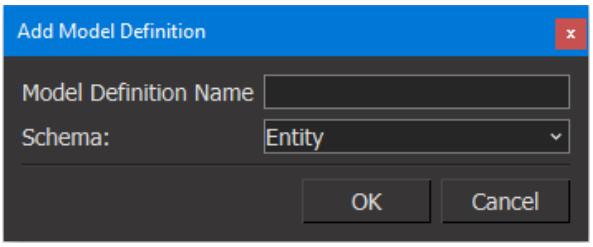
15.4. Creating a Model Definition

MAK ONE applications do not save changes to model definitions automatically. If you want to keep changes, you must export them to a file.

➤ **To create a model definition:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).
3. Optionally, filter the model definitions in the list. (For details, please see "15.7. Filtering the List of Model Definitions" on page 223.)
4. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens (Figure 81).

Figure 81. Add Model Definition dialog box



5. Type a name for the new model definition.

Model definitions for DI-Guy characters must end with the letters DIG. Model definitions for SpeedTrees must end in ST.

6. In the Schema list, select a schema for the model. (This list matches the list of schemas on the Schema Editor page.)
7. Add parameters to the model, as described in "15.8.1 Adding Parameters to a Model Definition" on page 224.
8. Save the model definition or export it.

15.5. 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 80 on page 218).
3. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, see "15.7. Filtering the List of Model Definitions" on the facing page.)
4. Select the model definition that you want to delete.
5. Click the Delete button ( ) next to the Model Definitions label. You are prompted to confirm the deletion.
6. Click Yes. The model definition is removed from the list.
7. Click the Export Selected button ( ). The Export Model Definitions dialog box opens.
8. Select the model definition file to save to.
9. Click Save.

15.6. 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.

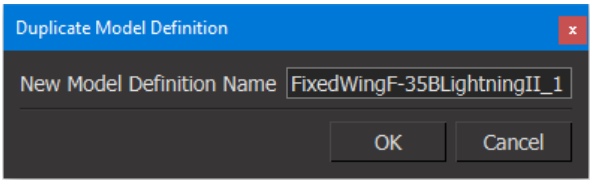
MAK ONE applications do not save changes to model definitions automatically. If you want to keep changes, you must export them to a file.

➤ **To copy a model definition:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).
3. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "15.7. Filtering the List of Model Definitions" on the facing page.)

4. Select the model that you want to copy.
5. Click the Duplicate button ( ). The Duplicate Model Definition dialog box opens (Figure 82).

Figure 82. 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 "15.8. Editing a Model Definition" below.
10. Export or save the model definition.

15.7. Filtering the List of Model Definitions

By default, the Model Definition Editor page lists all models that a MAK ONE application knows about. You can filter the list by schema or by source file to more easily work with a specific type of model.

> You can also filter the list by typing a text string in the Search box. All model definitions that contain the text string are listed.

➤ To filter the list of model definitions by schema, click the Filter button ( ) next to the Model Definitions label and select a schema type from the list.

You can filter the list of model definitions by source file or show All definitions.

➤ **To filter the list of model definitions by source file:**

1. In the File Location list, select the source file that you want to filter by. (To show all model definitions, select All.)
2. Click the Filter button ( ) next to the Entries label.

15.8. 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.

> MAK ONE applications do not save changes to model definitions automatically. If you want to keep changes, you must export them to a file.

If you change the model definition for a simulation object that is already in the VR-Vantage scene, the updated definition is not applied to the object. To apply an edited model definition to all simulation objects, disconnect from the exercise and then reconnect.

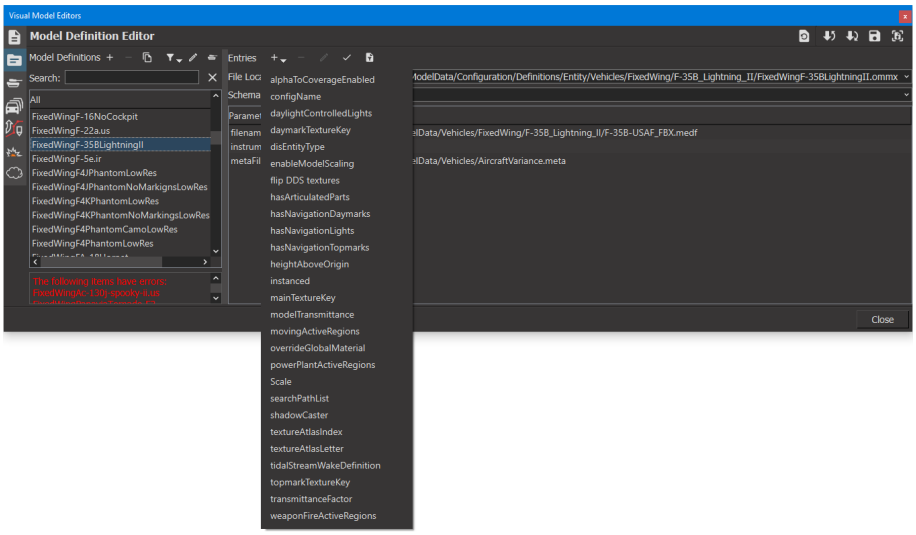
15.8.1 Adding Parameters to a Model Definition

When you create a new model definition, you must specify the parameters required by its schema. You can only add parameters that are permitted by the schema.

➤ **To add a parameter to a model:**

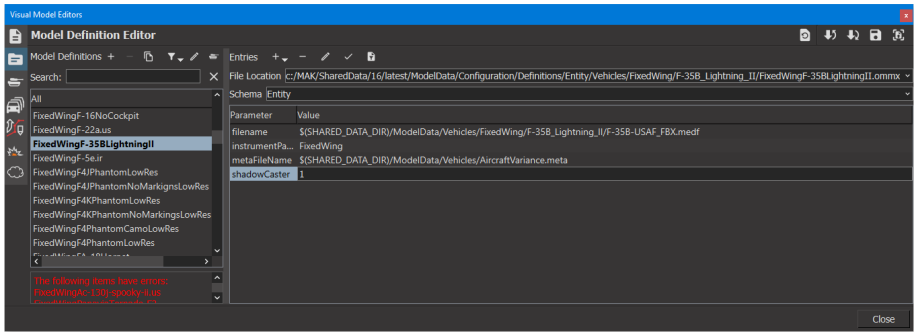
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).
3. Optionally, filter the model definitions list. (For details about filtering, please see "15.7. Filtering the List of Model Definitions" on the previous page.)
4. Select the model definition for which you want to add a parameter.
5. Click the Add Parameter button ( ). A list of parameters displays (Figure 83).

Figure 83. 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 84).

Figure 84. Editable parameter



8. Type or select a value for the parameter.
9. Click away from the field.
10. When you have finished editing the model definition, save or export it.

15.8.2 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 80 on page 218).
3. Optionally, filter the model definitions list. (For details about filtering, please see "15.7. Filtering the List of Model Definitions" on page 223.)
4. Select the model definition for which you want to edit a parameter.
5. Click the value for the parameter you want to edit. The field becomes editable (Figure 84 above).
6. Change the value for the parameter.
7. Click away from the field.
8. When you have finished editing the model definition, save or export it.

15.8.3 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 80 on page 218).
3. Optionally, filter the model definitions list. (For details about filtering, please see "15.7. Filtering the List of Model Definitions" on page 223.)
4. Select the model from which you want to delete a parameter.
5. Select the parameter that you want to delete.

6. Click the Delete button ( ) next to the Entries label. The parameter is removed.
7. Save the model definition.

15.9. Saving Model Definitions

If you add, edit, or delete model definitions, you must explicitly save the change to the existing source file or export the new or updated definitions to a new file.

You can save all model definitions and you can export individual model definitions. This allows you to create your own library of custom model definitions that you can easily migrate to newer releases of aMAK ONE application or copy to other MAK ONE application installs.

➤ **To export all model definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).
3. Click the Save button ( ). A confirmation prompt is displayed.
4. Click Yes.

15.9.1 Exporting Selected Model Definitions

➤ **To export selected model definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).
3. Optionally, filter the model definitions in the list. (For details, see "15.7. Filtering the List of Model Definitions" on page 223.)
4. Select the model definitions you want to save.
5. Click the Export Selected button ( ). The Export Model Definitions dialog box opens.
6. Select a file or type a name for the model definition file.
7. Click Save.

15.10. Importing Model Definitions

You can import model definitions and replace the loaded model definitions. You can also import model definitions and merge them with the loaded model definitions. For details about how merging works, see "Managing VR-Vantage Settings" in the *VR-Vantage Users Guide* or "Managing VR-Forces Settings" in the *VR-Forces Users Guide*.

➤ **To import model definitions and replace the existing definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).

3. Click the Import and Replace button ( ). The Import Model Definitions dialog box opens.
4. Select the model definition file you want to import.
5. Click Open.

➤ **To merge model definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).
3. Click the Import and Merge button ( ). The Merge Model Definitions dialog box opens.
4. Select the model definition file you want to merge.
5. Click Open.

15.10.1 Importing Model Definitions from the Command Line

You can merge in model definitions with the `--mergeModelDefs {`*`directory`*`|`*`filename`*`}` command-line option. If you specify a directory, a MAK ONE application merges all model definition files in the directory. If you specify a file, it just merges that file. You can have multiple instances of this option in a command.

You can specify that MAK ONE applications not load the model definitions in the *./SharedData/16/latest/ModelData/Configuration/Definitions* directory. To do this, start the application with the `--noAppDataModelDefs` command-line option.



16. Element Definitions

In "Section III. Visual Models in MAK ONE Applications" on page 197, we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit element definitions.

16.1. Creating and Editing Element Definitions 230
16.2. How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions Are Organized 230
16.2.1 Leaf File Descriptions 231
16.3. Work Flow Issues for Creating and Editing Element Definitions 232
16.4. Creating an Element Definition 233
16.5. Filtering the List of Element Definitions 234
16.6. Editing an Element Definition 234
16.6.1 Adding a Visual Definition 234
16.6.2 Editing a Visual Definition 235
16.6.3 Adding an Attribute to a Visual Definition 237
16.6.4 Deleting an Attribute from a Visual Definition 239
16.6.5 Copying and Pasting Visual Definitions 239
16.7. Jumping to a Model Definition 240
16.8. Changing an Element Definition's File Location 240
16.9. Deleting an Element Definition 241
16.10. Saving Element Definitions 241
16.10.1 Saving Entity, Unit, Spot Report, or Tactical Graphics Element Definitions 242
16.10.2 Saving Interaction and Smoke Element Definitions to .elem Files 243
16.11. Importing Element Definitions 243
16.11.1 Importing a Hierarchy File 244
16.11.2 Importing and Merging Element Definitions 244
16.11.3 Importing Element Definitions from the Command Line 244

16.1. Creating and Editing Element Definitions

Element definitions define how MAK ONE applications render scene elements (simulation objects, tactical graphics, interactions, cultural features, spot reports, and so on).

> If you are using VR-Forces, it is strongly recommended that you use the Simulation Object Editor to edit visual models, which includes the element definition.

Element definitions are organized in a tree-style hierarchy. At any level of the hierarchy an element definition inherits the visual definitions of its parents. This makes it possible to assign a visual definition to many elements at one time. You can change the inherited parameter values for a specific element at any level of the hierarchy. However, in most cases, you cannot delete an inherited visual definition.

You can create and edit element definitions. The procedures are the same for each element type except for the save procedure. Element definitions are not saved automatically. You must save them explicitly.

> In VR-Vantage, if you are connected to an exercise, changes to element definitions do not take effect until you disconnect and reconnect.
>
> In VR-Forces, if you edit an element definition in the Simulation Object Editor and save the change, you can update an open scenario to use the new element definition. For details, see "Reloading Visual Models in the VR-Forces GUI" in the *VR-Forces Users Guide*.

16.2. How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions Are Organized

Entity, unit, spot report, and tactical graphics element definitions are stored in two types of files: a hierarchy file (*.hier*) and one or more leaf definition files (*.leaf*). The hierarchy file contains the hierarchy of categories for the element definitions, such as Entity > Surface > Boat and the attribute inheritance information associated with each level of the hierarchy. The leaf files contain the definitions for each element at the leaf level[1].

Other types of element definitions do not save separate hierarchy and leaf files. They are saved to a file with the extension *.elem*.

When you add or edit entity, unit, spot report, and tactical graphics element definitions, you must save the changes to the hierarchy and leaf files separately. If you have only made leaf level changes, you do not have to save the hierarchy file.

[1]Technically, the leaf level of a tree structure is a node with no children. For element definitions, the leaf level consists of specific definitions. A category that has no members would be part of the hierarchy, not the leaf file.

You can create multiple hierarchy and leaf level files to meet your needs. However you can only load one hierarchy file at a time. When a MAK ONE application starts up, it loads the files in the various element definition directories. If it finds more than one hierarchy file for an element type, it only loads the first one (in alphabetical order). If it finds multiple leaf files, it loads them in alphabetical order. If there are any duplicate entries, the last file loaded takes precedence.

16.2.1 Leaf File Descriptions

The element definitions for entity, unit, and tactical graphics are organized into multiple leaf files. This organization is to accommodate the data structure of VR-Forces, MAK's computer generated forces (CGF) application. By default, a MAK ONE application loads all of the element definitions so that it can render whatever simulation objects VR-Forces sends. The element definitions are organized as follows:

- *./SharedData/16/latest/ModelData/Configuration/Definitions/Unit*. Element definitions for hierarchical units such as platoons and companies:
  - *EntityLevelUnitElementDefinitions.leaf*. Units used in entity-level scenarios.
  - *coreUnitElementDefinitions.leaf*. Element definitions for objects that do not have *.entity* files in VR-Forces.
  - *AggregateLevelUnitElementDefinitions.leaf*. Units used in aggregate-level scenarios.
  - *baseUnitElementDefinitions.leaf*. Unit elements used in both entity-level and aggregate level scenarios.
- *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity*. Element definitions for individual simulation objects such as humans, ships, and cars:
  - *EntityLevelEntityElementDefinitions.leaf*. Element definitions used in entity-level scenarios.
  - *coreEntityElementDefinitions.leaf*. Element definitions for objects that do not have *.entity* files in VR-Forces.
  - *baseEntityElementDefinitions.leaf*. Element definitions used in both entity-level and aggregate level scenarios.
- *./SharedData/16/latest/ModelData/Configuration/Definitions/TacticalGraphics*. Element definitions for tactical graphics:
  - *EntityLevelTacticalGraphicsElementDefinitions.leaf*. Tactical graphics used in entity-level scenarios.
  - *coreTacticalGraphicsElementDefinitions.leaf*. Element definitions for tactical graphics that do not have *.entity* files in VR-Forces.
  - *AggregateLevelTacticalGraphicsElementDefinitions.leaf*. Tactical graphics used in aggregate-level scenarios.
  - *baseTacticalGraphicsElementDefinitions.leaf*. Tactical graphics used in both entity-level and aggregate level scenarios.

- *./appData/definitions/SpotReports/ElementDefinitions*. Element definitions for VR-Forces spot reports are created in memory when you start the application. They are not saved unless you make changes in the Simulation Object Editor. If you do, the default files are:
  - *EntityLevelSpotReportsElementDefinitions.leaf*. Tactical graphics used in entity-level scenarios.
  - *coreSpotReportsElementDefinitions.leaf*. Element definitions for tactical graphics that do not have *.entity* files in VR-Forces.
  - *baseSpotReportsElementDefinitions.leaf*. Tactical graphics used in both entity-level and aggregate level scenarios.



As of VR-Vantage 3.0 and VR-Forces 5.0, the element definition configuration file location changed:

- *./data/Configuration/Definitions/Unit* to *./SharedData/16/latest/ModelData/Configuration/Definitions/Unit*.
- *./data/Configuration/Definitions/Entity* to *./SharedData/16/latest/ModelData/Configuration/Definitions/Entity*.
- *./data/Configuration/Definitions/TacticalGraphics* to *./SharedData/16/latest/ModelData/Configuration/Definitions/TacticalGraphics*.

If you are upgrading from a previous version of your MAK ONE application, you can copy your *.ommx* files into this new location and they will be picked up automatically.

16.3. Work Flow Issues for Creating and Editing Element Definitions

A default MAK ONE data installation has multiple element definition files for entities, units, spot reports, and tactical graphics. When you start a MAK ONE application, they are all loaded. When you open the Visual Model Editors dialog box, you see a list of all element definitions.

When you create or edit element definitions, a MAK ONE application does not automatically save them. Therefore, you are responsible for saving new and edited element definitions. This section describes the issues to consider when you work with entity, unit, spot report, and tactical graphics element definitions to get the results you want and avoid problems.

- If you have created new element definitions, the preferred work flow is to export them to a new leaf file. This makes them easily portable to newer releases of the application.
- If you have edited existing element definitions, you can export them to their original leaf file or to a new leaf file. If you export to a new file, make sure that the file name is alphabetically later in order than the original element definition

file. This ensures that when the new element definition is loaded, it overwrites the original one.

Interaction and smoke element definitions do not have hierarchy and leaf files. They work as follows:

- They are loaded from files that have the extension *.elem*.
- You can export specific interaction element definitions to individual files. If you have more than one *.elem* file in a directory, all of them are loaded.
- You cannot export specific smoke element definitions to individual files. When you export them, all definitions are exported to the selected file.

16.4. Creating an Element Definition

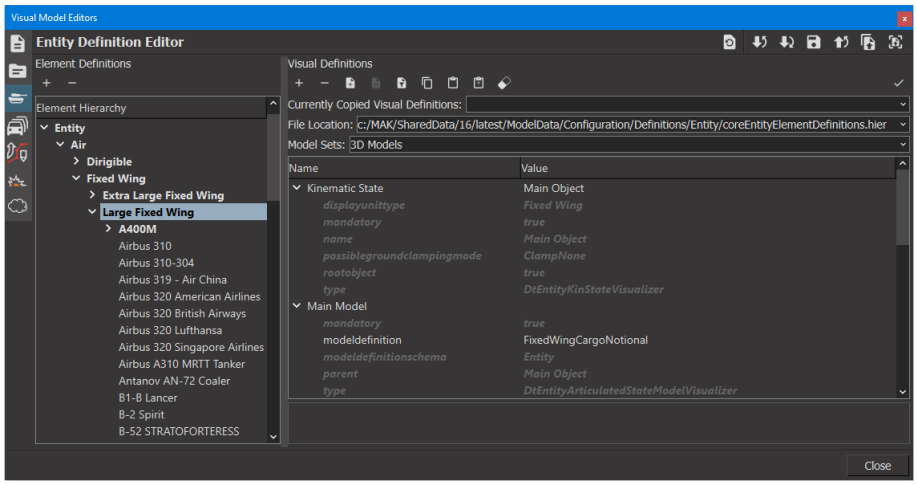
You can create new element definitions. Entity, unit and tactical graphics element definitions get exported to *.leaf* files. Smoke, spot report, and interaction element definitions get exported to *.elem* files.

> MAK ONE applications do not save changes to element definitions automatically. If you want to keep changes, you must export them to a file.

➤ **To create an element definition:**

1. Choose **Settings > Visual Model Editors.** The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to create.
3. Expand the element definition hierarchy to the level at which you want to add an element definition and select the parent element (Figure 85).

Figure 85. Entity Definition Editor page



4. Click the Add button ( ) under Element Definitions. The Create Element Definition dialog box opens.

5. Type a name for the element definition. (You cannot edit the name after you create it, so try to avoid typing errors.)
6. Click OK. The new element is added to the list.
7. Optionally, edit the visual definitions assigned to the element. For details, see "16.6.2 Editing a Visual Definition" on the facing page.
8. Optionally, add a visual definition to the element. For details, see "16.6.1 Adding a Visual Definition" below.
9. Save or export the element definition. If you made any changes to an entity, unit, or tactical graphics hierarchy, you also must export the hierarchy.

16.5. Filtering the List of Element Definitions

By default, the Visual Model Editors dialog box lists all element definitions. You can filter the list of element definitions based on the files from which the element definitions were imported.

➤ **To filter the list of model definitions by source file:**

1. In the File Location list, select the source file that you want to filter by. (To show all model definitions, select All.)
2. Click the Filter button ( ) below the Visual Definitions label.

16.6. Editing an Element Definition

An element definition consists of a list of visual definitions for that element. You can add visual definitions and you can edit their parameters. In most cases, you cannot delete an inherited visual definition.

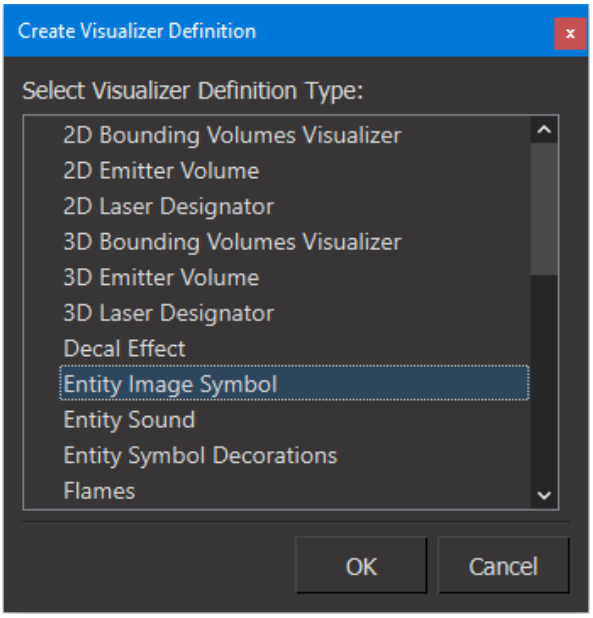
> MAK ONE applications do not save changes to element definitions automatically. If you want to keep changes, you must export them to a file.

16.6.1 Adding a Visual Definition

➤ **To add a visual definition to an element definition:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to edit (Figure 85 on the previous page).
3. Select the element definition that you want to edit.
4. Optionally, on the Model Sets list, select the model set you want to use.
5. Click the Add button ( ) next to the Visual Definitions label. The Create Visualizer Definition dialog box displays a list of available visual definitions.

Figure 86. Create Visualizer Definition dialog box



6. Select the visual definition that you want to add.
7. Click OK. The visual definition is added to the element.
8. Optionally, edit the parameters for the visual definition. For details, please see "16.6.2 Editing a Visual Definition" below.
9. When you have finished editing the element definition, save or export it.

16.6.2 Editing a Visual Definition

Visual definitions and their attributes are displayed either as italicized text or plain text (Figure 87). Italicized text indicates that the visual definition or attribute is inherited from a parent element. Visual definitions and attributes that are not italicized are particular to that element. (These entries could be inherited by child elements.)

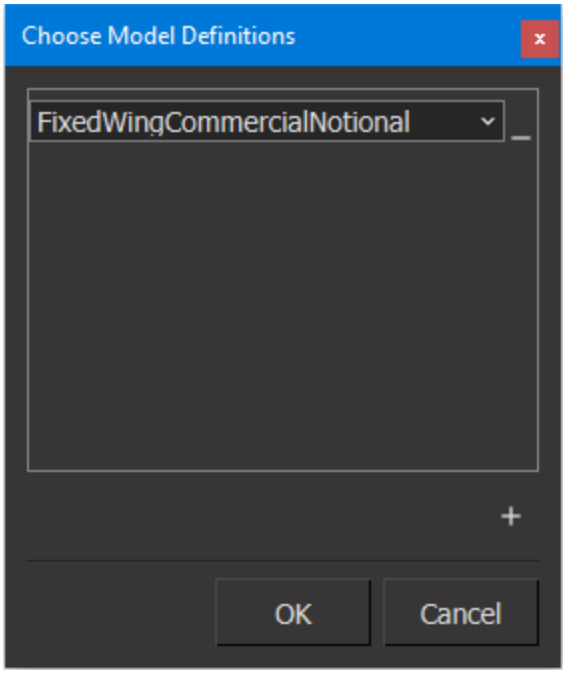
Figure 87. Visual definition entry



➤ **To edit a visual definition parameter:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to edit (Figure 85 on page 233).
3. Select the element definition that you want to edit.
4. In the Visual Definitions pane, click the value of the attribute that you want to edit. Depending on the attribute, a list of available values is displayed, a dialog box opens (Figure 88), or the field becomes editable.

Figure 88. Choose Model Definitions dialog box



5. For a list, select the new value. For an editable field, type a new value. For the dialog box:
   a. If the dialog box window does not have a list, click the Add button (+). A list of values is added to the dialog box window.
   b. Select the value that you want.
   c. Click OK.
6. When you have finished editing the element definition, save or export it.

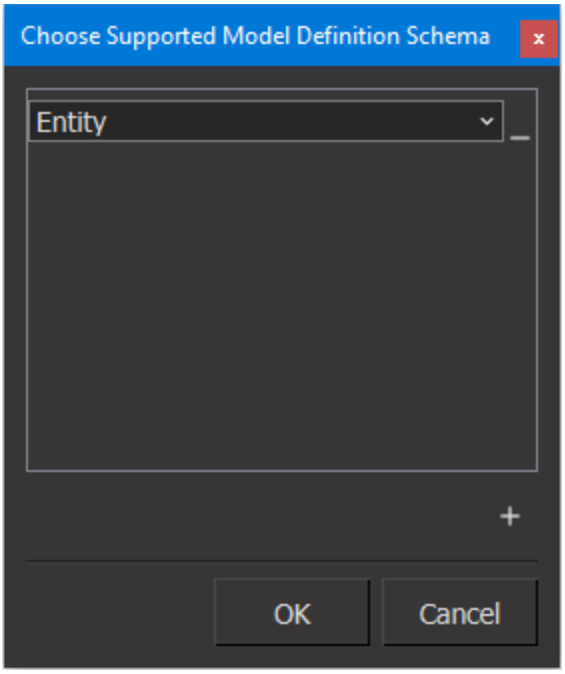
Mapping to Multiple Model Definitions or Schema

There may be cases where you want to map a visual definition to multiple model definitions or multiple schema. If you do this, the application randomly assigns the model definition or schema to instances of the object.

➤ **To map a visual definition attribute to multiple values:**

1. Choose **Settings > Visual Model Editors.** The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to edit (Figure 85 on page 233).
3. Select the element definition that you want to edit.
4. In the Visual Definitions pane, click the value of the **modeldefinition** or **modeldefinitionschema** parameter that you want to edit. A dialog box opens. If the attribute already has values, they are listed in the dialog box window as a series of lists (Figure 89).

Figure 89. Parameter with multiple values



5. To add a value, click the Add button (+). A new list is added to the window.
6. Select a value from the new list.
7. To remove a value, click the Delete button (—) next to the list for that value.
8. Click OK. The visual definitions list does not show all of the values, but clicking it opens the dialog box again and shows the selected values.
9. When you have finished editing the element definition, save or export it.

16.6.3 Adding an Attribute to a Visual Definition

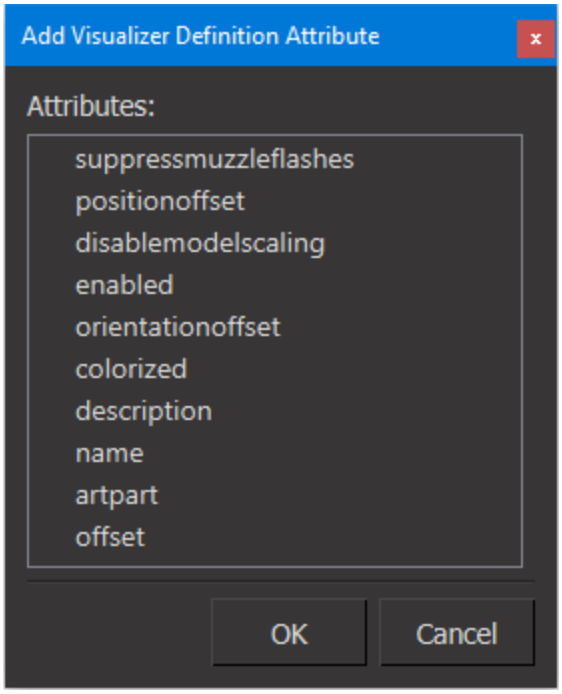
Each visual definition has a set of attributes. Some of them may be optional, so they are not specified as part of the factory definitions. You can add optional attributes to a visual definition.

> If you add an attribute to an element definition that is part of the element definition hierarchy, for example Entity > Land > Tracked, you must export all of the element definitions that inherit attributes from that definition. As a practical matter, this usually means exporting all of the definitions in that element definition file.

➤ **To add an attribute to a visual definition:**

1. 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 85 on page 233).
3. Select the visual definition that you want to edit.
4. Click the Add Attribute button ( ). The Add Visualizer Definition Attribute window opens. It lists the attributes that are available for the visual definition (Figure 90).

Figure 90. Add Visualizer Definition Attribute window



5. Select the attribute that you want to add.
6. Click OK. The attribute is added to the visual definition.
7. Specify a value for the attribute. This might require typing a value, selecting a value from a list, or selecting a value in another window.
8. When you have finished editing the element definition, save or export it.

16.6.4 Deleting an Attribute from a Visual Definition

You can delete individual attributes from a visual definition.

> The Delete button above the list of visual definitions deletes entire visual definitions. Do not click it thinking it will delete only the selected attribute.

➤ **To delete an attribute:**

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 85 on page 233).
3. Select the visual definition that you want to edit.
4. Select the attribute that you want to delete.
5. Click the Delete Attribute button ( ).
6. When you finish editing the element definition, save or export it.

16.6.5 Copying and Pasting Visual Definitions

You can copy visual definitions from an element definition and paste them into other element definitions. When you paste a visual definition, if a visual definition of the same type already exists, it is not overwritten. The pasted visual definition is added to the list of visual definitions for the element definition. If you only want one visual definition of that type, you should delete the one you no longer want.

➤ **To copy a visual definition**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to examine (Figure 85 on page 233).
3. Select the element definition whose visual definition you want to copy.
4. Select the visual definition you want to copy.
5. Click the Copy Visual Definition button ( ).

➤ **To paste a visual definition:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to examine (Figure 85 on page 233).
3. Select the element definition to which you want to paste a visual definition.
4. In the Currently Copied Visual Definitions list, select the visual definition that you want to paste.
5. Click the Paste Visualizer Definition button ( ).

Clearing the List of Copied Visual Definitions

Copied visual definitions are maintained in memory. They are listed in the Currently Copied Visual Definitions list. You can clear them from memory.

➤ **To clear the copied visual definitions from memory:**

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 clear visual definitions for (Figure 85 on page 233).
3. Click the Clear all Visualizer Definitions button ( ).

16.7. Jumping to a Model Definition

You can jump from an element definition directly to any of the model definitions that it uses.

➤ **To jump from an element definition to an associated model definition:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to examine (Figure 85 on page 233).
3. Right-click the element definition.
4. On the context menu, select the model definition you want to jump to. The Visual Model Editors dialog box switches to the Model Definitions page and selects the chosen definition.

16.8. Changing an Element Definition's File Location

The Visual Model Editors dialog box imports element definitions from several files. It remembers the source file for each element definition and displays it in the File Location list. You can change the file associated with a element definition.

The dialog box only lists files from which it has imported element definitions. If you want to save a element definition to a new file, you must export it.

➤ **To change the file location of an element definition:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Element Definition Editor page (Figure 85 on page 233).
3. Optionally, filter the model definitions list. (For details about filtering, please see "15.7. Filtering the List of Model Definitions" on page 223.)
4. Select the element definition whose file location you want to change.
5. Select a file in the File Location list.
6. Click the Change File Path button ( ).
7. Save the element definitions.

16.9. Deleting an Element Definition

When you delete an element definition, you change the hierarchy and remove a leaf file. You must update both.

> A MAK ONE application does not save changes to element definitions automatically. If you want to keep changes, you must export them to a file.

> You cannot delete an element definition if it is being used.

➤ **To delete an element definition:**

1. Choose **Settings > Visual Model Editors.** The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition that you want to delete (Figure 85 on page 233).
3. Select the element definition that you want to delete.
4. Click the Delete button ( ) next to the Entity Definitions label.
5. Confirm the deletion.
6. Click the Save All hierarchy and Leaf Files button ( ).

16.10. Saving Element Definitions

Smoke and interaction element definitions are saved to a file with the extension *.elem*. Entity, unit, spot report, and tactical graphics element definitions, are saved to separate hierarchy (*.hier*) and leaf (*.leaf*) files.

You cannot save portions of the hierarchy. The entire hierarchy is exported to file.

When the Visual Model Editors dialog box loads entity, unit, spot report, and tactical graphics element definitions it merges several files. The element definition files are sorted alphabetically and loaded in that order. In the case of conflicts, the last file loaded wins. Therefore:

- If you are only adding or editing a few element definitions, save them to one or more new leaf files rather than overwriting one of the default leaf files.
- When you export to new leaf files, consider carefully how you name them.

A MAK ONE application only ships one element definition file each for interactions (*InteractionsElementDefinitions.elem*) and tactical smoke (*SmokeElementDefinitions.elem*). However, you can save individual interactions, so it is possible to have more than one *.elem* file in the definitions directory for element types. In that case, a MAK ONE application loads all of the *.elem* files it finds. It loads the files in alphabetical order. If there are duplicate element definitions, the last one loaded is used. Therefore if you customize an element definition and export it to a file and you want that definition to be used, give it a name that is later in the alphabet than *InteractionsElementDefinitions.elem* or *SmokeElementDefinitions.elem*.

Changes to element definitions do not get saved automatically. You must explicitly save the changes. Options for saving changes are:

- Save All. Saves the hierarchy and all leaf files to the file locations specified in the File Location list for each element definition. In most cases this will be the original file location. However, you can change the file location if you want to.
- Export Hierarchy. Saves the hierarchy.
- Export All Leaves. Saves all leaves to the file locations specified in the File Location list for each element definition.
- Export Selected Leaves. Saves the selected element definitions to a user-specified file.

16.10.1 Saving Entity, Unit, Spot Report, or Tactical Graphics Element Definitions

If you want to export entity, unit, spot report, or tactical graphics element definitions, you must save the hierarchy and leaf information separately. If the hierarchy has not changed, you do not need to save it.

➤ **To export a hierarchy file:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to export.
3. Click the Export Hierarchy button ( ). A confirmation prompt displays.
4. Click Yes. The Export Element Definitions dialog box opens.
5. Select the hierarchy file you want to export to or type the name of a new file.
6. Click Save.

➤ **To save the hierarchy and all element definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to export.
3. Click the Save All button ( ). A confirmation prompt displays.
4. Click Yes.

➤ **To save all element definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to export.
3. Click the Export All Leaves button ( ). A confirmation prompt displays.
4. Click Yes.

➤ **To export selected definitions to a leaf file:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to export.

3. Select the element definitions you want to export.
4. Click the Export Selected Leaves button (). The Export Element Definitions dialog box opens.
5. Select the leaf file you want to export to or type the name of a new file.
6. Click Save.

16.10.2 Saving Interaction and Smoke Element Definitions to *.elem* Files

When you save interaction element definitions or tactical smoke element definitions, they get saved to a file with the extension *.elem*. You can save all definitions to one file or you can export selected definitions to a file. If you have multiple element definition files in a directory, they are loaded in alphabetical order. If there are duplicate element definitions, the application uses the last one it loaded.

➤ **To save all tactical smoke element definitions or interaction element definitions to a file:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the element definition type (Tactical Smoke Editor page or Interaction Definition Editor page).
3. Click the Save button (). You are prompted to confirm the save operation.
4. Click Yes.

➤ **To export selected interaction or tactical smoke element definitions to a file:**

1. Choose **Settings** > **Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the element definition type (Tactical Smoke Editor page or Interaction Definition Editor page).
3. Select the element definitions that you want to export.
4. Click the Export Selected button (). The Export Element Definitions dialog box opens.
5. Specify a file to export to.
6. Click Save.

16.11. Importing Element Definitions

You can import element definitions and replace the loaded element definitions. You can import element definitions and merge them with the loaded element definitions.

If you want to import entity, unit, spot report, or tactical graphics element definitions, you can import hierarchy files, leaf files, or both. If you import a hierarchy file, it removes all leaf file entries and replaces the previously loaded hierarchy.

When you import a leaf file, it is merged according to the criteria described in "Managing VR-Vantage Settings" in the *VR-Vantage Users Guide* or "Managing VR-Forces Settings" in the *VR-Forces Users Guide*.

You can also import or merge files in the legacy *.elem* format. If you import and replace entity, unit, or tactical graphics element definitions using an *.elem* file, the hierarchy and all element definitions are replaced with those in the elem file. If you merge an *.elem* file, the merge is additive only. Entities with names that already exist are skipped.

Interaction and smoke element definitions use the *.elem* format.

16.11.1 Importing a Hierarchy File

You can replace the current hierarchy with a different one by importing a hierarchy file or a file in the legacy *.elem* format.

When you import a hierarchy file, all leaf files are removed.

➤ **To import a hierarchy or element definition file and replace the existing definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to import.
3. Click the Import and Replace button (). The Import Element Definitions dialog box opens.
4. Select the hierarchy file or element definition file (extension *.elem* or *.hier*) you want to import.
5. Click Open.

16.11.2 Importing and Merging Element Definitions

You can import and merge element definitions from leaf files or legacy element definition (*.elem*) files.

You cannot import a leaf file unless a hierarchy is already loaded.

➤ **To merge element definitions:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the page for the type of element definition you want to import.
3. Click the Import and Merge button (). The Merge Element Definitions dialog box opens.
4. Select the element definition file (extension *.elem* or *.leaf*) you want to merge.
5. Click Open.

16.11.3 Importing Element Definitions from the Command Line

You can import and merge entity, unit, and tactical graphics element definitions when you start up a MAK ONE application using the following command-line options:

- `--entityDefsDir` *`directory`* merges in all *.hier* and *.leaf* files in the specified directory. The directory must be *./SharedData/16/latest/ModelData/Configuration/Definintions/Entity*, *./SharedData/16/latest/ModelData/Configuration/Definintions/Unit*, or *./SharedData/16/latest/ModelData/Configuration/Definintions/TacticalGraphics*, or be subordinate to one of them.
- `--mergeHierarchyEntityDef` *`filename`* imports the specified *.hier* file.
- `--mergeLeafEntityDef` *`filename`* merges in the specified *.leaf* file.
- `--mergeHierarchyTacticalGraphicsDef` *`filename`* merges in the specified tactical graphics *.hier* file.
- `--mergeLeafTacticalGraphicsDef` *`filename`* merges in the specified tactical graphics *.leaf* file.
- `--mergeHierarchyUnitDef` *`filename`* merges in the specified unit *.hier* file.
- `--mergeLeafUnitDef` *`filename`* merges in the specified unit *.leaf* file.

These commands merge element definitions into the default element definitions, which are usually those in *./SharedData/16/latest/ModelData/Configuration/Definintions*. You can also specify that the element definitions in *./SharedData/16/latest/ModelData/Configuration/Definintions* not be loaded. In that case, only those files specified on the command line would get loaded. If none are specified, no element definitions would be loaded.

To specify that the element definitions in *./SharedData/16/latest/ModelData/Configuration/Definintions* not be loaded, start MAK ONE applications with the `--noAppDataEntityDefs`, `--noAppDataUnitDefs`, or `--noAppDataTacticalGraphicsDefs` command-line options.



17. Mapping Models and Effects

This chapter explains how to map element definitions to DIS and HLA object types.

17.1. Introduction to Object Type Mapping 248
17.2. Work Flow Issues for Object Type Mappings 248
17.3. Adding an Object Type Mapping 249
17.4. Editing an Object Type Mapping 250
17.5. Filtering the Element Definition List 251
17.6. Changing an Object Type Mapping's File Location 252
17.7. Jumping to an Element Definition 252
17.8. Deleting an Object Type Mapping 253
17.9. Saving Object Type Mappings 253
    17.9.1 Exporting Selected Object Type Mappings 253
17.10. Importing Object Type Mappings 254
17.11. Starting Without Default Object Type Mappings 254

17.1. Introduction to Object Type Mapping

To visualize a scene element (entity, unit, detonation, and so on), MAK ONE applications must map its object type enumeration to an element definition. This chapter explains how to do that.

When you add simulation objects and other scene elements in the Simulation Object Editor, it creates model definitions and handles object type mapping. Although this chapter explains all of the details of the Simulation Object Type Mappings dialog box, using the Simulation Object Editor to add simulation objects is the preferred method.

You map object types to element definitions in the Simulation Object Type Mappings dialog box. It has pages for mapping:

- Cockpit displays
- Detonation effects
- Entities
- Firing effects
- Tactical graphics
- Tactical smoke
- Units

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

- You can map more than one element definition to an object type. If you do this, when entities of this type appear in a simulation, MAK ONE applications randomly assign one of the element definitions to the entity.
- You can map an element definition to more than one object type.

17.2. Work Flow Issues for Object Type Mappings

A default MAK ONE applications installation has multiple object type mapping files. When you start a MAK ONE application, it loads all of those mapping files. When you open the Simulation Object Type Mappings dialog box, all of the mappings are listed. When you create or edit object type mappings, MAK ONE applications do not automatically save them. Therefore, you are responsible for saving new and edited mappings.

The default object type mapping files are stored in *./SharedData/16/latest/ModelData/Configuration/Definitions/object_type*, where object_type is Detonation, Entity, Fire, Smoke, Tactical Graphics, or Unit. Cockpit type mappings are in *./appData/settings/stealth/default_CockpitTypeMap.mhtx*. Detonation,

fire, cockpits, and smoke have one default mappings file. The other object types each have four files (to accommodate VR-Forces, MAK's CGF application), which contain the following types of mappings:

- Aggregate level mappings. Mappings for simulation objects in VR-Forces aggregate-level scenarios.
- Entity level mappings. Mappings for simulation objects in VR-Forces entity-level scenarios.
- Base mappings. Mappings for simulation objects used by both aggregate-level and entity-level scenarios.
- Core mappings. Mappings for simulation objects for which VR-Forces does not have *.entity* files.

Before you add, delete, or edit a simulation object type mapping, consider which of the following work flows best meets your need:

- If you have created new model mappings, the preferred work flow is to export them to a new file. This is the easiest way to avoid overwriting other mappings and makes them easily portable to newer releases of MAK ONE applications.
- If you have edited existing mappings, you can save them to their original file or to a new file. If you export to a new file, make sure that the file name is alphabetically later in alphabetical order than the original mapping file. This ensures that when the new mapping is loaded it will overwrite the original one.

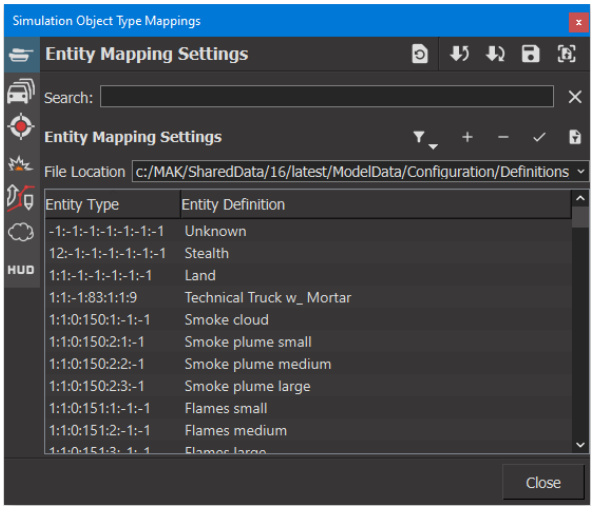
17.3. Adding an Object Type Mapping

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

➤ **To add an object type mapping:**

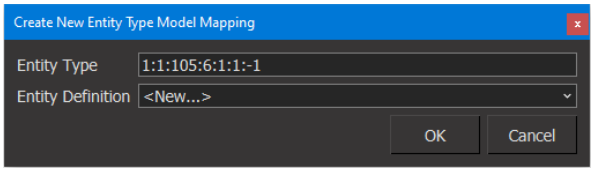
1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 91). The page shows a list of object type enumerations in the Entity Type column. The entity definition mapped to the object type is listed in the Entity Definition column.

Figure 91. Entity Mapping Settings page



3. Click the Add button (+). The Create New Entity Type Model Mapping dialog box opens (Figure 92). It lists a default object type enumeration.

Figure 92. Create New Entity Type Model Mapping dialog box



4. In the Entity Type text box, type the enumeration you want to map to a model. (Separate each value with a colon(:).)
5. In the Entity Definition list, select the entity definition that you want to map to this object type.
6. Click OK.
7. The new model mapping is added to the list.
8. Save the new model mapping to a new or existing file.

17.4. Editing an Object Type Mapping

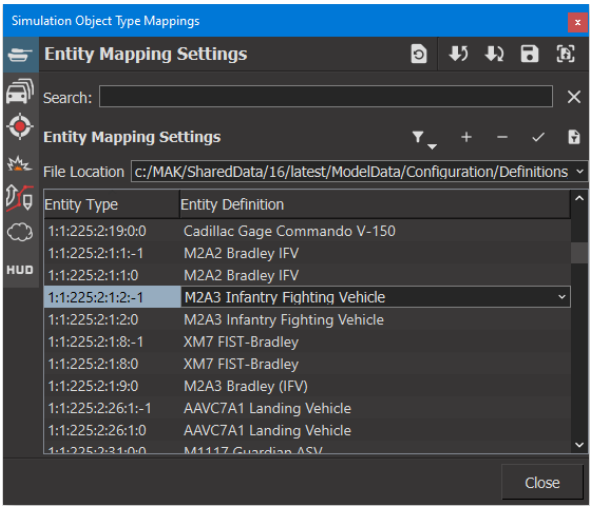
This procedure describes how to edit an object type mapping for a simulation object. The procedure is the same for any page on the Simulation Object Type Mappings dialog box.

➤ **To edit an object type mapping:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 91 above).

3. Optionally, filter the list by typing a text string in the Search box. All object types that include the text string in their name are listed.
4. Select the model mapping that you want to edit.
5. 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.
6. To change the element definition associated with an enumeration:
    a. Optionally, filter the list of available element definitions. (For details, please see "15.7. Filtering the List of Model Definitions" on page 223.)
    b. In the Entity Definition column, click the element definition to make it an editable list (Figure 93).
    c. Select the new element definition from the list.
    d. Click someplace else in the window.

Figure 93. Editable entity definition



7. Save the model mapping to a new or existing file.

17.5. Filtering the Element Definition List

The Simulation Object Types Mapping dialog box supports two types of filtering:

- You can filter the list of object type mappings based on the file from which they were loaded. This type of filtering is useful for entity, unit, and tactical graphics, which are loaded from multiple files. It is not meaningful for the other object types unless you have created new mappings and exported them to their own files.
- You can filter the list of element definitions that are available in the Entity Definition (Figure 93 above) list for mapping to entity types.

This type of filtering does not change the list of simulation objects in the Entity Mapping Settings window. It just changes the list of simulation objects in the Entity Definition list when you change a specific mapping. To change the list of objects in the window, type a text string in the Search box.

➤ **To filter the list of element definitions:**

1. On any page of the Simulation Object Type Mappings dialog box, click the Filter button (). A menu displays. A check mark indicates the type of element definition that will be displayed when you make a element definition entry editable. (You may have to expand the hierarchy to see which elements are checked.)
2. Select the element definition type that you want to see in the list or select All to see all element definitions.

➤ To filter the list of mappings by file location, on any page of the Simulation Object Type Mappings dialog box, click the Filter button ().

17.6. Changing an Object Type Mapping's File Location

The Simulation Object Type Mappings dialog box imports object type mappings from several files. It remembers the source file for each mapping and displays it in the File Location list. You can change the file associated with an object type mapping.

➤ **To change the file location of an object type mapping:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the type of simulation object whose mappings you want to edit.
3. Select the object type mapping whose file location you want to change.
4. Select a file in the File Location list.
5. Click the Change File Path button ().
6. Save the object type mappings.

17.7. Jumping to an Element Definition

You can quickly jump from an object type mapping to the associated element definition in the Visual Model Editors dialog box.

➤ **To jump from an object type mapping to the associated element definition:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the settings you want to examine.
3. Right-click the entity type.

4. On the context menu, select *object type* Definition. (For example, Entity Element Definition or Interaction Definition.) The Visual Model Editors dialog box opens to the selected definition.

17.8. Deleting an Object Type Mapping

The procedure for deleting an object type mapping is the same for any page on the Simulation Object Type Mappings dialog box.

➤ **To delete an object type mapping:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 91 on page 250).
3. Use the Import and Replace procedure to load the object type mapping file that you want to delete a mapping from. (For details, please see "17.10. Importing Object Type Mappings" on the next page.)
4. Optionally, filter the list by typing a text string in the Search box. All object types that include the text string in their name are listed.
5. Select the model mapping that you want to delete.
6. Click the Delete button ( ).
7. Save the model mappings.

17.9. Saving Object Type Mappings

New and changed simulation object type mappings do not get saved automatically. You must save them to an existing file or export them to a new file.

➤ **To save object type mappings:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page whose mappings you want to save.
3. Click the Save button ( ). All object type mappings get saved.

17.9.1 Exporting Selected Object Type Mappings

If you create new object type mappings, it is recommended that you export them to a new file. This will make it easy to migrate them to newer versions of the application. You can also export edited mappings to a new file. In this case make sure that you give them a name that is alphabetically later than the filename that they mapping was originally a part of so that it will override the original mapping.

➤ **To export selected object type mappings to a file:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the mapping you want to export.

3. Click the Export Selected button ( ). The Export Mapping Settings dialog box opens.
4. Specify the file to export to.
5. Click Save.

17.10. Importing Object Type Mappings

You can import simulation object type mapping files. An imported file replaces or merges with the previously loaded mappings.

> When you merge type mappings, mappings from the file being merged in overwrite matching mappings that have already been loaded. If you have multiple models mapped to an object enumeration, all of the mappings are replaced by the new mappings.

➤ **To import simulation object type mappings:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the type of simulation object whose mappings you want to import.
3. To replace the current set of mappings, click the Import and Replace button ( ). The Import Mapping Settings dialog box opens.

   To merge mappings, click the Import and Merge button ( ). The Merge Mapping Settings dialog box opens.

1. Select the settings file you want to import.
2. Click Open.

17.11. Starting Without Default Object Type Mappings

When you start a MAK ONE application, you can specify that it not load the entity, unit, or tactical graphics type mappings in *./SharedData/16/latest/ModelData/Configuration/Definitions* or *./appData/settings/<application>*. To do this, start the MAK ONE application with the `--noAppDataEntityTypeMap`, `--noAppDataEntityTypeMap`, or `--noAppDataEntityTypeMap` command-line option.



18. Wakes, Cockpits, 2D Icons, and Other Models

"Section III. Visual Models in MAK ONE Applications" on page 197 provides a general introduction to model schemas, model definitions, and element definitions. This chapter describes how to configure several specific types of visual models.

18.1. Configuring 2D Icons 257
18.1.1 Editing Font-Based 2D Icons 257
18.1.2 Using Images for 2D Icons 258
18.1.3 Adding Images for Simulation Object Icons 263
18.2. Adding New Cockpit Display Models 264
18.2.1 Installing a Cockpit DLL 265
18.2.2 Creating a Model Definition for a Cockpit 265
18.3. Adding Object Models as Props 268
18.3.1 Adding a Category to the Props Palette 268
18.4. Configuring Light Points 268
18.4.1 Configuring Light Point Type Models 270
18.4.2 Configuring Light Point Group Models 270
18.4.3 Adding a New Light Point Type 271
18.5. Configuring a Light Source for a Specific Sensor Mode 271
18.6. Configuring Wakes in the Visual Model Editor 271
18.6.1 Configuring Tidal Stream Wakes 274
18.6.2 Displaying Propeller Wash 274
18.7. Configuring Persistent Tracks in Model Definitions 274
18.8. Configuring Floating Lifeforms 276
18.9. Adding Wind-based Controls to Models 276
18.10. Using Texture Atlases to Skin Models 278
18.10.1 Configuring Texture Mapping in the Model Definition 278
18.10.2 Metafiles for Texture Mapping 279
18.11. Adding Dynamically Changeable Markings to Models 280
18.11.1 Adding @dis annotations to the Model 281
18.11.2 The Metafile for Markings 282

18.11.3 The Texture Mapping File 284
18.12. Flipping DDS Textures for a Model 284
18.13. Configuring SpeedTree Trees 285
18.13.1 Randomizing the Size of SpeedTrees 286
18.13.2 Displaying SpeedTrees in Planar Reflections 288
18.14. Best Practices for Creating Models 288
18.15. Disabling Indirect Rendering for an OpenFlight Model 289
18.16. Compressing Model Files 289

18.1. Configuring 2D Icons

The 2D icons in MAK ONE applications are configured in visual definitions. Font-based icons are configured in the Military Symbol Icon visual definition. Image-based icons use the Entity Image Symbol visual definition.

> You can quickly change the 2D icon for a simulation object in the VR-Forces Simulation Object Editor. For details, please see "Editing a Simulation Object's Visual Model" in the *VR-Forces Users Guide*.

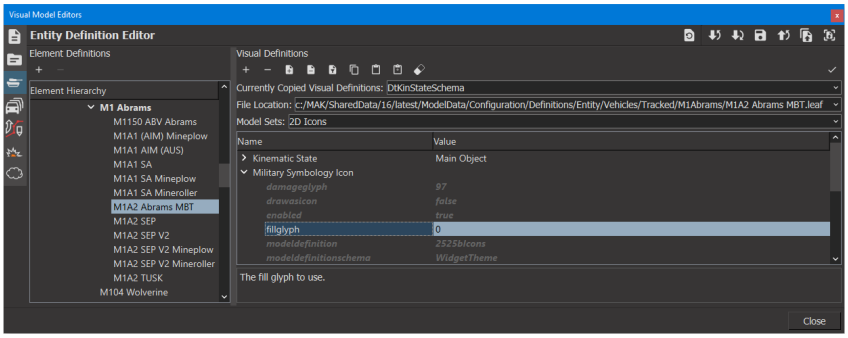
18.1.1 Editing Font-Based 2D Icons

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

➤ **To change a simulation object's icon:**

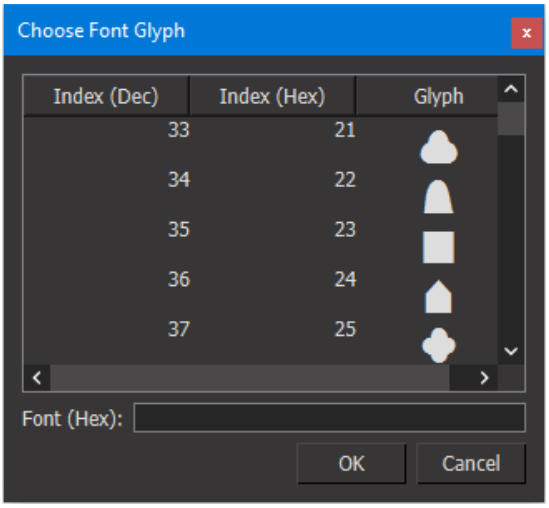
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 85 on page 233).
3. Select the simulation object whose icon you want to edit.
4. In the Model Sets list, select 2D Icons.
5. To change the fill glyph:
   a. Under the Military Symbology Icon visual definition, select the **fillglyph** attribute (Figure 94).

Figure 94. Military Symbol Icon visual definition



   b. Click the value in the Value column. The Choose Font Glyph dialog box opens (Figure 95).

Figure 95. Choose Font Glyph dialog box



c. Select the glyph with the fill pattern you want to use. Click OK.

6. To change the **outlineglyph** attribute, follow the same procedure as changing the fill glyph.

18.1.2 Using Images for 2D Icons

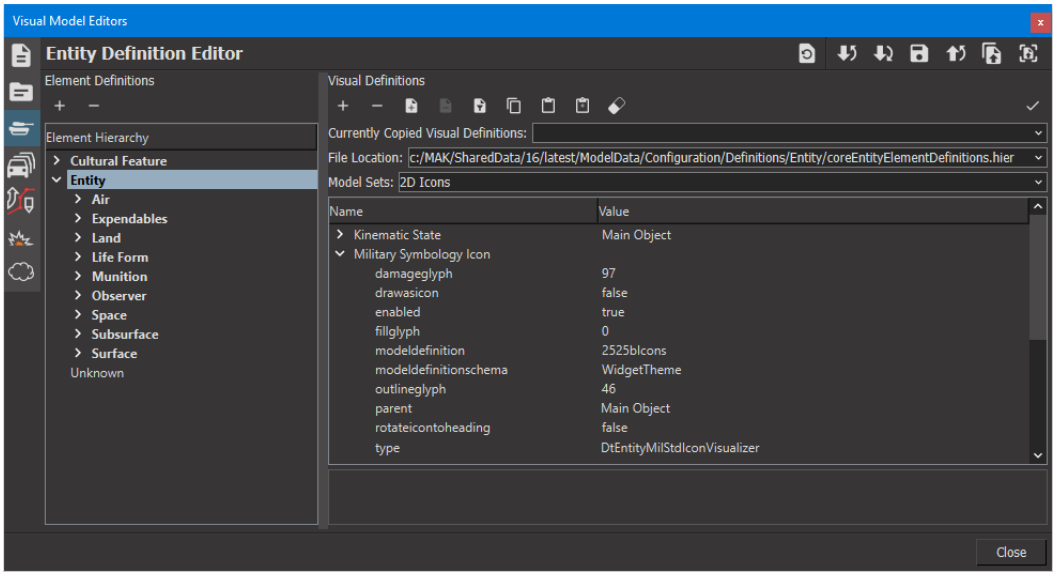
By default, MAK ONE applications use TrueType fonts for 2D icons. You can replace the font-based icons with images using the Entity Image Symbol visual definition.

> If you configure images for icons, be sure to save the element definitions to a file other than the default settings. If you revert to factory defaults or otherwise switch back to font-based icons and you have not saved the image mappings, you will not be able to recover them without repeating the entire mapping process again.

➤ **To use images for entity icons:**

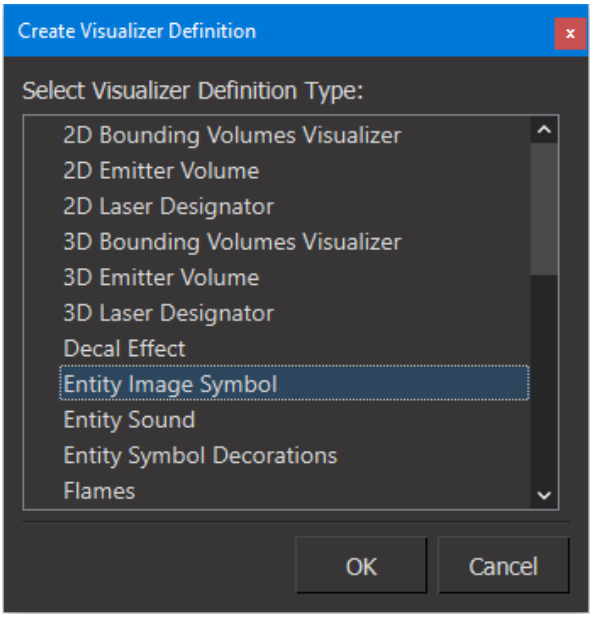
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 85 on page 233). (If you want to change the icon for a unit, select the Unit Definition Editor page. To change the icon for a waypoint, select the Tactical Graphics page.)
3. In the Model Sets list, select 2D Icons.
4. In the Element Definitions hierarchy, select Entity. (If you are editing unit images, in the Element Definitions hierarchy select Unit. If you are editing tactical graphics images, select the specific tactical graphic or hierarchy level that you want to edit.)
5. In the Visual Definitions list, select Military Symbology Icon (Figure 96). (Some tactical graphics do not have this visual definition.)

Figure 96. Entity element definition with Military Symbology Icon visual definition



6. Click the Delete button ( ). The visual definition is deleted.
7. Click the Add button ( ). The Create Visualizer Definition window opens. It displays a list of visual definitions (Figure 97).

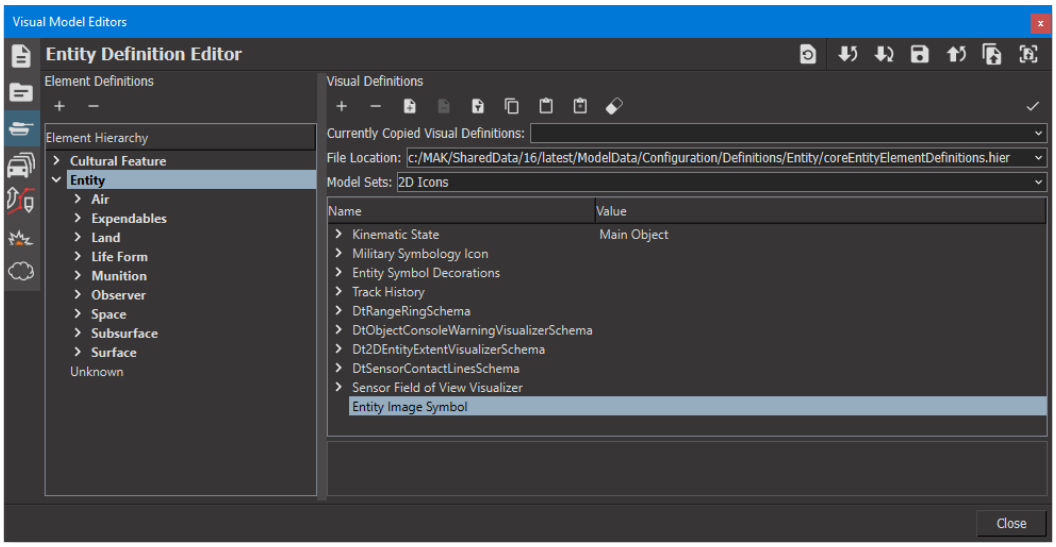
Figure 97. Create Visualizer Definition window



8. Select Entity Image Symbol. (If you are configuring images for units, select Aggregate Image Symbol; for tactical graphics, select Tactical Graphic Image Symbol.)
9. Click OK.

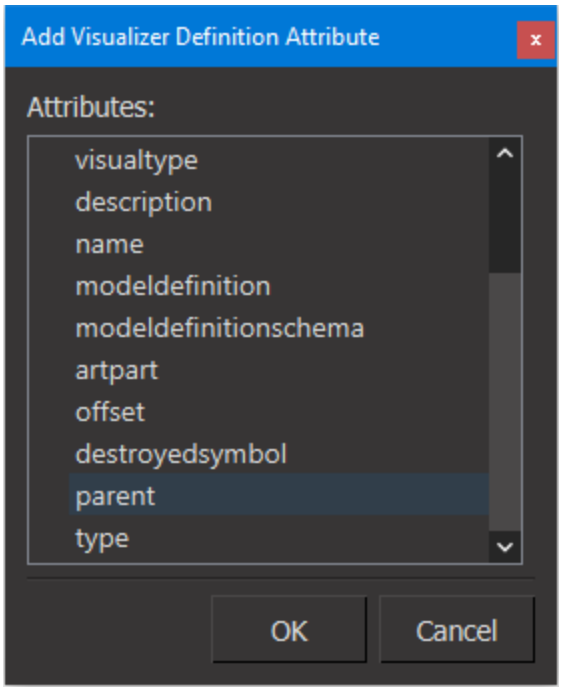
10. In the Visual Definitions list, select Entity Image Symbol (Figure 98).

Figure 98. Entity element definition with Entity Image Symbol visual definition



11. Click the Add Attribute button ( ). The Add Visualizer Definition Attribute window is displayed (Figure 99).

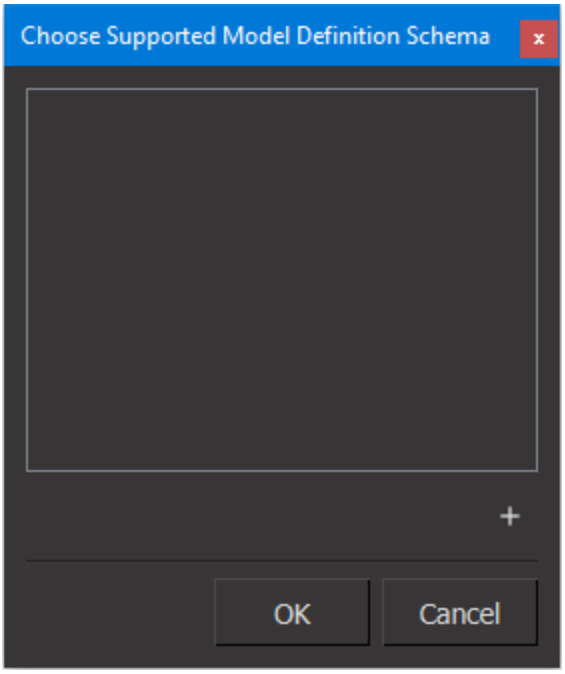
Figure 99. Add Visualizer Definition Attribute window



12. In the attribute list, select **parent**.
13. Click OK. The **parent** attribute is added to the visual definition.
14. Click the value field to make it editable.

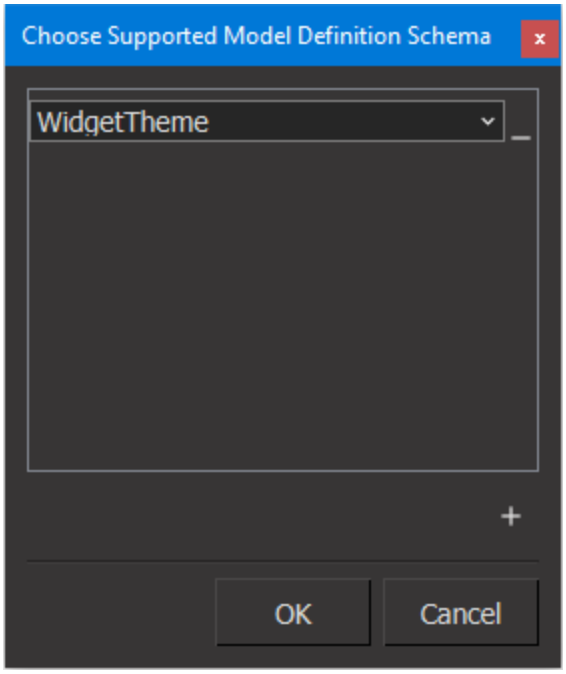
15. Type Main Model and then press **Enter**.
16. Click the Add Attribute button.
17. In the attribute list, select **type**and then click OK. The value populates automatically.
18. Click the Add Attribute button.
19. In the attribute list, select **modeldefinitionschema** and then click OK. The Choose Supported Model Definition Schema dialog box opens (Figure 100).

Figure 100. Choose Supported Model Definition Schema dialog box



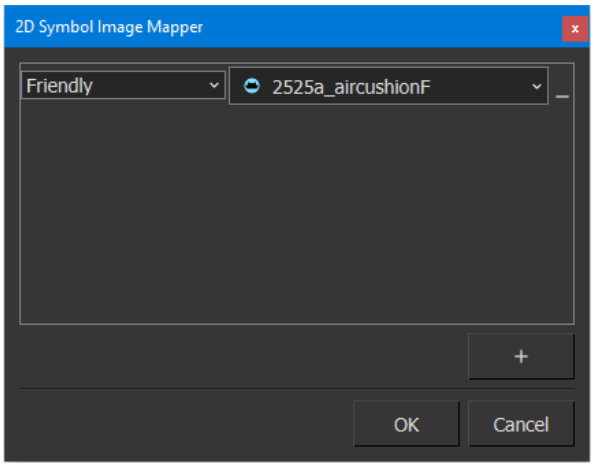
20. Click the Add button. A list is added to the dialog box.
21. In the list, select **WidgetTheme** (Figure 101).

Figure 101. Completed Choose Supported Model Definition Schema dialog box



22. Click OK.
23. Click the Add Attribute button.
24. In the attribute list, select **imagesymbolmap** and then click OK. The 2D Symbol Image Mapper dialog box opens.
25. Click the Add button. A list is added to the dialog box. It lists the forces for which you can map images and the images that you can apply to a particular force.
26. Select a force and an image from the lists (Figure 102).

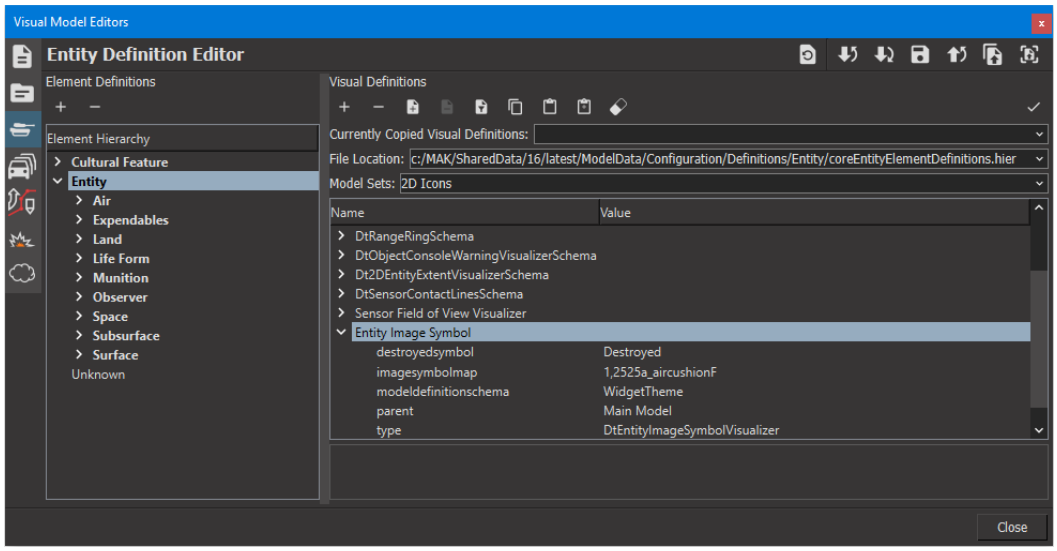
Figure 102. 2D Symbol Image Mapper dialog box with force chosen



27. Add an entry for each force that you want to represent with an image.

28. Click OK.
29. Click the Add Attribute button.
30. In the attribute list, select **destroyedsymbol** and then click OK. The **destroyedsymbol** attribute is added to the visual definition.
31. In the **destroyedsymbol** list, select Destroyed. The completed visual definition should look similar to Figure 103.

Figure 103. Entity Image Symbol visual definition



32. Export the updated setting.

The settings for the Entity Image Symbol visual definition are inherited by each entity in the hierarchy from the base Entity definition. Unless you configure the various entity categories, all entities use the images you configured for the Entity category. Therefore, at a minimum, change the **imagesymbolmap** values for each force at each generic object type, such as Fixed Wing and Life Form, to use the proper image for that object type. Optionally, you can specify an image for each entity definition at the leaf level.



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

18.1.3 Adding Images for Simulation Object Icons

MAK ONE applications include a representative set of MIL-STD 2525a images. They are in *./SharedData/16/latest/images*. If you want to add your own images, you can do so by adding a new model definition or changing the image mapping for an existing model definition. The new images are automatically added to the list of images for the **imagesymbolmap** in the Entity Image Symbol visual definition.

➤ **To add images for entities:**

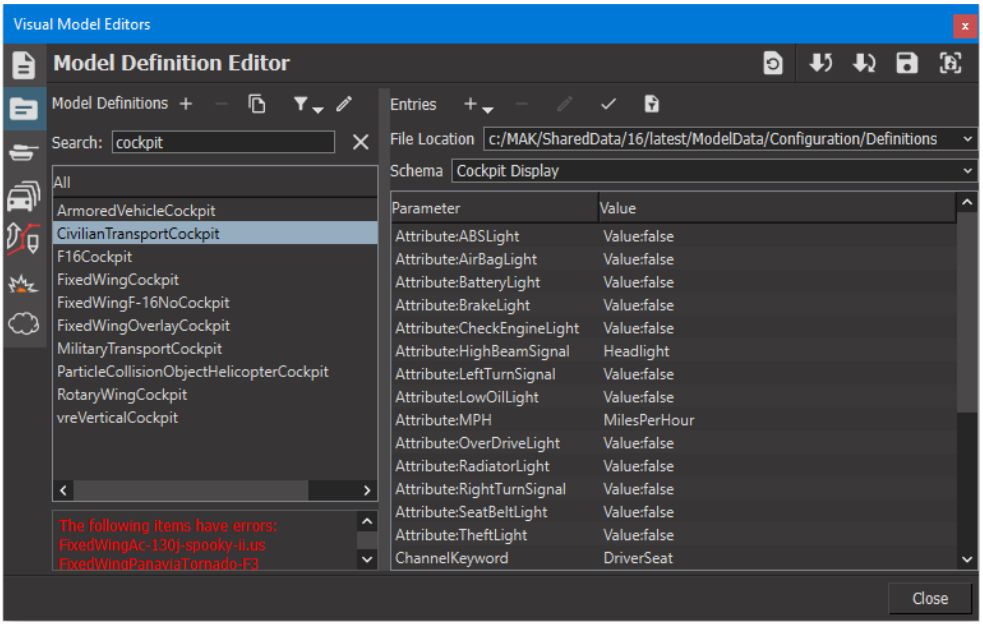
1. Copy the new images to *./SharedData/16/latest/images*.
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 80 on page 218).
4. In the Schema list, select WidgetTheme.
5. If you want to change the image used for an existing model definition:
   a. Select the model definition that you want to change, for example, 2525a_groundF.
   b. Change the file specified for the **backgroundImage** attribute.

   If you want to specify an image for a new model definition:
   a. Create a new model definition, as described in "15.1. Creating and Editing Model Definitions" on page 218. It is recommended that you copy an existing model definition for a 2525a symbol.
   b. Specify the image that you want to use for the new model definition.

18.2. Adding New Cockpit Display Models

MAK ONE applications support 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 104 shows the model definition for a cockpit.

Figure 104. Cockpit model definition



Cockpit updaters are associated with an attribute name that is known to the RSO. The value you provide for a cockpit-specific attribute is the name of an updater for dynamically changing values or a constant value. Constant values are commonly used

to configure a cockpit when it is loaded. MAK ONE applications provide 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 MAK ONE applications only covers the cockpits delivered with the product. If you add your own cockpits, you require GL Studio run-time licenses.

The general procedure for adding a new cockpit model is:

1. Install the cockpit's DLL.
2. Create a model definition for the cockpit.
3. Map object types to the model definition, as described in "17.3. Adding an Object Type Mapping" on page 249.

18.2.1 Installing a Cockpit DLL

If you create your own cockpit, it is implemented as an RSO DLL that you must add to the MAK ONE application's file structure.

➤ To install a new cockpit, place the DLL for the cockpit in *./SharedData/16/latest/Huds/platform* (where platform is the appropriate folder) and then create a model definition for it.

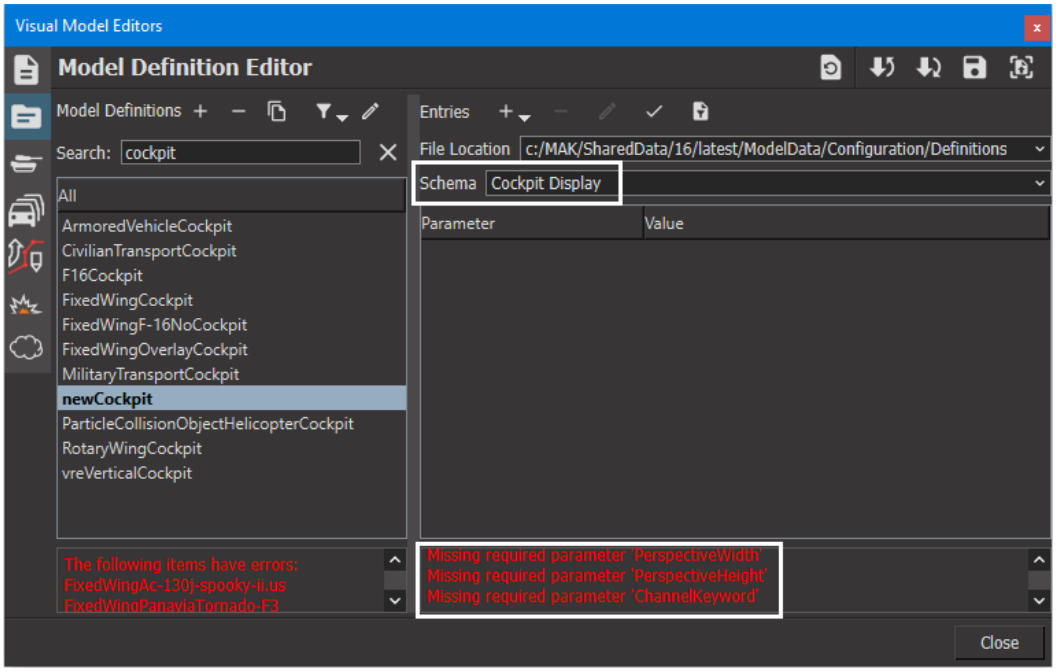
18.2.2 Creating a Model Definition for a Cockpit

Cockpit models are configured using schemas and model definitions, just like any other model in a MAK ONE application. 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 "15.4. Creating a Model Definition" on page 221. Be sure to select Cockpit Display as the schema. A new model definition is added to the display (Figure 105). The message window lists the parameters that you must add. These are the parameters defined by the schema.

Figure 105. New cockpit model definition



2. Add the required parameters, as described in "15.8.1 Adding Parameters to a Model Definition" on page 224. Table 21 describes the required parameters.

Table 21. Required parameters for cockpit models

Parameter	Description
Filename	The name of the DLL without the file extension.
RSOName	The name of the class to be loaded from the DLL. Default: *filename*Class.
Perspective	2D or 3D. This parameter refers to the perspective of the cockpit display, not the terrain.
PerspectiveWidth **PerspectiveHeight**	Adjusts the orthogonal projection for a 2D cockpit. Adjusts the perspective projection for a true 3D cockpit.
ChannelKeyword	A keyword that tells a channel to display any object that has the keyword in its definition. Default: showCockpits. (This keyword means that any channel that has the showCockpits keyword will show cockpits.)
OffsetX **OffsetY** **OffsetZ**	Moves the cockpit offset around the screen. The OffsetX value shifts the cockpit left or right. The OffsetY value shifts it up or down. OffsetZ shifts a 3D cockpit forward and back. It has no effect on 2D cockpits.

3. Add the parameters required by the cockpit RSO. To do this, add an Attribute parameter, as follows:
    a. Add an Attribute parameter, as described in "15.8.1 Adding Parameters to a Model Definition" on page 224.
    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 MAK ONE applications, as described in Table 22.
4. Map an object type to your definition, as described in "17.3. Adding an Object Type Mapping" on page 249.

Table 22. Updaters provided

Updater	Description
AOA	Angle of Attack. Returns the pitch of the entity clamped (5 to 30 degrees).
FeetAGL	Feet Above Ground Level. Returns the feet above the terrain (or buildings below).
FeetMSL	Feet Above Mean Sea Level. Returns the altitude above the sea level.
FeetPerMinute	Feet Per Minute. Returns the climb rate.
KilometersPerHour	Speed of the entity in kilometers per hour.
MilesPerHour	Speed of the entity in miles per hour.
KnotsPerHour	Speed of the entity in nautical miles per hour.
Pitch	Returns the pitch of the entity clamped (-180 to 180 degrees).
Roll	Returns the roll of the entity clamped (-180 to 180 degrees).
Yaw	Returns the yaw of the entity clamped (0 to 360 degrees).
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.

18.3. Adding Object Models as Props



Props are terrain objects that you can select independently of the terrain. "12.2. Extracting Props from a Terrain" on page 191 explains how to create them by extracting them from a terrain or a feature layer. You can also add props by creating model definitions for them. In VR-Forces, these props are added to the Props Palette and are then available to add to a terrain.

The steps involved in adding an object model as a prop are:

1. Add a new model definition, as described in "15.1. Creating and Editing Model Definitions" on page 218. Use the Entity, Tree, or SpeedTree schema, as appropriate. When you name the model definition, if you preface it with one of the categories on the Props Palette, such as Buildings, it automatically is placed in that category on the palette. For example, BuildingsMyBuilding puts MyBuilding in the Buildings category.
2. Add the filename parameter.
3. Set the value for the filename to the location of your object model file. You can select any supported format. (Some formats, such as Collada (DAE) are not included in the file filter. To select such a format, choose All Files in the filters list.)

18.3.1 Adding a Category to the Props Palette



The list of categories on the Props Palette is controlled by the configuration file *./appData/settings/vrfGui/VRF3DTerrainPropCategories.xml*. To add a new category, edit the file. The format should be obvious from examining the existing XML elements.



If you edit *VRF3DTerrainPropCategories.xml*, be sure to update the <count> element to reflect additions or deletions.

18.4. Configuring Light Points

MAK ONE applications support configuration of light points using light point types. A light point type is an optional configuration for a light point that allows consistent behavior across models and terrains.

Light point type settings are configured in *./appData/importConfig/Lights_DefaultValues_3_2.csv*. You can edit this file and add additional light point types. Table 23 describes the parameters specified in *Lights_DefaultValues_3_2.csv*.

Table 23. Light point type settings

Parameter	Descriptions
Code	Integer that specifies the light point type for this row.
Description	String that specifies a human readable description for the light point type. It is currently unused in MAK ONE applications.

Table 23. Light point type settings (continued)

Parameter	Descriptions
Intensity	Floating point number that specifies the intensity of the light point. Usually (0-1.0) but does support values higher than 1.0.
Red	Floating point number that specifies the red color value for the light point. Usually (0-1.0), but supports values higher than 1.0.
Green	Floating point number that specifies the green color value for the light point. Usually (0-1.0), but supports values higher than 1.0.
Blue	Floating point number that specifies the blue color value for the light point. Usually (0-1.0), but supports values higher than 1.0.
Directionality	String that specifies the directionality of the light point. Valid strings are: • **Omni-directional** - Light point has the same intensity when viewed from any direction. • **Directional** - Light point has the regularly specified intensity when viewed from the within the lobe and the residual intensity when viewed outside the lobe. • **Bidirectional** - Light point has the regularly specified intensity when viewed from the lobe or another lobe in the opposite direction and the residual intensity when viewed outside the two lobes.
Horizontal Lobe	Floating point value in degrees for the angle that describes the horizontal lobe away from the light point's direction. Only used when **Directionality** is set to **Directional** or **Bidirectional**. (0.0 - 360.0)
Vertical Lobe	Floating point value in degrees for the angle that describes the vertical lobe away from the light point's direction. Only used when **Directionality** is set to **Directional** or **Bidirectional**. (0.0 - 360.0)
Residual Intensity	Floating point value that specifies the intensity of the light point when the light point is viewed from a direction that is outside the lobe. Only used when **Directionality** is set to **Directional** or **Bidirectional**. Usually (0.0 - 1.0) but does support values greater than 1.0.
Frequency	Floating point value that specifies the number of times per second that the light point should blink. 0.0 turns off blinking.
Duty Cycle	Floating point value that specifies the fraction of time that the light point should be on. (0.0 - 1.0)
Min Pixel Size	Positive floating point value that specifies the minimum number of pixels for the light point's diameter.
Max Pixel Size	Positive floating point value that specifies the maximum number of pixels for the light point's diameter.

Table 23. Light point type settings (continued)

Parameter	Descriptions
Max Visible Distance	Positive floating point value that specifies the maximum distance in meters that the light point should be visible. This only has an effect if the light point is bright enough to be visible from this distance.
Radius	Positive floating point value that specifies the size of the light point in meters.
Daylight Controlled	Specifies whether or not the light point should be visible during the day. If true, then the light point turns off during the day.
Bulb Type	Specifies the type of bulb that the light point is using. It is currently only used by SensorFX and matches the entries in the *mm_lsources.dat* table. Please see SensorFX documentation for a description of how to configure light point bulb types in SensorFX.

18.4.1 Configuring Light Point Type Models

You can configure light point types for a particular light point in the OpenFlight model using Creator in a couple of different ways.

The simplest way is to use the light point palette manager in Creator to specify the Appearance name for a particular light point appearance index. The name can either be a CDB type name, which can be set using the **Edit** > **CDB Light Point Palette Names** menu or it can be an integer value that is one of the codes specified in *Lights_DefaultValues_3_2.csv*.

The CDB names can also be found in *./appData/importConfig/Lights_DefaultValues_3_2.csv* (in the fourth field, which is labeled Hierarchy). For example, the light point type with code 1 would have the CDB name of \Light\Platform.

18.4.2 Configuring Light Point Group Models

You can group light points into light point groups, which let you control the color of a group of light points. These groups can group light points for things like runways, airports, parks, and cities.

Light point groups are automatically generated for the procedural airports and the CDB database airports using the runway and airport names.

A terrain that has an OpenFlight airport generated by the Creator Airport or Runway wizards will also automatically have light point groups if the airport name follows the format “Airport - *{name here}*” where *{name here}* specifies the airport name.

The default grouping name style for airports and runways is **{Airport Name}:{Runway Name}:***{optional type}* where *{optional type}* is Approach or Runway.

Another way to set the light point group is to use comments on the light point group node. If the comment contains **@dis light_group** *{group name}* where *{group name}* is the name for this light point group, then all light points under that particular light point group node will be in that light point group.

You can set a light point group color using the VR-Vantage view control messages, CIGI component control messages, or the developers toolkit.

18.4.3 Adding a New Light Point Type

You can add new light point types to the default configuration files. To add a new light point type, you add an entry to the end of *./appData/importConfig/Lights_DefaultValues_3_2.csv* and to *./appData/importConfig/Lights.xml*. You can add it anywhere in the *Lights.xml* hierarchy, but it is recommended that you add it to the end of the **Lights** element.

The following example shows how to add light point type 488 to the configuration.

1. Add the following to the bottom of *Light_DefaultValues_3_2.csv*:

   ```
   488, Red
   Custom,0.6,1,0,0,Omnidirectional,57.2958,57.2958,0,0,0.5,1,20,
   100000,0.2
   ```

2. Change the bottom of the *Lights.xml* from:

   ```
       </Light>
   </Lights>
   ```

   To:

   ```
       </Light>
       <Light type="Red_Custom" code="488">
           <Description>Red Custom Light Type</Description>
       </Light>
   </Lights>
   ```

3. Create a model that uses the new custom light point type.

18.5. Configuring a Light Source for a Specific Sensor Mode

Some light sources are only visible when viewed using infrared or night vision goggles. You can configure sensor mode specific light sources by adding a comment on the light source node. The comment must be:

```
@dis lightsource SENSOR
```

where SENSOR is `IR`, `MWIR`, `LWIR`, or `NVG`.

18.6. Configuring Wakes in the Visual Model Editor

When dynamic ocean is enabled, MAK ONE applications simulate wakes on surface entities. Wakes are configured in the Ship Wake visual definition of an entity's element definition. If you add your own model, you may want to change the default configuration values for its wake so that it more closely matches the dimensions and velocity of the object type being modeled.

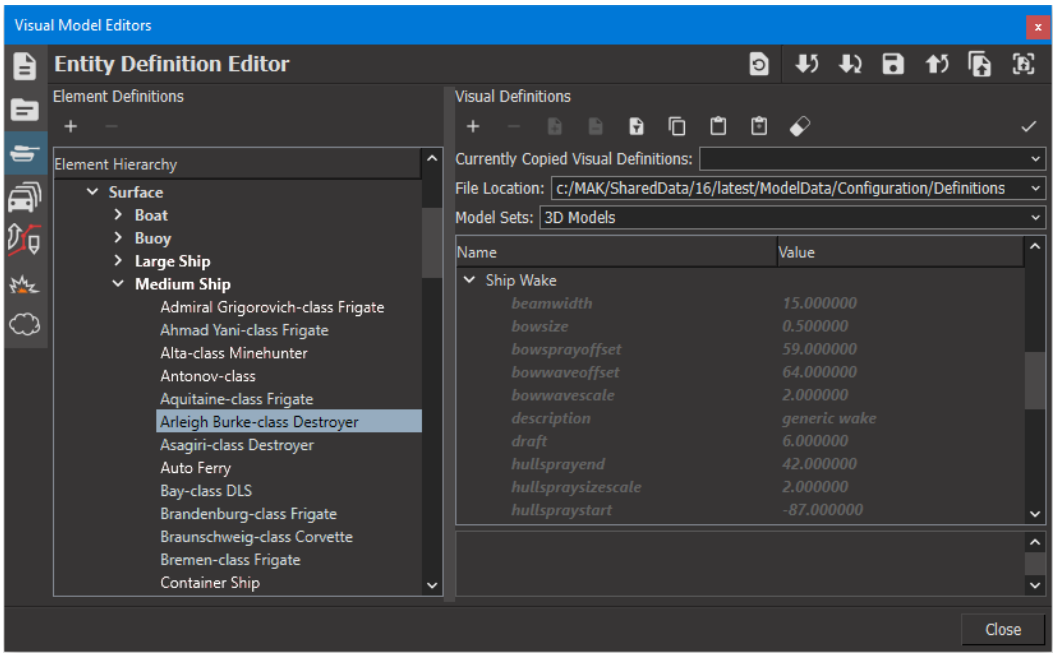


You can also configure wakes in the Simulation Object Editor. For details, please see "Configuring Wakes" in the simulation models section of the *VR-Forces Users Guide*.

➤ **To configure a ship wake in the Visual Model Editors dialog box:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 106).

Figure 106. Ship Wake visual definition



3. In the Element Hierarchy, expand the Entity list to the Surface level and select the entity that you want to configure.
4. In the Visual Definitions list, select and expand the Ship Wake visual definition.
5. Edit the parameters. Table 24 describes each parameter.

Table 24. Ship Wake visual definition attributes

Parameter	Description
beamwidth	The width of the ship at its widest point.
bowsize	The size of the bow in world units. This affects the wavelength of the bow wave and the initial spread of spray particles at the bow. Use 0 for a pointy bow.
bowsprayoffset	An offset from the ship position from which to emit spray particles.
bowwaveoffset	An offset from the ship position, along the direction of travel, at which bow waves will originate.
bowwavescale	A scaling factor for the bow wave's amplitude at the bow of the ship.

Table 24. Ship Wake visual definition attributes (continued)

Parameter	Description
draft	The depth of the hull underwater.
hullsprayend	The offset from the ship at which hull spray effects end.
hullspraysizescale	A scaling factor applied to the hull spray particles.
hullspraystart	The offset from the ship at which hull spray effects begin.
hullsprayvelocityscale	The initial velocity of hull spray particles as a percent of the ship velocity (0-1).
hullsprayverticaloffset	A vertical offset to the starting point of new hull spray particles.
loddistance	The distance from the camera beyond which wakes are not drawn. Cannot be greater than the value set by **max-kelvin-wake-distance** in the MAK Overrides section of *./data/Environment/Ocean/Triton.config*.
maxbowwave	The maximum amplitude of the bow wave, or -1.0 for unbounded.
numhullsprays	The number of spray particles emitted periodically along the hull of the ship.
parent	The visualizer's parent.
propwashenabled	Enable or disable propeller backwash effects.
propwashfadetime	Number of seconds during which the propeller backwash effects will dissipate.
propwashoffset	An offset from the ship position from which to generate propeller backwash effects.
shiplength	A value that, in combination with the entity's speed, is used to determine the length of the bow wake.
sprayenabled	If enabled, the wake emits spray particles.
spraysizescale	A scaling factor applied to the size of the bow spray particles and their random spread.
sprayvelocityscale	A scaling factor for spray effects at the bow of the ship. This is applied to the initial velocity of the spray particles.
sternwaveenabled	Enable or disable the stern wake.
sternwaveoffset	An offset for the stern wakes.
type	The type of visualizer.
wakeoffset	An offset to the model's origin that all other offsets are relative to. Changing this offset lets you move all of the wake effects in tandem if the entity model changes.

18.6.1 Configuring Tidal Stream Wakes

Tidal stream wakes are supported as standalone models and are automatically attached to streamed buoys and beacons. Model definitions for wakes to be used with each of the supported buoy and beacon models are included (BuoysPillarWake, BeaconsLatticeWake, and so on). You may want to edit them. We recommend wake lengths of at least 20m for best results. (Some of the model definitions specify values less than that.)

By default, MAK ONE applications start with a tidal stream current speed of 0. To see tidal stream wakes, change the speed on the Scene Settings dialog box, Environment Conditions Settings page. For more information, see:

- "Configuring Marine Conditions" in the *VR-Vantage Users Guide*
- "Configuring Marine Conditions" in the *VR-Forces Users Guide*

For VR-Vantage, you can also change the speed and direction of the tidal stream current using a CIGI Component Control or Short Component Control message. The message uses a component class of Regional Sea Surface(4), a component ID of 1, and two float component data fields — the first is the tidal stream current speed (m/s), and the second is the current direction (0-2PI radians, clockwise from North). The Region ID is ignored.

18.6.2 Displaying Propeller Wash

If propeller wash causes many wakes in a scene, they may cut off or disappear. This occurs when the number of propeller wash segments exceeds those specified in the *./data/Environment/Ocean/Triton.config* file. You can increase the number to ensure that the propeller wash displays realistically.

➤ **To view more propeller wash segments:**

1. Open the *Triton.config* file.
2. Edit the **max-prop-wash-opengl** value, increasing the number. The default is **50**, which you may need to change to **100** or another number.
3. Save the file.

18.7. Configuring Persistent Tracks in Model Definitions

On dynamic terrains, VR-Forces ground vehicles can create tracks that do not fade out. There are three parameters for the Decal schema model definitions (Figure 107) that allow you to link a tracks model definition to a persistent tracks message:

- PersistentTrackType. Type for this model definition to match with. Required to show persistent tracks with this model definition. It can be one of five strings: single-wheeled, wheeled, double-wheeled, tracked, or mixed.
- PersistentTrackSoilType. Optional parameter to specify the soil type for this model definition to match with. See "Entity Movement and Soil Type" in the *VR-Forces Users Guide* to learn what soil types are sent.

- PersistentTrackDepth. Optional parameter to specify the track depth type for this model definition to match with. It can be one of four strings: none, light, medium, or heavy.

Entity definitions allow you to specify decal effects as needed (Figure 108).

For information on model and element definitions, see "Chapter 15. Model Definitions" on page 217 and "Chapter 16. Element Definitions" on page 229.

Figure 107. Persistent track parameters in Decal Schema

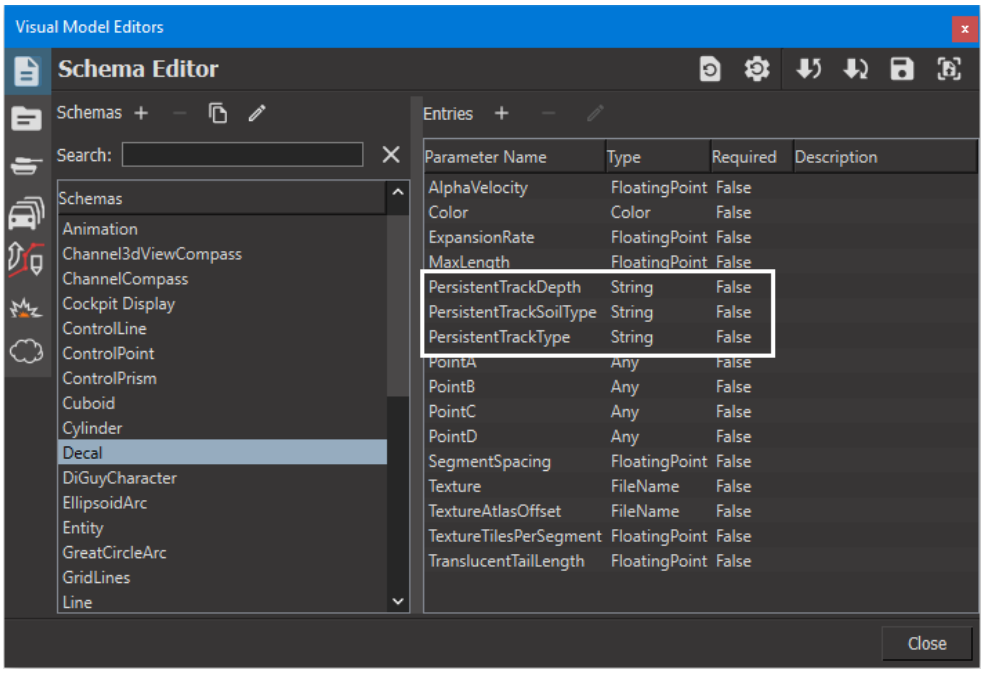
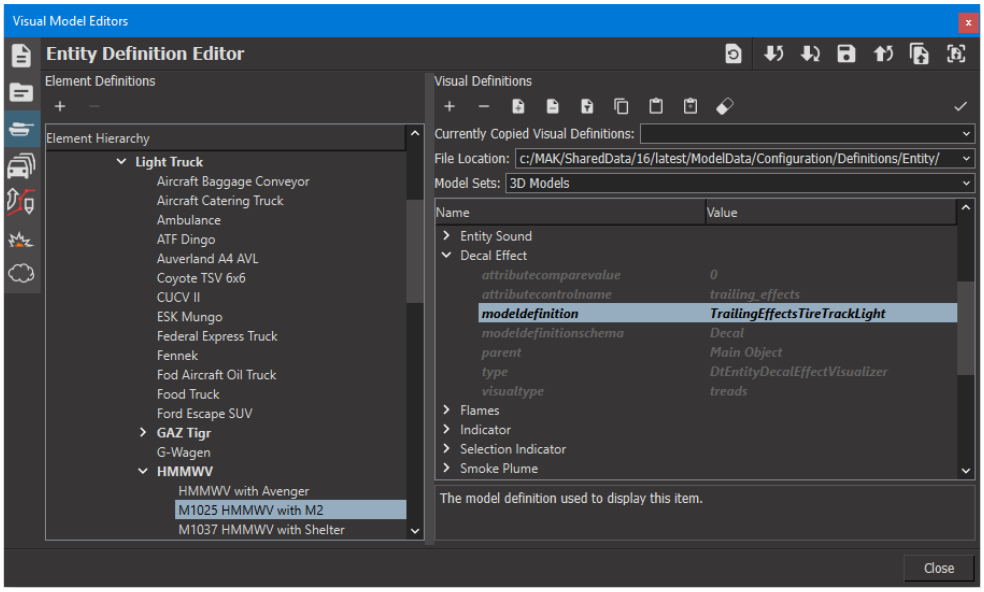


Figure 108. Decal Effect definition in Entity Definition Editor



18.8. Configuring Floating Lifeforms

Some entities have swimming animations. By default, most lifeforms do not float.

➤ **To configure a lifeform to float:**

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, please see "15.3. Model Definition Files" on page 220.)
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 80 on page 218).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "15.10. Importing Model Definitions" on page 226.)
5. Optionally, filter the model definitions list by typing a text string in the Search box or by selecting a schema. (For details about filtering by selected a schema, please see "15.7. Filtering the List of Model Definitions" on page 223.)
6. Select the model definition for which you want to add a parameter.
7. Add the **waterlineOffset** parameter to the model definition. -1.55 is a good value for an adult male.
8. Optionally, add the **buoyancyLODDistance** parameter. It determines the distance at which a floating entity is no longer visible.
9. Save the updated model definition.
10. If you want to export the edited element definition to an existing leaf file, use the Load and Replace procedure to reload the hierarchy. (This deletes all element definitions.) Then load the element definition leaf file. (For details, please see "16.11. Importing Element Definitions" on page 243.)
11. Select Entity Definition Editor page (Figure 85 on page 233).
12. Select the element definition that you want to edit.
13. Expand the Kinematic State definition.
14. If the **isamphibious** attribute is not present, add it.
15. Set **isamphibious** to **True.**
16. If the **possiblegroundclampingmode** attribute is not present, add it.
17. Set **possiblegroundclampingmode** to **ClampWater**.

18.9. Adding Wind-based Controls to Models

MAK ONE applications can simulate the movement of flags and windsocksflags, windsocks, and fans based on the current simulated wind speed and direction on terrains. It can also simulate movement of flags and windsocks on models added through connections or programmatically (this is not supported for props or external references). These features require models with appropriate switches and articulations, as well as model-specific or terrain-specific metafiles.

For generic information about the metafile format and syntax, please see section 8.6, Transforming Nodes Programmatically, in the *VR-Vantage Developers Guide* or *VR-Forces Front-End Developers Guide*.

The metafiles for windsocks and flags have sections that configure wind control of nodes, as follows:

- Control switch nodes based on wind speed.
- Control the heading (Z-Axis) of articulating (degree of freedom) nodes based on wind direction.
- Control the pitch (X-Axis) rotation rate (for wind turbines) of articulating nodes based on wind speed.

The metafile used for wind features supports three features — WindSwitchPart and WindDirPart, and WindRotPart.

- WindSwitchPart. The WindSwitchPart feature changes the state of a switch node based on the current wind speed. The flag models used by MAK include a switch with three states. This feature has the following parameters:
  - **WindSpeed**. A list of wind speeds (in km/h).
  - **WindState**. A list of the switch state values that correspond to the values in the **WindSpeed** parameter.

  The number of items in each list should be the same and only contain switch states supported by the node specified in the **Linkage** entry.
- WindDirPart. The WindDirPart feature modifies the heading value (Z-Axis) of an articulation (degree of freedom) node based on wind direction. It has the following parameters:
  - **RotOffsetDeg**. Allows a rotation offset to be applied.
  - **RotDir**. Indicates if the node should be rotated in a clockwise or counterclockwise direction.
- WindRotPart - The WindRotPart feature is used to modify the pitch value (X-Axis) of an articulation (degree of freedom) node based on wind speed. It has the following parameters:
  - DelayBeforeRot. Specifies a wind speed below which no rotation occurs.
  - DegPerKPH. Specifies the addition rate of rotation (in degrees per second) to increase the rotation rate for each kph of wind speed above the **DelayBeforeRot** value.

Some of the metafiles used to configure flags, windsocks, and wind turbines include *./SharedData/16 /latest/TerrainData/Terrain/Hawaii/Geometry/OpenFlightAlaMonana/AlaMoanaGeocentric.meta*, *./SharedData/16/latest/ModelData/Structures/Poles/FlagPole/FlagPoleFlag.meta*, *./SharedData/16/latest/ModelData/Structures/Poles/WindSock/WindSock.meta*, and *./SharedData/ 16/latest/ModelData/Structures/Towers/WindTurbine3Blades/model.meta*.

18.10. Using Texture Atlases to Skin Models

When a MAK ONE application renders a vehicle model, it takes into account information about the geometry of the vehicle, which specifies the dimensions, shape and so on, and textures, which specify the surface characteristics (in other words, how the vehicle looks.) If several models have the exact same geometry, but only differ in how they look (their textures), you can save disk space and some processing time by using the same geometry file for all of the models and a texture atlas to specify the textures to use for each model. (A texture atlas is a composite from multiple textures with the corresponding 3D models having uv coordinates shifted.)

For example, MAK ONE applications have four different versions of the Airbus 320 jet, one each for American Airlines, British Airways, Lufthansa, and Singapore Airlines. Instead of having four geometry files with associated textures, you can use one geometry file for all four models and reference the appropriate texture in a texture atlas.

Texture skinning requires a MEDF (geometry) file and a texture atlas that has been prepared for the entities that will share the texture atlas. The portions of the texture for a given model are accessed via UV offsets into the texture atlas. This information is stored in a metafile (*.meta*) that references a *.dtxc* file. You must also specify the metafile in the model definition for each model that is to use it and a **configName** parameter that is used to identify the correct texture to use.

18.10.1 Configuring Texture Mapping in the Model Definition

To use a texture atlas for a model, you must provide values for the **metaFileName** and **configName** parameters in the model definition.

➤ **To configure texture mapping in a model definition:**

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, see "15.2. Work Flow Issues for Model Definitions" on page 219.)
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 80 on page 218).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "15.10. Importing Model Definitions" on page 226.)
5. Optionally, filter the model definitions in the list. (For details, please see "15.7. Filtering the List of Model Definitions" on page 223.)
6. Select the model definition that you want to edit.
7. Click the Add Parameter button (). A list of parameters is displayed.
8. Select metaFileName. The **metaFileName** parameter is added to the model definition.
9. In the value column, specify the location of the metafile you want to use for this model.

10. Click the Add Parameter button and select configName. The **configName** parameter is added to the model definition.
11. In the value column, type a string that should look for in the metafile to identify the texture pattern to use.
12. Export the model definitions.

18.10.2 Metafiles for Texture Mapping

MAK ONE applications use metafiles to configure texture skinning. For conceptual information about metafile format and syntax, please see section 8.6, Transforming Nodes Programmatically, in the *VR-Vantage Developers Guide* or *VR-Forces Front-End Developers Guide*.

A metafile is associated with a terrain or model in one of the following ways:

- Place it in the same directory as the terrain or model's MEDF file. The metafile must have the same name as the MEDF file, but with a *.meta* extension.
- Specify it in the **metaFileName** parameter of the model definition used to import the model.

The metafiles support two features that are relevant to texture mapping:

- UvAtlasMapper. Defines a texture atlas for texture skinning.
- SwitchStateMapper. Specifies the child of a switch node to use. For example, you could have an M1A1 with mine rollers, with mine plows, or DU armor, depending on the switch node used.

The following example shows the metafile for the Airbus 320 jet *Airbus320a.meta*:

```
{
  "NodeFeatures": [
    {
      "Linkage": {
        "Comment": "Traversal with match up this feature based on
NodeName match OR AnnotationName match",
        "NodeName": "",
        "AnnotationName": "@dis annotation texture"
      },
      "FeatureData": [
        {
          "FeatureName": "UvAtlasMapper",
          "AtlasFile": "$(SHARED_DATA_
DIR)\\ModelData\\Vehicles\\FixedWing\\Airbus320a\\Airbus320a.dtxc",
          "Mappings": [
            {
              "ConfigName": "AmericanAirlines",
              "AtlasKeys": {
                "Pattern": "AmericanAirlines"
              }
            },
            {
```

```
            "ConfigName": "BritishAirways",
            "AtlasKeys": {
              "Pattern": "BritishAirways"
            }
          },
          {
            "ConfigName": "Lufthansa",
            "AtlasKeys": {
              "Pattern": "Lufthansa"
            }
          },
          {
            "ConfigName": "SingaporeAirlines",
            "AtlasKeys": {
              "Pattern": "SingaporeAirlines"
            }
          }
        ]
      }
    ]
  }
]
}
```

When a MAK ONE application loads an Airbus 320, the **metaFileName** parameter in the model definition points it to the *Airbust320a.meta* file. The **configName** parameter in the model definition specifies which texture to use, for example, AmericanAirlines.

The MAK ONE application reads *Airbust320a.meta* and associates the **configName** string, with the pattern value, also AmericanAirlines. Then it looks in *Airbus320a.dtxc* and gets the UV offset of AmericanAirlines so it knows which textures to use from the texture atlas.

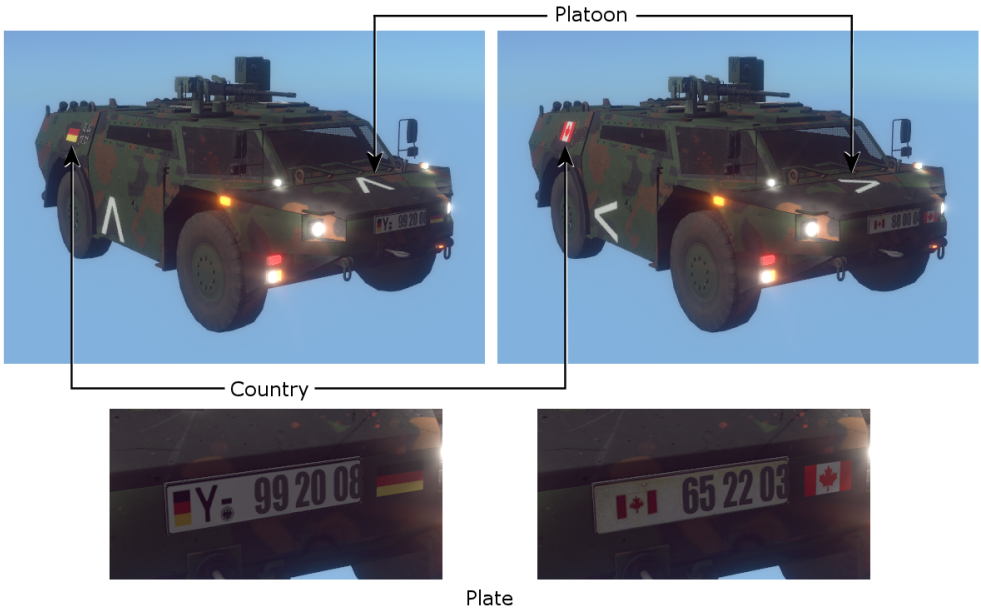
The UV offsets for these models are configured in *Airbus320a.dtxc*. (The file contains explanatory comments.)

```
texture,COLPAT,AmericanAirlines,0.0,0.0
texture,COLPAT,Lufthansa,0.0,0.189612
texture,COLPAT,SingaporeAirlines,0.0,0.344823
texture,COLPAT,BritishAirways,0.0,0.500039
```

18.11. Adding Dynamically Changeable Markings to Models

Similarly to the way that you can use a texture atlas to configure texture skinning, you can use texture atlases to create dynamically changeable markings, such as country insignia and license plate numbers, on vehicle models. MAK ONE applications include a model, the Fennek Armored Car, that has dynamically changeable markings (Figure 109).

Figure 109. Fennek armored car with different markings



➤ **To add markings to a model, you must:**

1. Add **@dis annotation** switch attributes to the polygons on the model that you want to support markings.
2. Create a *.meta* file that specifies the UVAtlasMapper.
3. Create a *.dtxc* file for each set of markings that maps the annotation names to the coordinates of the associated textures.
4. Create a PNG file with the textures for each texture map.

To change the marking for a particular model, you can configure it in the model definition, as described in "18.10.1 Configuring Texture Mapping in the Model Definition" on page 278. You can also configure markings in the Simulation Object Editor and you can change the markings at runtime with the Set Markings set data request. (For details, please see "Specifying Visual Markings" and "Markings" in the *VR-Forces Users Guide*.)

All of the files for configuring markings on the Fennek are in *./SharedData/16/latest/ModelData/Vehicles/Wheeled/FennekArmoredCar*.

18.11.1 Adding @dis annotations to the Model

In your modeling application, add @dis notes (part of the WRM Specification) to the polygons in your model. (For details about the WRM Specification, please see "Chapter 19. The WRM Specification (DIS Notes)" on page 293).

Figure 110 shows the locations of polygons on the Fennek Armored Car for which annotations have been specified.

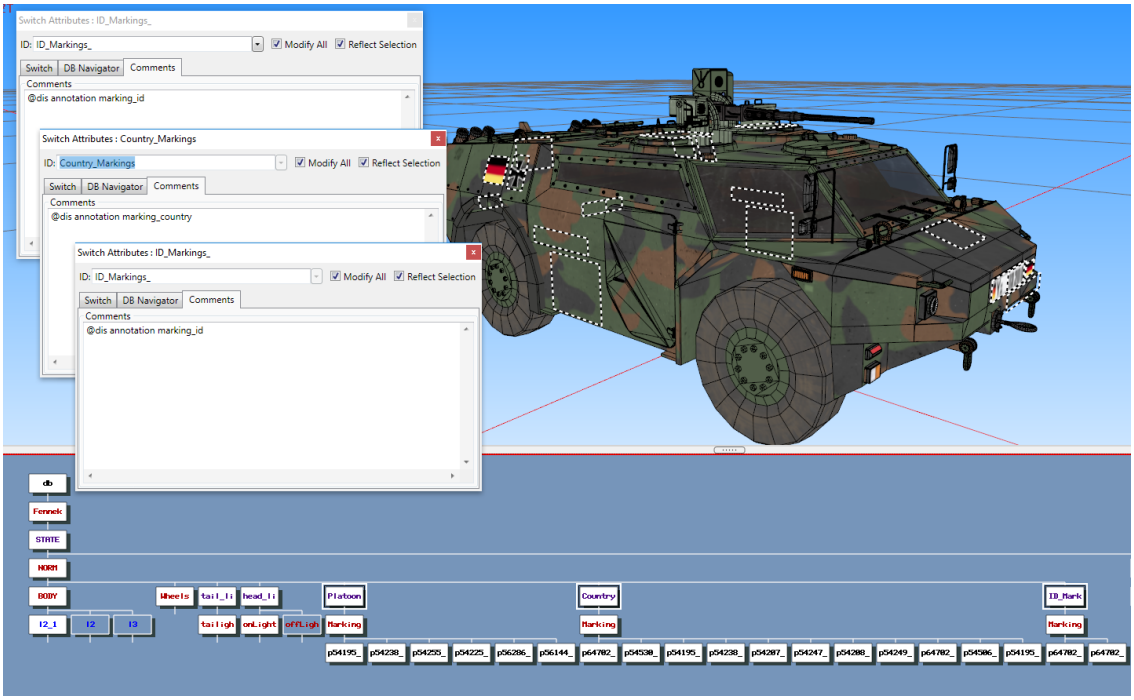
The annotations are in the format:

```
"@dis annotation annotation_name"
```

The specific annotations for the Fennek are:

```
"@dis annotation marking_platoon"
"@dis annotation marking_country"
"@dis annotation marking_id"
```

Figure 110. Specifying @dis annotations in modeling tool



18.11.2 The Metafile for Markings

As with texture skinning, you must create a metafile to configure markings. The following is a portion of the metafile for the Fennek (*./SharedData/16/latest/ModelData/Vehicles/Wheeled/FennekArmoredCar/FennekArmoredCar.meta*). It shows some of the mappings for the **marking_country** annotation and the **marking_id** annotation.

```
{
  "NodeFeatures": [
    ...
    {
      "Linkage": {
        "LinkageName": "country",
        "Comment": "Traversal with match up this feature based on
NodeName match OR AnnotationName match",
        "NodeName": "",
        "AnnotationName": "@dis annotation marking_country",
        "DisplayName": "Country Marking"
      },
      "FeatureData": [
        {
```

```
          "FeatureName": "UvAtlasMapper",
          "AtlasFile": "$(SHARED_DATA_
DIR)\\Vehicles\\Wheeled\\FennekArmoredCar\\CountryMarkings.dtxc",
          "Mappings": [
            {
              "ConfigName": "German",
              "AtlasKeys": {
                 "Pattern": "German"
               }
             },
             {
               "ConfigName": "Canadian",
               "AtlasKeys": {
                 "Pattern": "Canadian"
               }
             },
...
            }
          ]
         }
       ]
     },
     {
       "Linkage": {
         "LinkageName": "id",
        "NodeName": "",
         "AnnotationName": "@dis annotation marking_id",
         "DisplayName": "ID Marking"
       },
       "FeatureData": [
         {
           "FeatureName": "UvAtlasMapper",
           "AtlasFile": "$(SHARED_DATA_
DIR)\\Vehicles\\Wheeled\\FennekArmoredCar\\IDMarkings.dtxc",
           "Mappings": [
             {
               "ConfigName": "03 20 01",
               "AtlasKeys": {
                 "Pattern": "03 20 01"
               }
             },
             {
               "ConfigName": "21 12 02",
               "AtlasKeys": {
                 "Pattern": "21 12 02"
               }
             },
...
          ]
```

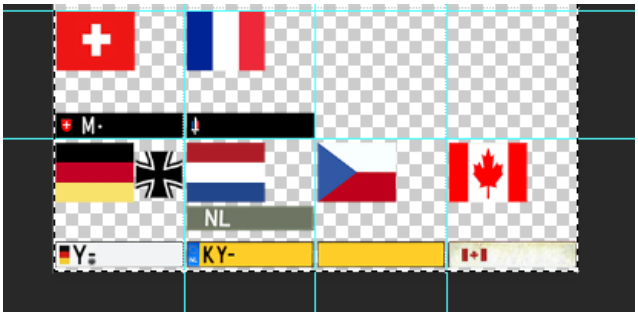
```
        }
      ]
    }
  ]
}
```

18.11.3 The Texture Mapping File

For each type of marking you want to use, you need a *.dtxc* file. It maps the marking annotations to the UV coordinates of the textures in the texture file. The file *CountryMarkings.dtxc* maps the textures for the flags in *CountryMarkings_png.meif* (Figure 111). (Textures are created as PNG files. They are shipped in compressed format.)

```
marking_country,COLPAT,German,0.0,0.0
marking_country,COLPAT,Dutch,0.25,0.0
marking_country,COLPAT,Czech,0.5,0.0
marking_country,COLPAT,Canadian,0.75,0.0
marking_country,COLPAT,Swiss,0.0,0.25
marking_country,COLPAT,French,0.25,0.25
```

Figure 111. CountryMarkings textures



18.12. Flipping DDS Textures for a Model

If a model is displayed upside down because its DDS textures are not what the MAK ONE application expects, you can specify that the model be flipped. (For more information about DDS textures, see "11.9. Displaying DDS Textures Correctly" on page 167.)

➤ **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 80 on page 218).
3. Optionally, filter the model definitions in the list.
4. Select the model for which you want to change the DDS parameter.
5. If the model has the **flip DDS textures** parameter in its parameters list, set it to 1.

If the model does not have the **flip DDS textures** parameter in its parameters list, click the Add Parameter button (+▾) to see if this is a valid parameter for this model. If it is, you can add the parameter and set it to 1.

6. Export the updated setting.

18.13. Configuring SpeedTree Trees

SpeedTree trees are high resolution models of trees. To improve performance, you can set clipping plane and level-of-detail (LOD) values to vary the resolution of the trees placed in the scene based on their distance from the observer. As one example, SpeedTree trees outside of the Fade from High LOD to Low LOD and the Fade from Low LOD to Billboard ranges do not move in the wind.

Table 25 describes the settings you can change. For details about clipping planes, see:

- "Setting the Clipping Planes" in the *VR-Vantage Users Guide*
- "Setting the Clipping Planes" in the *VR-Forces Users Guide*

Table 25. SpeedTree settings

Parameter	Description
Enable SpeedTrees	Enable or disable display of SpeedTrees.
Enable Billboard Rendering	Enable or disable use of billboard rendering. Billboard rendering is used for trees that are further than the Fade from Low LOD to Billboard value. The billboards can be turned off if you do not like the look of them and want to only see the 3D trees. This parameter is primarily for debugging purposes.
Enable 3D Tree Mesh Rendering	Enable or disable use of 3D tree mesh rendering. Turning off the 3D mesh rendering can improve performance slightly, but it requires adjusting the fade values for the LODs to be much smaller so that the billboards show up closer to the camera. This parameter is primarily for debugging purposes.
Enable Shadows on Trees	Enable or disable shadows on trees. Disabling shadows can improve performance slightly.
Clipping Distance	The distance at which trees are clipped out of the scene.
Fade from High LOD to Low LOD	The range (near to far) in which high LOD trees are gradually replaced with low LOD trees.
Fade from Low LOD to Billboard	The range (near to far) in which low LOD trees are gradually replaced with billboard trees. This range is usually greater than the range for high LOD to low LOD.
Fade out Range for Billboards	Specifies the distance at which SpeedTrees fade out. Billboards start to fade out at the clipping distance minus the fade out range. Default: 500 meters.

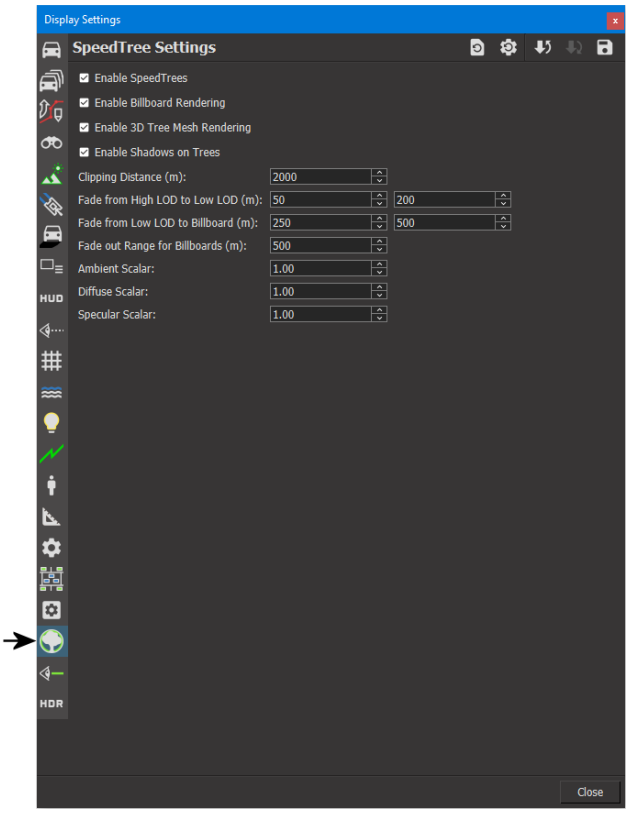
Table 25. SpeedTree settings (continued)

Parameter	Description
Ambient Scalar	Multiplies the built-in ambient lighting value in each SpeedTree by the specified amount.
Diffuse Scalar	Multiplies the built-in diffuse lighting value in each SpeedTree by the specified amount.
Specular Scalar	Multiplies the built-in specular lighting value in each SpeedTree by the specified amount.

➤ **To configure SpeedTree trees:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the SpeedTree Settings page (Figure 112).

Figure 112. SpeedTree Settings page



3. Update parameters as desired.
4. Click Close.

18.13.1 Randomizing the Size of SpeedTrees

SpeedTrees can be created as props or streamed into the scene using an earth file. They have a default size. You can configure MAK ONE applications to scale the size of SpeedTrees by a fixed amount or you can scale them randomly within a specified range. The scale applied to a SpeedTree is the product of several factors.

If a SpeedTree is a prop, it is scaled as the product of the following settings:

- The **scale** attribute in its model definition. If the **scale** attribute is not part of the model definition, the value defaults to 1.0.
- A random scale factor as specified on the SpeedTree Randomization page on the Terrain Settings dialog box, if enabled.
- A feet-to-meters conversion factor of 0.3048.

If a SpeedTree is a streamed in using an earth file, it is scaled as the product of the following settings:

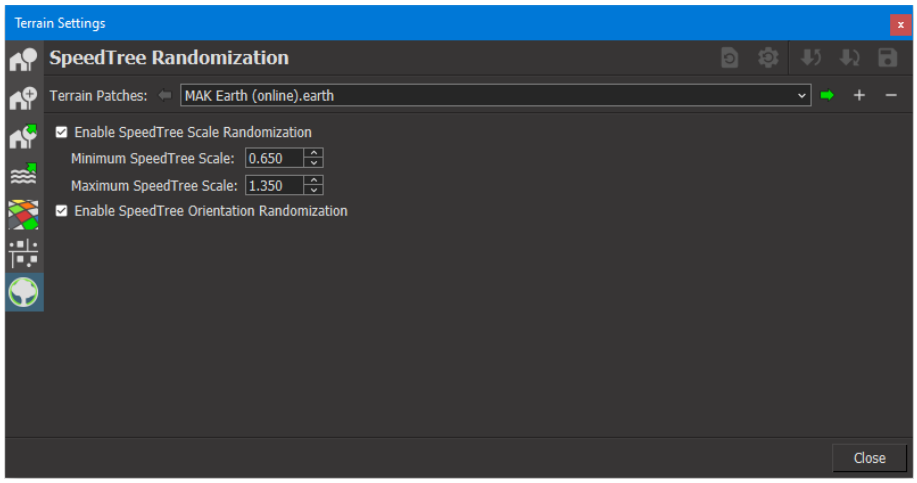
- The **scale** attribute in its model definition. If the scale attribute is not part of the model definition, the value defaults to 1.0.
- The **marker-scale** attribute in the **model** element that configures the trees.
- A random scale factor as specified on the SpeedTree Randomization page on the Terrain Settings dialog box, if enabled.
- A feet-to-meters conversion factor of 0.3048.

You can also randomize the orientation of SpeedTrees.

➤ **To specify randomization of SpeedTree scaling:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the SpeedTree Randomization page (Figure 113).

Figure 113. SpeedTree Randomization page



3. To enable randomization of the SpeedTree scale, select the Enable SpeedTree Scale Randomization check box.
4. If you enabled SpeedTree scale randomization, specify values for the Minimum SpeedTree Scale and Maximum SpeedTree Scale.
5. To enable randomization of tree orientation, select the Enable SpeedTree Orientation Randomization check box.

18.13.2 Displaying SpeedTrees in Planar Reflections

By default, SpeedTrees are not displayed in planar reflections. You can enable their display. For details, see:

- "Configuring Ocean Planar Reflections" in the *VR-Vantage Users Guide*
- "Configuring Ocean Planar Reflections" in the *VR-Forces Users Guide*

18.14. Best Practices for Creating Models

If you build your own models for use in MAK ONE applications, following these guidelines will result in models that work efficiently:

- Keep all textures to powers of two, for example 256x256, 128x512, or 2048x2048:
  - Make sure textures use as many of the pixels as possible and that there are no large black areas in the texture.
  - Keep alpha (transparency) to a minimum.
- Minimize the number of textures:
  - Create a texture atlas that uses multiple textures combined into one.
  - Group polygons together that use the same texture.
  - Use the same textures for all LOD nodes.
- Minimize the number of nodes:
  - Keep the hierarchy as flat as possible.
  - Move all LOD nodes to the top of the hierarchy.
  - Minimize unnecessary transforms.
  - When you reuse geometry in the model, use instancing.
- Minimize the number of polygons that have different attributes:
  - Use the same material on polygons whenever possible.
  - Use two polygons with flipped normals instead of using the double sided attribute.
  - Use the same color on polygons whenever possible.
  - Group polygons with similar attributes.
- Consider using a more aggressive LOD structure:
  - Reduce the switchout distance to be smaller so that the LODs switch sooner.
  - Minimize the number of LODs.
  - Make sure that models switch out completely at a reasonable distance.
- Reduce the number of external references:
  - Merge external reference files.
  - Create instances from external merged reference files.

18.15. Disabling Indirect Rendering for an OpenFlight Model

In general, indirect rendering improves performance without affecting the quality of models. However, there may be cases where using indirect rendering causes artifacts in models. If there is a part of a terrain that rendered well in VR-Vantage 2.3.1, VR-Forces 4.6.1, or earlier versions of these applications, but does not render well with indirect rendering, you can disable indirect rendering for the OpenFlight model.

To be sure that indirect rendering is the cause of the problem, disable indirect rendering in the Add Terrain Patch dialog box. If the problem goes away, then you can disable indirect rendering for the affected portion of the model.

To disable indirect rendering within the model, add the tag **@dis DisableIndirect** to the comment for the appropriate node in the OpenFlight model. Indirect rendering will be disabled for that node and all of its child nodes. The rest of the model will continue to benefit from indirect rendering.

18.16. Compressing Model Files

The models and image files provided with MAK ONE applications are shipped in compressed formats (MAK Encrypted Data Format (MEDF) and MAK Encrypted Image Format (MEIF)). You may want to compress your models to save space or protect intellectual property. You can convert supported formats to MEDF and MEIF with *./bin64/medfTool.exe*. The compression tool supports the 3D modeling formats supported by MAK ONE applications, such as OpenFlight and 3DS and raster formats such as RGB and PNG.



When a MAK ONE application processes a request to load a file, such as an external reference in a terrain, it checks the cache and the directory of the requested file for an MEDF version of the file. If it does not exist, it loads the file in the original format.

The syntax for medfTool is as follows:

```
medfTool [-b -d --directory directory --extendLodsEqual distance
  --extendLodsGreaterOrEqual distance --extendLodsTo distance -f
  filename --filter paths -h -i --genDDS -m extensions
  --norecurse -o -p <integer> -q simple_filename -r #comment
  --removeHighestLod -s directory --scaleLod scalar -t -- -V -v
  -x extensions -z optimization]
```

Table 26 describes the arguments.

Table 26. medfTool arguments

Argument	Description
(-- | --ignore_rest)	Ignore all arguments after this one.
(-b | --ignoreNonBaseTex)	Ignores any non-base texture (diffuse texture) for a model. This is useful when you do not want any multiple diffuse textures or any effect textures associated with it.
(-d | --dds_flip)	Flip DDS images before encrypting them.
--directory *directory*	Specifies the directory to search for image and geometry files to convert into MEDF and MEIF. All subdirectories are visited unless the **--norecurse** command line option is used.
--extendLodsEqual *distance*	Match LODs with switch-in distance equal to this argument and extend them to the value from the **--extendLodTo** argument
--extendLodsGreaterOrEqual *distance*	Match LODs with switch-in distance equal to or greater than this argument and extend them to the value from the **--extendLodsTo** argument
--extendLodsTo *distance*	The LOD switch-in distance that **--extendLodsEqual** and **--extendLodsGreaterOrEqual** extend to.
(-f | --file) *filename*	Convert just this file. The extension of the file must be specified using **-x** or **-m** to indicate it is an image or a geometry file.
--filter *paths*	Regular expression to match for filtering directories. Matches are not recursed into. Accepted multiple times.
--genDDS	Generate DDS files.
(-h | --help)	Displays usage information.
(-i | --ignore_errors)	Ignore errors in the encryption process and continue without exiting.
(-m | --image_extensions) *extension*	Specifies the extensions of image formats which will be converted into MEIF format. Use this argument to encrypt a directory. Can be used multiple times.
--norecurse	Do not recurse through subdirectories. Default: recurse through subdirectories.
(-o | --onlyIfNew)	Only generates an encrypted file if the expected output file does not exist.

Table 26. medfTool arguments (continued)

Argument	Description
(-p | --threads) *threads*	Specifies the number of threads to run. Specifying zero results in the number of threads equaling the number of logical processors the host machine has.
(-q | --forceOpaque) *simple_filename*	Specifies a filename and extension, but no path, to force polygons with these filenames to be opaque. Accepted multiple times.
(-r | --removeNodes) *#comment*	Removes any any node tagged with the comment from the scene graph in a model. This is used in special cases to remove nodes that are not needed. Example, **-r #missionfunctiononly**.
--removeHighestLod	Removes the highest LOD under a group if it has more than one LOD.
(-s | --search) *directory*	Adds a directory to the search path when resolving external references. During processing, any external references not found are removed from the generated files. Files in the search path are not converted.
--scaleLod *scalar*	Scales the switch in/out distance of all the children under an osg::LOD node by the specified value.
(-v | --version)	Display version information and exit.
(-V | --verbose)	Verbose output.
(-x | --extensions) *extension*	Specifies the extensions of geometry file formats that will be converted into MEDF format. Use this argument to encrypt a directory. Can be used multiple times.
(-z | --optimize) *optimization*	Specifies the optimization level. Maximum optimization may provide better optimization than the default setting, but it is significantly slower. Possible values are: • 0 = None. • 1 = Default. • 2 = Maximum optimization.

The following example recursively compresses all FLT, RGB, RGBA, PNG, and DDS files in *%MAK_DATADIR%/Terrain*:

```
 .%MAK_IGDIR%/bin64/medfTool.exe
--directory %MAK_DATADIR%/Terrain --extensions flt
--image_extensions rgb --image_extensions rgba
--image_extensions png --image_extensions dds
```

The following example compresses all FLT, RGB, RGBA, PNG, and DDS files in *%MAK_DATADIR%/Lifeforms*. It does not compress files in subdirectories.

```
 .%MAK_IGDIR%/bin64/medfTool.exe
--directory %MAK_DATADIR%/Lifeforms --norecurse
--extensions flt --image_extensions rgb
--image_extensions rgba --image_extensions png
--image_extensions dds
```



19. The WRM Specification (DIS Notes)

Some OpenFlight files contain information specified by the *WRM Entity Model Specification for DIS Interoperability*. This information makes it possible for MAK ONE applications to depict the state of an entity's articulated and attached parts correctly and to switch among different entity appearances.

This chapter contains information that can help you build entity models that comply with the WRM specification, or add that compliance to existing models.

The WRM (world reference model) specification was developed by MAK Technologies, Paradigm Simulation Inc., MultiGen Inc., and SAIC for the ARPA War Breaker project. (Paradigm Simulation and MultiGen merged and were later acquired by Presagis.)



Although the WRM specification was developed for DIS, it is also used in the HLA RPR FOM.

19.1. Introduction 294
19.2. The OpenFlight File Format 294
19.2.1 Node Name and Comment Fields 294
19.2.2 External References and Instancing 295
19.3. Mapping FBX Model Nodes to DIS Notes 295
19.3.1 Specifying the FBX Texture Path 296
19.4. Modeling for DIS Interoperability 296
19.4.1 The DIS Attribute Lexicon (DAL) 296
19.4.2 DAL Keywords 297
19.4.3 Model Coordinate Systems 297
19.5. Articulated Parts 300
19.6. Entity Appearances 300
19.7. Animation 303
19.8. DIS Keyword Values 303

19.1. Introduction

In order for MAK's visualization applications to change the state of a model in response to network messages, they must have certain information about the model. For example, which piece of the model's geometry is associated with which articulated part, and which piece of a model is associated with a particular entity appearance attribute.

This chapter describes a method of encoding information in an OpenFlight format model so that applications can make these associations. For the most part, information is associated with a MultiGen node by including it as a comment associated with the node. Comments that contain this information begin with the token **@dis**.

MAK ONE applications depict the movement of articulated parts and changes in entity appearances only when using models constructed to this specification.

This chapter represents a subset of the Model Specification for DIS Interoperability (version 001). For more details, please refer to that document.

Although MAK ONE applications implement only a subset of the functionality enabled by the WRM Model Specification, we recommend that models conform to as much of the specification as is appropriate for the model, to promote compatibility with future versions of MAK ONE applications and with visualization applications from other vendors.

19.2. The OpenFlight File Format

The OpenFlight file format organizes geometry into the following types of nodes:

- Face nodes represent individual polygons.
- Object nodes represent face nodes that are logically or geographically related. Object nodes can have only face nodes (polygons) as children.
- Level of detail (LOD) nodes provide a means to select alternate geometries based on a user-definable range. Using lower resolution geometry for objects that are far from the observer saves memory and improves performance.
- Degree of Freedom (DOF) nodes specify translational or rotational freedom, or both, for portions of a model. They can be used for DIS articulated parts or animated special effects. They specify a type of transformation and its limits, such as "rotate about the x axis between 45 and 135 degrees". The OpenFlight approach to degrees of freedom is hierarchical.
- Group nodes represent logically-related or geographically-related object nodes. Group nodes are the most general, and may have any type of nodes, except face nodes, as children.

19.2.1 Node Name and Comment Fields

Each node contains a comment field for user-definable data. The WRM Specification defines keywords that, when placed in a comment field, provide the information that DIS applications need to properly render a model.

19.2.2 External References and Instancing

The following OpenFlight constructs provide a means to avoid duplicating massive amounts of geometry:

- An external reference (or xref) is a pointer to another OpenFlight file that contains additional geometry.
- A transformation matrix is usually attached to an xref to locate the position and orientation of the referenced object's coordinate system geometry. Each instance may have a transformation matrix controlling the position and orientation of the child's coordinate system.

19.3. Mapping FBX Model Nodes to DIS Notes

The FBX model format does not have a standard convention for commenting nodes for runtime use. Therefore, MAK has had to develop a way to interpret the nodes in FBX files so that it can visualize LODs and articulated parts.

For LODs, the MAK convention roughly follows the conventions used by the Unity engine. For articulated parts, MAK follows the conventions used by Aegis Corp.

Mapping of FBX nodes to DIS notes depends on using articulated part names for the nodes in the models. You can do this if you are using 3DS Max.

If you are creating FBX models for use in MAK ONE applications, following the MAK convention will allow you to import the model into a MAK ONE application with a minimum of difficulty. If you do not follow the conventions in naming nodes, you will need to map the nodes in your FBX files to DIS notes by hand as described in this section.

FBX nodes are mapped in the file *./appData/importConfig/FBX_NodeNameMap.csv*. The following example shows a portion of this file:

```
Node,Comment,Type,Description
turret_turn,@dis articulated_part 4096,comment,Primary Turret Number 1
main_cannon_elev,@dis articulated_part 4416,comment,Primary Gun Number
1
m2_group_turn,@dis articulated_part 4160,comment,Primary Turret Number
3
m2_group_elev,@dis articulated_part 4480,comment,Primary Gun Number 3
m240_group_elev,@dis articulated_part 4448,comment,Primary Gun Number 2
wheel_1_1_group,@dis articulated_part 9120,comment,1st Left Wheel
wheel_1_2_group,@dis articulated_part 9152,comment,1st Right Wheel
wheel_10_1_group,@dis articulated_part 9696,comment,10th Left Wheel
wheel_10_2_group,@dis articulated_part 9728,comment,10th Right Wheel
turret_trav_group,@dis articulated_part 4096,comment,Primary Turret
Number 1
cannon_elev_group,@dis articulated_part 4416,comment,Primary Gun Number
1
SimpleVehicle,Wreck,DamageState,Match node name to NORMAL/DAMAGE state
High,@dis lod 0 switch_in 50 switch_out 0,,
Med,@dis lod 1 switch_in 200 switch_out 50,,
Low,@dis lod 2 switch_in 1000 switch_out 200,
```

You can edit this file in a text editor or spreadsheet application. There are four columns, as follows:

- Node. The name of the node in the FBX file.
- Comment. The DIS note to apply to this node, as specified in the WRM specification.
- Type. The type of DIS note, for example a comment or an LOD switch.
- Description. A description of what the node represents.

When you create an FBX model, you can refer to this file and use the mappings that MAK has created for the FBX models that it includes in the application distribution. Or, you can update the file to map the nodes in your FBX models to the appropriate DIS notes. You can also use a combination of these approaches.

19.3.1 Specifying the FBX Texture Path

Most of the models provided with MAK ONE applications have their textures in the same directory as the model files. Imported models might not follow this convention. Therefore, MAK provides a way to specify search paths for textures. To specify an alternate texture directory, you can edit the file *./appData/importConfig/FBX_TexturePath.csv*.

19.4. Modeling for DIS Interoperability

The WRM Specification provides uniform standards for locating and orienting the model coordinate system and a way to map DIS codes to associated models and parts. By conforming to these standards, applications that support the WRM Specification do not need customized software to interpret the multiple potential model formats that might otherwise exist.

19.4.1 The DIS Attribute Lexicon (DAL)

DIS attributes are specified using the following syntax:

`@dis` *`keyword values`* `# comment`

where:

- **`@dis`** must precede the keyword.
- *`keyword`* is one of the DAL keywords listed below.
- *`values`* can be a number or string. Depending on the requirements of the keyword, you can specify an individual value, a comma-separated list, or a range.
- **`#`** indicates the start of a comment. Comments are case sensitive. A comment continues until the next **`@dis`** entry.

Remember that all comments will be parsed by MAK ONE applications, so spelling is important.

19.4.2 DAL Keywords

The WRM Spec supports the following keywords:

- **articulated_part** *part*. Identifies the movable parts of a model with a unique DIS Articulated Part code. This keyword is placed in a Group node that is a child to the DOF node that identifies an articulated part, and above the geometry that is going to be articulated. This keyword is always followed by the motion keyword. The articulated part codes are specified in DIS Standard SISO-REF-010-2006 Enumerations and Bit Encoded Values Articulated Parts. They are listed in Table 40 on page 320 .

  This keyword is always followed by the switch keyword.

- **switch** *appearance*. Identifies a model appearance attribute with a DIS keyword. This keyword is placed in a Switch node in the model above the geometry that is going to be switched.

  The switch keyword takes the name value in DIS Standard SISO-REF-010-2006 Enumerations and Bit Encoded Values Entity Appearance (General and Specific), joined by the underscore character (_) if the name is multiple words, as its value. Note that frozen_status, power_plant_status, and state are preceded by an abbreviation for their domain, as follows: gm_ for guided munitions and lf_ for life forms. For a list of supported values, please see Table 32 on page 311.

- **state** *state*,*state*... . Specifies the state of the appearance attribute with a DIS appearance state code. This keyword is placed in a Group node just below the switch node. For a list of supported values, please see Table 32 on page 311.

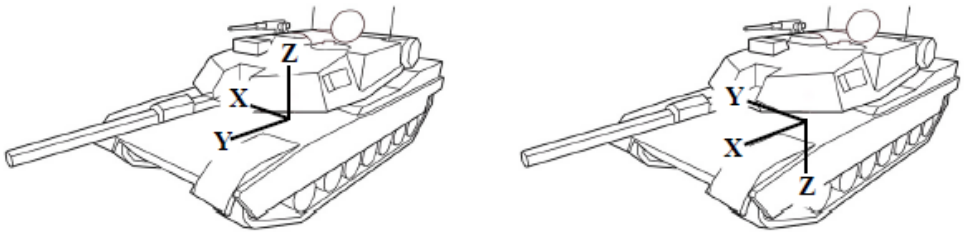
19.4.3 Model Coordinate Systems

OpenSceneGraph (OSG) and the DIS Standard use different right-hand coordinate systems. Table 27 describes the two systems. Figure 114 illustrates them.

Table 27. Model coordinate systems

Coordinate	OpenSceneGraph	DIS
X	Right	Forward
Y	Forward	Right
Z	Up	Down

Figure 114. Model coordinate systems



OSG coordinate system DIS coordinate system

When you create models, it is recommended that you use the OpenSceneGraph coordinate system. The simulation application will convert the coordinates, as follows:

- xmodel = yDIS.
- ymodel = xDIS.
- zmodel = -zDIS.

Location of the Origin for Models

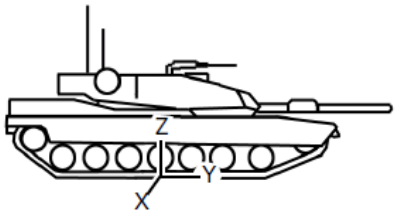
The origin of the model's coordinate system should be located as follows, depending on the domain of the entity:

- Land: On the base of the vehicle, on the axis of yaw rotation.
- Surface: On the axis of the center of buoyancy, in the plane of the waterline for unloaded ship.
- Subsurface: On the axis of the center of buoyancy, in the plane of the waterline for surfaced submarine.
- Air and Space: At the center of gravity.

Figure 115 illustrates the location of a model's origin for each platform domain.

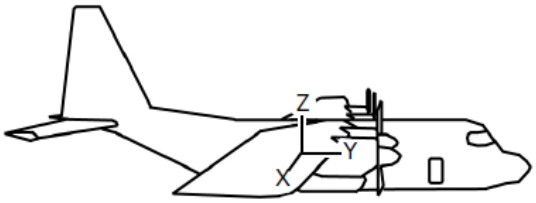
Figure 115. Location of origin by domain

Land



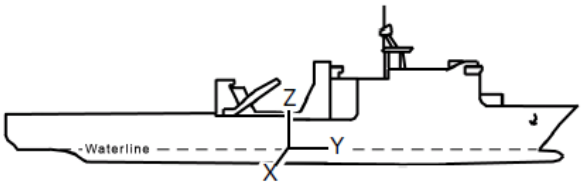
On the plane of ground contact, at the axis of yaw rotation.

Air



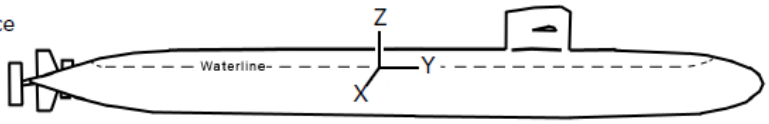
At the center of gravity.

Surface



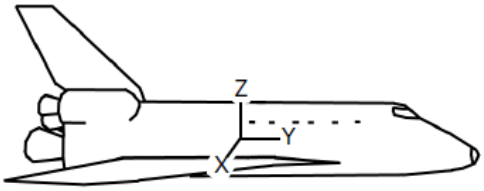
On the axis of the center of buoyancy, in the plane of the waterline for an unloaded shi

Subsurface



On the axis of the center of buoyancy, in the plane of the waterline of a surfaced submari

Space



At the center of gravity.

19.5. Articulated Parts

For MAK ONE applications to properly depict the state of articulated parts, models must conform to the following specifications:

- There must be one DOF (degree of freedom) node for each articulated part. (This means that if there are several LODs, they should be children of the DOF.) Each articulation DOF should be a descendant of the DOF for the part to which it is attached (unless the parent part is just the base model itself). For example, the DOF for a gun should be a descendant of the DOF for a turret.
- Each articulated part DOF node must have all of the geometry and other nodes associated with the part underneath it. This DOF node must have a comment associated with it. The comment must be of the form:

  `@dis articulated_part` *`part-number`*

  where *`part-number`* is the enumeration for the particular type of part that this part of the model represents. For example, the part number of a turret is 4096, and the part number of a gun is 4416.

  Therefore, the child group of a turret's DOF must be commented with:

  ```
  @dis articulated_part 4096
  ```

  The child group of a gun's DOF must be commented with:

  ```
  @dis articulated_part 4416
  ```

- The WRM Model Specification specifies that information about the legal motions of each articulated part be encoded in the comment field. Currently, MAK ONE applications do 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.

19.6. Entity Appearances

MAK ONE applications support switching between different pieces of a model's geometry based on appearance attributes. For instance, a tank's turret should look different if the hatch is open or closed. An entire tank should look different if it is damaged or destroyed.

Switch nodes are used to represent a choice between several different children, based on a particular appearance attribute.

We define a switch node as a group node with a comment of the form:

`@dis switch` *`attribute`*

where *`attribute`* is the name of the attribute that should be used to select a child. (For example, damage, smoke, hatch, and so on.)

The children of the switch node represent the different portions of a model among which MAK ONE applications will select, based on the appearance attribute. If a switch node is commented with:

```
@dis switch hatch
```

it may, for example, have three children containing geometry for a closed hatch, popped hatch, and open hatch.

These children of the switch node must be commented so that MAK ONE applications know which child matches which values of the appearance attribute. The comment is of the form:

```
@dis state value-list
```

value-list is of the form *m[,m]+* where *m* is one of the following:

- An integer representing the enumeration of a value that an appearance attribute may take on, for example, 1 for hatch closed.
- Two integers, a and b, separated by a hyphen, representing all enumerations from a through b, for example, 2-5 for all states whose enumerations are 2, 3, 4, or 5.
- The integer -1, which indicates that this child is the default child, to be used when the attribute state matches no values in any children.

For instance, the child of the hatch switch node that represents a closed hatch might be used when the hatch attribute value is either 0 (not-applicable), 1 (closed), or any value other than those enumerated in other children. This child would be commented:

```
@dis state 0,1,-1
```

The child with the popped hatch geometry should be used when the value is 2 (popped) or 3 (popped with person visible) and is commented:

```
@dis state 2,3
```

The child that contains the open hatch geometry should be used when the value is 4 (open) or 5 (open with person visible) and is commented:

```
@dis state 4,5
```

There may be many switch nodes for the same attribute. For example, there may be several pieces of a tank's geometry that are affected by the destroyed attribute.

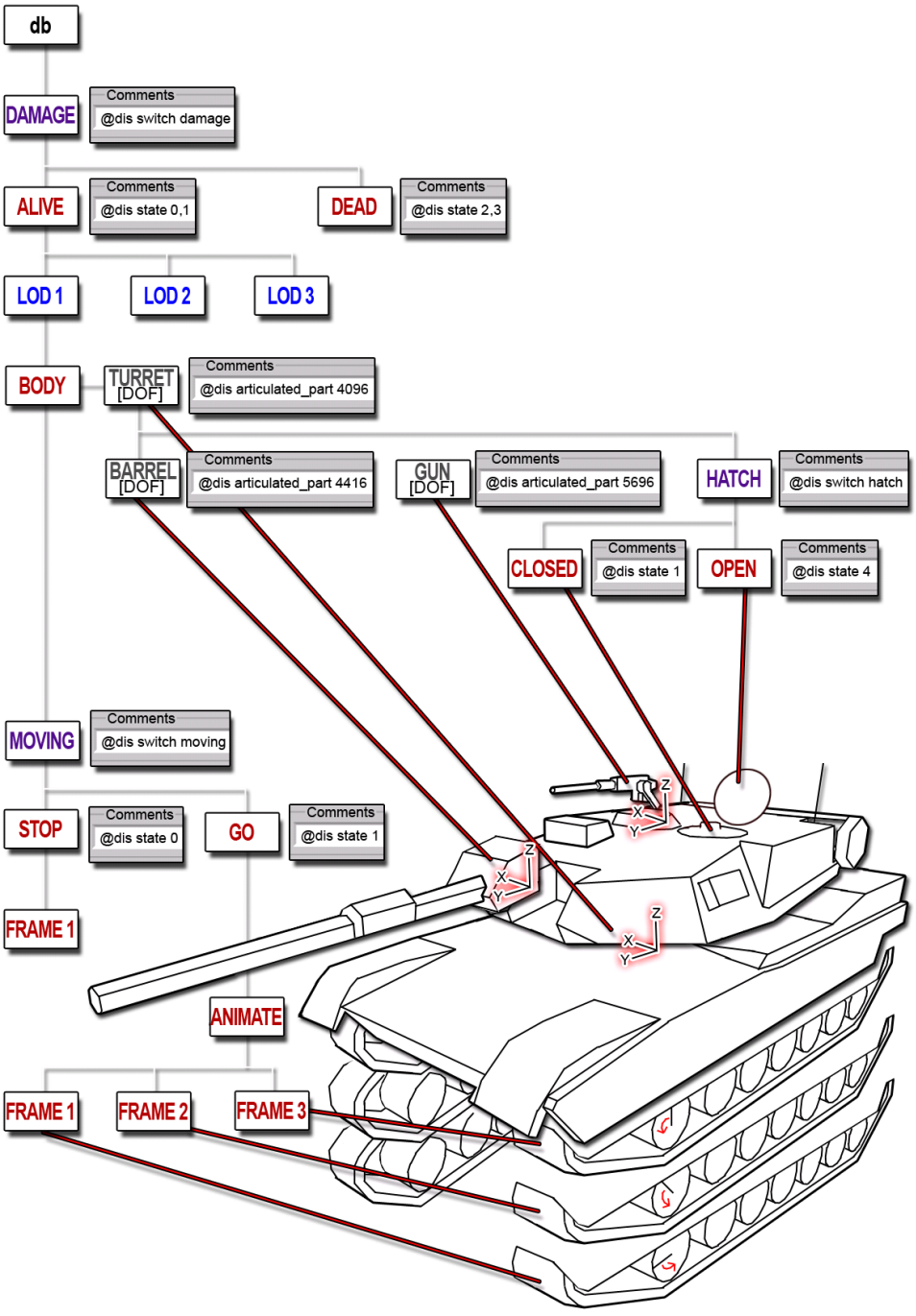
The WRM Model Specification also discusses the inheritance of appearance attributes from ancestor nodes. MAK ONE applications do not support this functionality. To tell a MAK ONE application to, for example, display a camouflaged tank when the paint scheme attribute is set to camouflage, and a normal tank otherwise, you must use the switch node method described previously.

In Figure 116, the turret DOF node is attached to the body Group node. The barrel, gun, and hatch DOF nodes are attached to the turret. This allows the turret to be positioned in the coordinate system of the body, and the barrel, gun, and hatch to be positioned in the coordinate system of the turret.

This model also contains a switch node that is flagged with animation. The comment on this node is **@dis switch moving**. This switches between two Group nodes – Stop and Go. Under the Stop node there is a single mesh of the treads of the tank. Under the Go node there is a Group node, with the animation flag set on. This creates a flipbook animation of all nodes below this Group node. There are three meshes in different states of moving: Frame 1, Frame 2 and Frame 3. At runtime OpenSceneGraph will animate these meshes when the velocity is not 0.

If multiple color schemes are needed to represent camouflage, winter, and desert versions, another switch is needed.

Figure 116. 3D model structure



19.7. 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 a MAK ONE application that animation instructions follow.
- *`num_frames_per_second`* is the rate at which the application should display the frames.
- `skip` and `noskip` indicate whether the elapsed time should be used to choose the next frame, or whether at most one frame should be advanced each frame.
- You can also specify how many times to loop through an animation and how long to hold the last frame with the following comments:
- `@dis loopCount` *`count`*, where *`count`* is the number of times to loop through the animation. Default: -1 (loop forever). 0 means do not loop.
- `@dis lastFrameTime` *`time`*, where *`time`* specifies how long, in seconds, to hold the last frame of the animation.

The `loopCount` and `lastFrameTime` comments only work with non-instanced models. You must use them with an `animation` comment, for example:

```
@dis animation 2 skip @dis loopCount 3 @dis lastFrameTime 11.5
```

19.8. DIS Keyword Values

Table 28 through Table 42 list the values that you can use for the `@dis switch` keyword by kind and domain. For any switch keyword, the supported state values (`@dis state [value 1],[value 1]...`) are listed in the State column. The Bit column lists the appearance bit for setting DIS appearances. (Some bit numbers are repeated because they are valid only for a particular platform.) In the Values column, parentheses enclose older values that still work but have been replaced by the value above them.

- Table 28. DIS appearance and states — platform kind, land domain on the next page
- Table 29. DIS appearance and states — platform kind, air domain on page 306
- Table 30. DIS appearance and states — platform kind, surface domain on page 309
- Table 31. DIS appearance and states — platform kind, subsurface domain on page 310
- Table 32. DIS appearance and states — platform kind, space domain on page 311
- Table 33. DIS appearance and states — lifeform kind on page 312

- Table 34. DIS appearance and states — munition kind on page 314
- Table 35. DIS appearance and states — environmental kind on page 315
- Table 36. DIS appearance and states — cultural feature kind on page 316
- Table 37. DIS appearance and states — sensor kind on page 316
- Table 38. DIS appearance and states — supply kind on page 318
- Table 39. DIS appearance and states — expendable kind on page 319
- Table 40. DIS Articulations on page 320
- Table 41. DIS Attachments on page 326
- Table 42. DIS Animations on page 327

Table 28. DIS appearance and states — platform kind, land domain

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
2	fire_power_killed (fire_power)	Characteristics of fire-power kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	trailing_dust_cloud (trailing_effects)	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large

Table 28. DIS appearance and states — platform kind, land domain (continued)

Bit	Values	Description	State/ Description
9-11	primary_hatch (hatch)	State of the primary hatch.	0 = Not Applicable 1 = Closed 2 = Popped 3 = Popped and Person Visible 4 = Open 5 = Open and Person Visible 6 = Custom1 7 = Custom2
12	head_lights_on (head_lights)	State of headlights.	0 = No 1 = Yes
13	tail_lights_on (tail_lights)	State of tail lights.	0 = Off 1 = On
14	brake_lights_on (brake_lights)	State of brake lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	launcher/operational (launcher)	State of the platform's primary launcher.	0 = Not Raised 1 = Raised
17-18	camouflage_type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused
19	concealed_position (concealed)	Type of concealment.	0 = Not Concealed 1 = In Concealed Position
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_status)	State of the entity’s power-plant.	0 = Off 1 = On
24	tent_extended (tent)	State of a tent extension.	0 = Not Extended 1 = Extended

Table 28. DIS appearance and states — platform kind, land domain (continued)

Bit	Values	Description	State/ Description
25	ramp_extended (ramp)	State of a ramp.	0 = Down 1 = Up
26	blackout_lights_on (blackout_lights)	State of blackout lights.	0 = Off 1 = On
27	blackout_brake_lights _on (blackout_brake_ lights)	State of blackout brake lights.	0 = Off 1 = On
28	spot/search_light _#1_on (spot_lights)	Describes whether spotlights are on or off.	0 = Off 1 = On
29	interior_lights _(forward)_on (interior_lights)	State of interior lights.	0 = Off 1 = On
30	occupants_ surrendered	Describes whether the vehicle's occupants have surrendered.	0 = No 1 = Yes
31	masked/cloaked (masked_cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 29. DIS appearance and states — platform kind, air domain

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	propulsion_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
2	nvg_mode	Describes whether air platform lighting is in covert or overt mode.	0 = Overt Lighting 1 = Covert Lighting (Night Vision Goggles)
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes

Table 29. DIS appearance and states — platform kind, air domain (continued)

Bit	Values	Description	State/ Description
7-8	trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
9-11	canopy/troop_door (hatch)	State of the canopy or troop door.	0 = Not Applicable 1 = Single Canopy/Single Troop Door Closed 2 = Front and Rear Canopy/Left and Right Troop Door Closed 3 = Front Canopy/Left Troop Door Open 4 = Single Canopy/Single Troop Door Open 5 = Rear Canopy/Right Troop Door Open 6 = Front and Rear Canopy/Left and Right Troop Door Open
12	landing_lights_on (landing_lights)	State of landing lights.	0 = Off 1 = On
13	navigation_lights_on (navigation_lights)	State of navigation lights.	0 = Off 1 = On
14	anti-collision_lights_on (anticollision_lights_on) (anticollision_lights)	State of anticollision lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	afterburner_on (afterburner)	State of an air platform's afterburner.	0 = Off 1 = On
17	lower_anti-collision_lights_on (lower_anti_collision_lights_on)	Describes whether the lower anticollision lights are on or off.	0 = Off 1 = On

Table 29. DIS appearance and states — platform kind, air domain (continued)

Bit	Values	Description	State/ Description
18	upper_anti-collision _lights_on (upper_anti_collision _lights_on)	Describes whether the upper anticollision lights are on or off.	0 = Off 1 = On
19	anti-collision_lights_ day/night (anti_collision_ lights_day/night)	Day/night status of the anticollision lights.	0 = Day 1 = Night
20	is_blinking	Indicates whether any lights are blinking.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On
23	state	Describes whether the entity is active.	0 = Deactivated 1 = Active
24	formation_lights	State of formation lights.	0 = Off 1 = On
25	landing_gear _extended	Describes whether the landing gear is wholly retracted or extended.	0 = Retracted 1 = Extended
26	cargo_doors_opened	Describes whether the cargo doors (main door) are closed or open.	0 = Closed 1 = Open
27	navigation/position _brightness	Describes the brightness of the navigation/position lights.	0 = Dim 1 = Bright
28	spot/search_light_#1 _on (spot_lights)	Describes whether spotlights are on or off.	0 = Off 1 = On
29	interior_lights_on (interior_lights)	State of interior lights.	0 = Off 1 = On
30	reverse_thrust _engaged	Describes whether the air platform has engaged reverse thrust.	0 = No 1 = Yes
31	weight-on-wheels	Describes whether the air platform has weight on its main landing gear.	0 = No 1 = Yes

Table 30. DIS appearance and states — platform kind, surface domain

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	wake_size (trailing_effects)	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
12	running_lights_ on (running_lights)	State of running lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity’s power-plant.	0 = Off 1 = On
23	State	Describes whether the entity is active.	0 = Deactivated 1 = Active

Table 30. DIS appearance and states — platform kind, surface domain (continued)

Bit	Values	Description	State/ Description
28	spot_lights_on (spot_lights)	Describes whether spotlights are on or off.	0 = Off 1 = On
29	interior_lights_ on (interior_lights)	State of interior lights.	0 = Off 1 = On

Table 31. DIS appearance and states — platform kind, subsurface domain

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
9-11	hatch	State of the primary hatch.	0 = Not Applicable 1 = Closed 2 = Popped 3 = Popped and Person Visible 4 = Open 5 = Open and Person Visible 6 = Custom1 7 = Custom2
12	running_ lights_on (running_ lights)	State of running lights.	0 = Off 1 = On

Table 31. DIS appearance and states — platform kind, subsurface domain (continued)

Bit	Values	Description	State/ Description
15	is_flaming (flaming)	Describes whether flames are visible.	0 = No 1 = Yes
21	is_frozen (frozen_ status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_ on (power_plant_ status)	State of the entity's power plant.	0 = Off 1 = On

Table 32. DIS appearance and states — platform kind, space domain

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke or vapor is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
15	is_flaming (flaming)	Describes whether flames are visible.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_ on (power_plant_ status)	State of the entity's power plant.	0 = Off 1 = On

Table 33. DIS appearance and states — lifeform kind

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5-8	compliance_status (compliance)	Compliance of life form.	0 = Lifeform None 1 = Detained 2 = Surrender 3 = Using Fists 4 = Verbal Abuse 1 5 = Verbal Abuse 2 6 = Verbal Abuse 3 7 = Passive Resistance 1 8 = Passive Resistance 2 9 = Passive Resistance 3 10 = Non-Lethal Weapon 1 11 = Non-Lethal Weapon 2 12 = Non-Lethal Weapon 3 13 = Non-Lethal Weapon 4 14 = Non-Lethal Weapon 5 15 = Non-Lethal Weapon 6
11	signal_smoke_in_use	Describes whether signal smoke is being used.	0 = No 1= Yes
12	flash_lights_on (flashlight)	State of lights.	0 = Off 1 = On
13	signal_mirror_in_use	Describes whether a signal mirror is being used.	0 = No 1= Yes

Table 33. DIS appearance and states — lifeform kind (continued)

Bit	Values	Description	State/ Description
14	ir_stobe_on	Describes whether an IR strobe is on or off.	0 = Off 1 = On
15	ir_illuminator_on	Describes whether an IR illuminator (flare) is on or off.	0 = Off 1 = On
16-19	life_form_posture (life_form_state)	Life form action.	0 = None 1 = Upright Still 2 = Upright Walking 3 = Upright Running 4 = Kneeling 5 = Prone 6 = Crawling 7 = Swimming 8 = Parachuting 9 = Jumping 10 = Sitting 11 = Squatting 12 = Crouching 13 = Wading 14 = Surrender 15 = Detained
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	mounted/hoisted _status	Describes whether the life form is mounted or hoisted on a platform such as a troop transport or helicopter hoist.	0 = Not Mounted/Hoisted 1 = Mounted/Hoisted
24-25	weapon/implement_1 (weapon_1)	State of primary weapon.	0 = None 1 = Stowed 2 = Deployed 3 = Firing Position
26-27	weapon/implement_2 (weapon_2)	State of secondary weapon.	0 = None 1 = Stowed 2 = Deployed 3 = Firing Position

Table 33. DIS appearance and states — lifeform kind (continued)

Bit	Values	Description	State/ Description
28-29	camouflage_type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused
30	concealed_stationary (concealed)	Describes whether the life form is in a concealed position.	0 = Not Concealed 1 = Prepared Concealed Position
31	concealed_movement	Describes whether the life form uses concealment during movement.	0 = Open Movement 1 = Rushes Between Covered Positions

Table 34. DIS appearance and states — munition kind

Bit	Values	Description	State/ Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke or vapor is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	launch_flash_ present (launch_flash)	Describes the presence of a guided munitions launch flash.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen

Table 34. DIS appearance and states — munition kind (continued)

Bit	Values	Description	State/ Description
22	power_plant_on (power_plant_ status)	State of the entity's power plant.	0 = Off 1 = On
24-25	cover/shroud	Status of the cover or shroud.	0 = Closed 1 = Opening 2 = Blown/Detached 3 = Open/Attached
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 35. DIS appearance and states — environmental kind

Bit	Values	Description	State/ Description
16-19	density	Density of the environment (visibility).	0 = Clear 1 = Hazy 2 = Dense 3 = Very Dense 4 = Opaque 5 = Unused
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 36. DIS appearance and states — cultural feature kind

Bit	Values	Description	State/ Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5-6	is_smoke_ emanating (smoke)	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	internal_heat_on (internal_heat_ status)	Describes whether the internal heat is on or off.	0 = Off 1 = On
28	exterior_lights_ on (exterior_lights)	Describes whether the exterior lights are on or off.	0 = Off 1 = On
29	interior_lights_ on (interior_lights)	State of interior lights.	0 = Off 1 = On
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 37. DIS appearance and states — sensor kind

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
2	mission_killed (mission)	Ability to carry out mission.	0 = No 1 = Yes

Table 37. DIS appearance and states — sensor kind (continued)

Bit	Values	Description	State/ Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
12	lights_on (lights)	State of lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	antenna_raised (antenna)	Describes whether the antenna is raised or not	0 = Lowered 1 = Raised
17-18	camouflage_ type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused
19	concealed_ position (concealed)	Type of concealment.	0 = Not Concealed 1 = In Concealed Position
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_ on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On
24	tent_extended (tent)	State of a tent extension.	0 = Not Extended 1 = Extended

Table 37. DIS appearance and states — sensor kind (continued)

Bit	Values	Description	State/ Description
26	blackout_ lights_on (blackout_ lights)	State of blackout lights.	0 = Off 1 = On
29	interior_lights_ on (interior_ lights)	State of interior lights.	0 = Off 1 = On

Table 38. DIS appearance and states — supply kind

Bit	Values	Description	State/ Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
7-8	parachute_status (exp_parachute)	Status of a supply entity's parachute.	0 = None 1 = Deployed 2 = Collapsed 3 = Malfunction/ Streamer
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On
24-25	deployed_status	Describes the deployment status of a supply.	0 = Not Applicable 1 = Stowed 2 = Deployed 3 = Deployed and Active
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 39. DIS appearance and states — expendable kind

Bit	Values	Description	State/ Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	State of smoke.	0 = Off 1 = On
7-8	parachute_status (exp_parachute)	Status of an expendable entity's parachute.	0 = None 1 = Deployed 2 = Collapsed 3 = Malfunction/ Streamer
9-11	flare/smoke_ color (exp_color)	The color of a flare's light output or smoke emanating from a smoke expendable.	0 = White 1 = Red 2 = Green 3 = IR 4-7 = unspecified
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
17-18	flare/smoke_ status	Status of a flare or smoke expendable.	0 = Not Ignited 1 = Burning 2 = Burned Out
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On
24-25	spot_chaff_status	Describes the status of spot chaff.	0 = None 1 = Deployed 2 = Malfunction
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 40 lists the supported values for the **@dis articulated_part** keyword. String values are part of the WRM specification, but are not currently supported.

Table 40. DIS Articulations

Value	String	Description
1025	rudder	Rudder.
1056	left_flap	Left flap.
1088	right_flap	Right flap.
1120	left_aileron	Left aileron.
1152	right_aileron	Right aileron.
1184	helicopter_-_main_ rotor	Describes main rotor rotation.
1216	helicopter_-_tail_rotor	Describes tail rotor rotation.
1248	other_aircraft_control_ surfaces_defined_as_ needed	Other articulated parts.
2048	periscope	Periscope on subsurface watercraft.
2080	generic_antenna	Antenna.
2112	snorkel	Snorkel.
2144	other_extendible_ parts_ defined_as_needed	Other extendible parts.
3072	landing_gear	Landing gear on a plane.
3104	tail_hook	Tail hook on a plane for aircraft carrier landings.
3136	speed_brake	Speed brake for aircraft.
3168	left_door_of_primary _weapon_bay	Left door to weapon bay.
3200	right_door_of_primary _weapon_bay	Right door to the weapon bay.
3296	bridge_launcher	Bridge launcher.
3328	bridge_section_1	Bridge section 1.
3232	tank_or_apc_hatch	Hatch on a tank or APC.
3264	wingsweep	The angle at which an aircraft's wings are moved forward or back.
3360	bridge_section_2	Bridge section 2.
3392	bridge_section_3	Bridge section 3.
3424	primary_blade_1	Primary blade 1.
3456	primary_blade_2	Primary blade 2.
3488	primary_boom	Primary boom.

Table 40. DIS Articulations (continued)

Value	String	Description
3520	launcher_arm_primary	Primary launcher arm.
3552	other_fixed_position_parts_defined_as_needed	Other fixed position parts.
4096	primary_turret_number_1	Primary turret number 1.
4128	primary_turret_number_2	Primary turret number 2.
4160	primary_turret_number_3	Primary turret number 3.
4192	primary_turret_number_4	Primary turret number 4.
4224	primary_turret_number_5	Primary turret number 5.
4256	primary_turret_number_6	Primary turret number 6.
4288	primary_turret_number_7	Primary turret number 7.
4320	primary_turret_number_8	Primary turret number 8.
4352	primary_turret_number_9	Primary turret number 9.
4384	primary_turret_number_10	Primary turret number 10.
4416	primary_gun_number_1	Primary gun number 1.
4448	primary_gun_number_2	Primary gun number 2.
4480	primary_gun_number_3	Primary gun number 3.
4512	primary_gun_number_4	Primary gun number 4.
4544	primary_gun_number_5	Primary gun number 5.
4576	primary_gun_number_6	Primary gun number 6.
4608	primary_gun_number_7	Primary gun number 7.

Table 40. DIS Articulations (continued)

Value	String	Description
4640	primary_gun_number_8	Primary gun number 8.
4672	primary_gun_number_9	Primary gun number 9.
4704	primary_gun_number_10	Primary gun number 10.
4736	primary_launcher_1	Primary launcher 1.
4768	primary_launcher_2	Primary launcher 2.
4800	primary_launcher_3	Primary launcher 3.
4832	primary_launcher_4	Primary launcher 4.
4864	primary_launcher_5	Primary launcher 5.
4896	primary_launcher_6	Primary launcher 6.
4928	primary_launcher_7	Primary launcher 7.
4960	primary_launcher_8	Primary launcher 8.
4992	primary_launcher_9	Primary launcher 9.
5024	primary_launcher_10	Primary launcher 10.
5056	primary_defense_systems_1	Primary defense systems 1.
5088	primary_defense_systems_2	Primary defense systems 2.
5120	primary_defense_systems_3	Primary defense systems 3.
5152	primary_defense_systems_4	Primary defense systems 4.
5184	primary_defense_systems_5	Primary defense systems 5.
5216	primary_defense_systems_6	Primary defense systems 6.
5248	primary_defense_systems_7	Primary defense systems 7.
5280	primary_defense_systems_8	Primary defense systems 8.
5312	primary_defense_systems_9	Primary defense systems 9.
5344	primary_defense_systems_10	Primary defense systems 10.
5376	primary_radar_1	Primary radar 1.

Table 40. DIS Articulations (continued)

Value	String	Description
5408	primary_radar_2	Primary radar 2.
5440	primary_radar_3	Primary radar 3.
5472	primary_radar_4	Primary radar 4.
5504	primary_radar_5	Primary radar 5.
5536	primary_radar_6	Primary radar 6.
5568	primary_radar_7	Primary radar 7.
5600	primary_radar_8	Primary radar 8.
5632	primary_radar_9	Primary radar 9.
5664	primary_radar_10	Primary radar 10.
5696	secondary_turret_number_1	Secondary turret number 1.
5728	secondary_turret_number_2	Secondary turret number 2.
5760	secondary_turret_number_3	Secondary turret number 3.
5792	secondary_turret_number_4	Secondary turret number 4.
5824	secondary_turret_number_5	Secondary turret number 5.
5856	secondary_turret_number_6	Secondary turret number 6.
5888	secondary_turret_number_7	Secondary turret number 7.
5920	secondary_turret_number_8	Secondary turret number 8.
5952	secondary_turret_number_9	Secondary turret number 9.
5984	secondary_turret_number_10	Secondary turret number 10.
6016	secondary_gun_number_1	Secondary gun number 1.
6048	secondary_gun_number_2	Secondary gun number 2.
6080	secondary_gun_number_3	Secondary gun number 3.
6112	secondary_gun_number_4	Secondary gun number 4.

Table 40. DIS Articulations (continued)

Value	String	Description
6144	secondary_gun_ number_5	Secondary gun number 5.
6176	secondary_gun_ number_6	Secondary gun number 6.
6208	secondary_gun_ number_7	Secondary gun number 7.
6240	secondary_gun_ number_8	Secondary gun number 8.
6272	secondary_gun_ number_9	Secondary gun number 9.
6304	secondary_gun_ number_10	Secondary gun number 10.
6336	secondary_launcher_1	Secondary launcher 1.
6368	secondary_launcher_2	Secondary launcher 2.
6400	secondary_launcher_3	Secondary launcher 3.
6432	secondary_launcher_4	Secondary launcher 4.
6464	secondary_launcher_5	Secondary launcher 5.
6496	secondary_launcher_6	Secondary launcher 6.
6528	secondary_launcher_7	Secondary launcher 7.
6560	secondary_launcher_8	Secondary launcher 8.
6592	secondary_launcher_9	Secondary launcher 9.
6624	secondary_launcher_10	Secondary launcher 10.
6656	secondary_defense_ systems_1	Secondary defense systems 1.
6688	secondary_defense_ systems_2	Secondary defense systems 2.
6720	secondary_defense_ systems_3	Secondary defense systems 3.
6752	secondary_defense_ systems_4	Secondary defense systems 4.
6784	secondary_defense_ systems_5	Secondary defense systems 5.
6816	secondary_defense_ systems_6	Secondary defense systems 6.
6848	secondary_defense_ systems_7	Secondary defense systems 7.

Table 40. DIS Articulations (continued)

Value	String	Description
6880	secondary_defense_ systems_8	Secondary defense systems 8.
6912	secondary_defense_ systems_9	Secondary defense systems 9.
6944	secondary_defense_ systems_10	Secondary defense systems 10.
6976	secondary_radar_1	Secondary radar 1.
7008	secondary_radar_2	Secondary radar 2.
7040	secondary_radar_3	Secondary radar 3.
7072	secondary_radar_4	Secondary radar 4.
7104	secondary_radar_5	Secondary radar 5.
7136	secondary_radar_6	Secondary radar 6.
7168	secondary_radar_7	Secondary radar 7.
7200	secondary_radar_8	Secondary radar 8.
7232	secondary_radar_9	Secondary radar 9.
7264	secondary_radar_10	Secondary radar 10.
7296	deck_Elevator_#1	Deck elevator 1.
7328	deck_Elevator_#2	Deck elevator 2.
7360	catapult_#1	Catapult 1.
7392	catapult_#2	Catapult 2.
7424	jet_blast_deflector_#1	Jet blast deflector 1.
7456	jet_blast_deflector_#2	Jet blast deflector 2.
7488	arrestor_wires_#1	Arrestor wires 1.
7520	arrestor_wires_#2	Arrestor wires 2.
7552	arrestor_wires_#3	Arrestor wires 3.
7584	wing_(or rotor)_fold	Fold on plane's wings or helicoptor's rotors.
7616	fuselage_fold	Fold on plane's body.
32704	wind_direction	Parts that rotate into the wind.
32736	wind_blades	Parts that rotate based on the wind speed, such as turbine blades.
9120	1st_left_wheel	Left wheel 1.
9152	1st_right_wheel	Right wheel 1.
9184	2nd_left_wheel	Left wheel 2.
9216	2nd_right_wheel	Right wheel 2.

Table 40. DIS Articulations (continued)

Value	String	Description
9248	3rd_left_wheel	Left wheel 3.
9280	3rd_right_wheel	Right wheel 3.
9312	4th_left_wheel	Left wheel 4.
9344	4th_right_wheel	Right wheel 4.
9376	5th_left_wheel	Left wheel 5.
9408	5th_right_wheel	Right wheel 5.
9440	6th_left_wheel	Left wheel 6.
9472	6th_right_wheel	Right wheel 6.
9504	7th_left_wheel	Left wheel 7.
9536	7th_right_wheel	Right wheel 7.
9568	8th_left_wheel	Left wheel 8.
9600	8th_right_wheel	Right wheel 8.
9632	9th_left_wheel	Left wheel 9.
9664	9th_right_wheel	Right wheel 9.
9696	10th_left_wheel	Left wheel 10.
9728	10th_right_wheel	Right wheel 10.

Table 41 lists the values for the **@dis attachment_station** keyword. None of the values are currently supported.

Table 41. DIS Attachments

Value	Value/String	Description
	0	Nothing, Empty
	1-511	Sequential IDs for model-specific stations
	512-639	Fuselage Stations
	640-767	Left-wing Stations
	768-895	Right-wing Stations
	896	M16A42 rifle
	897	M249 SAW
	898	M60 Machine gun
	899	M203 Grenade Launcher
	900	M136 AT4
	901	M47 Dragon
	902	AAWS-M Javelin
	903	M18A1 Claymore Mine

Table 41. DIS Attachments (continued)

Value	Value/String	Description
	904	MK19 Grenade Launcher
	905	M2 Machine Gun
	906-1023	Other attached parts
	antenna_(1-10)	Describes the antenna is up or down
	bulldozer	Describes if there is a bulldozer attachment
	driver	Position of a driver
	gunner_location	Position of a gunner
	mine_plow	Mine clearing plow typically pushed by a tank
	mine_plow_roller	Mine clearing roller typically pushed by a tank
	passenger_(1-255)	Position of (a) passenger(s)
	reactive_armor	Explosive Armor on Tanks
	slat_armor	Defensive Armor on Tanks, APCs
	snow_plow	Describes if there is a snowblower attachment
	weapon_attached_(1-255)	Attached weapon or weapon system

Table 42 lists the values for the **@dis animation** keyword. String values are not currently supported.

Table 42. DIS Animations

Value	String)	Description
	animation	
	chassis	Motion of the chassis of a vehicle based on its direction of movement and acceleration using physics
	frame	
	muzzle_ flash	Describes all the aspects of
	recoil	Recoil of a gun after it has been fired
	rotation	
	track	
	turning	
	wheel rotation	



20. The MAK FBX Specification

This chapter describes the FBX specification developed and used by MAK to use FBX models in distributed simulations.

20.1. Introduction 330
20.1.1 Meshes (Geometry Objects), Nulls (Dummies, Groups or Locators) 330
20.1.2 Node Name and Comment Fields 330
20.2. Modeling for Simulation Interoperability 330
20.3. The Simulation Attribute Lexicon (SAL) 331
20.3.1 SAL Syntax 331
20.3.2 SAL Keywords and Values 331
20.4. Model Coordinate System 338
20.4.1 Orientation of the Model Coordinate System 338
20.4.2 Origin of the Model Coordinate System 339
20.5. Modeling Best Practices for Efficient Mapping 340
20.5.1 Comment the Parent Model Header 340
20.5.2 Attached Parts 341
20.5.3 Articulated Parts 343
20.5.4 Entity Appearance 345
20.5.5 Animation Sequences 346
20.6. Tagging the Location of Weapon Fire Special Effects 348
20.7. Encoding Simulation Details for Non-Entity Dynamic Objects 349
20.7.1 Specific Appearance Keyword Values used by MAK 349
20.7.2 Specific Articulation Keyword Values used by MAK 366
20.7.3 Shader-based Effects Maps in FBX 371
20.7.4 Using Texture Atlases, Metafiles, and Markings 371
20.7.5 Using Material-Classified (Sensor) Textures 371
20.8. Encoding Damage to Non-Entity Dynamic Objects 372
20.9. Mapping FBX Model Nodes to DIS Notes 372
20.9.1 Specifying the FBX Texture Path 373

20.1. Introduction

The Filmbox (FBX) interchange format, managed by Autodesk, is a widely used format for developing models in the 3D modeling industry. However, there is no FBX-specific standard for encoding the simulation-specific information needed by DIS and High-Level Architecture (HLA)-based distributed simulations.

This chapter describes a specification developed by MAK to:

- Provide support for FBX as a lower-cost, industry standard.
- Ensure compatibility with existing software and content.
- Be consistent with the WRM specification, so that it could also be made publicly available as a companion to the original WRM specification.

The MAK DBX specification parallels the WRM specification. In some cases, this specification goes beyond the original WRM specification to add capabilities that did not exist in the original WRM specification. These capabilities are also directly translatable to the WRM specification for use in OpenFlight.

The original WRM specification focused only on moving (Entity) models. However, there are many other objects in the simulation environment that can be controlled by the simulation, such as turning dynamic lights on and off, destroying buildings, and interacting with objects like gates, doors, windows, and so on. MAK uses the same approach as the WRM specification for encoding this information, both for FBX and OpenFlight.

20.1.1 Meshes (Geometry Objects), Nulls (Dummies, Groups or Locators)

FBX file format uses a straightforward system to organize geometry within the file, consisting of several types of nodes.

- Mesh nodes represent faces which are logically or geometrically related.
- Null nodes are empty nodes used as Switch, Lightpoint, Animation, Degrees of Freedon (DOF), Level of Detail (LOD), or External Reference nodes.

20.1.2 Node Name and Comment Fields

Each node may be given a name. Most nodes contain object properties (the comment is created within the User Defined Properties dialog box) for user-definable data. The user defined field is used to provide an implementation of the features described in this document.

20.2. Modeling for Simulation Interoperability

Techniques for modeling for a distributed simulation environment, whether it is DIS or HLA, are divided into three groups:

- Adoption of uniform standards for locating and orienting the model coordinate system.

- Techniques which assist in the mapping from well-defined enumeration codes to associated models and parts.
- General methods for eliminating individual support software for models.

These features will be implemented within the existing FBX file structure, primarily using the node comment fields.

20.3. The Simulation Attribute Lexicon (SAL)

This section describes the lexicon used to embed simulation-specific attributes in the FBX file format when creating a model for simulation interoperability.

20.3.1 SAL Syntax

The SAL syntax is simple and straightforward:

`@dis` *`keyword value`*

where:

- `@dis` precedes all other keywords and denotes the start of a simulation-specific directive.
- *`keyword`* is the specific simulation attribute being modeled.
- *`value`* is the specific attribute information - a single entry, a comma separated list, or a range of values (state keyword). (The `motion` keyword requires min and max values to follow the keyword value.)

20.3.2 SAL Keywords and Values

The `@dis` keyword derives from the original support for the DIS protocol, but now is a general term for any distributed interactive simulation environment that could be supported using DIS, HLA, TENA, or other distributed simulation protocols. Table 43 lists SAL keywords and describes them.

Table 43. SAL keywords and values

Keyword	Description	Values	Implementation Details
@dis	Main keyword that precedes all other keywords. It also denotes the termination of a free form comment.	Takes another SAL keyword as its value.	The keyword "@dis" is used at the beginning of each line in the comment field to signal the model loader of a feature specific to simulation modeling. Other key words and values follow.
Example: `@dis <keyword> <value>`			
model_spec_ version	Identifies the version level of this document used to create the models.	Takes the version number on the front cover as its value.	This keyword is placed in the parent model header as well as in any externally referenced models' headers. For more information, please see "20.5.1 Comment the Parent Model Header" on page 340.
Example: `@dis model_spec_version 001`			
entity_type	Identifies the model with a unique SISO Enumeration Entity code.	Takes the numeric values in SISO Enumerations, Section 4.2.3 Comprehensive Entity-Type tables separated by colons (:) as its value.	This keyword is placed in the parent model header as well as in any externally referenced models' headers. For more information, please see "20.5.1 Comment the Parent Model Header" on page 340.

Table 43. SAL keywords and values (continued)

Keyword	Description	Values	Implementation Details
Example: `@dis entity_type 1:1:225:3:1:1:0`			
#	Delineates the end of an @dis field and the start of a free form comment that continues until the next @dis keyword. Comment fields will be interpreted sensitive to case.	Takes user defined alphanumeric text as its value.	The pound sign "#" is used to stop the interpretation of an **@dis** field and allows a free-form comment to follow. Any new line not starting with **@dis** will be ignored and may be used for comments.
Example: `@dis entity_type 1:1:225:3:1:1:0 # M88A1 Armored Utility Vehicle, only armored vehicle in this exercise`			
attachment_ station	Identifies the location on the parent model where externally referenced child models may be placed via a unique DIS Attached Part code.	Takes a numeric value in SISO Enumeration, Section 4.7.1 Attached Parts as its value. It may optionally take the default specifier -1 as its value to indicate that any non exact match is to be attached to this location.	This keyword is placed in a Null node in the parent model. For more information, please see "Comment Attached Parts Stations in Primary Models" on page 341.

Table 43. SAL keywords and values (continued)

Keyword	Description	Values	Implementation Details
Example: `@dis attachment_station 512`			
attach_entity	Identifies the models that may be attached to an attachment_ station via a unique SISO Enumeration Entity code.	Takes the numeric values in SISO Enumeration, Section 4.2.3 Comprehensive Entity-Type tables separated by colons (:) as its value.	This keyword is placed in the same Null node as the attachment station in the parent model.For more information, please see "Comment Attached Parts Stations in Primary Models" on page 341.
Example: `@dis attach_entity 2:9:225:1:15:3:0 #Mk-84 bomb PAVEWAY II laser-guided`			
attachment_ point	Identifies the location where the externally referenced child model will be attached to the parent model's attachment stations via x, y, z offsets.	Takes the x, y, and z offsets from the model's origin that uniquely identifies the attachment point.	This keyword is placed in a Null node in the parent model if it is an attachable model. For more information, please see "Comment the Attachment Point in Attachable Models" on page 342.

Table 43. SAL keywords and values (continued)

Keyword	Description	Values	Implementation Details
Example: `@dis attachment_point 0 0 0.75`			
articulated_part	Identifies the moveable parts of a model with a unique SISO Enumeration Articulated Part code.	Takes the numeric value in SISO Enumeration, Section 4.7.2 Articulated Parts as its value.	This keyword is placed in a NULL or Mesh node that identifies an articulated part. This keyword is always followed by the motion keyword described below. For more information, please see "Embedding Simulation Articulation Data in the Model" on page 344.
Example: `@dis articulated_part 4096 # primary turret`			
motion	Identifies the type and range of motion for an articulated part.	Takes a value from Table 4.3.3.2 to identify the kind of motion followed by minimum and maximum range values.	This keyword is placed in a Group node that is a child to the NULL node that identifies an articulated part. For more information, please see "Embedding Simulation Articulation Data in the Model" on page 344.

Table 43. SAL keywords and values (continued)

Keyword	Description	Values	Implementation Details
Example: `@dis motion azimuth 0.0 360.0`			
switch	Identifies a model appearance attribute with a DIS keyword.	Takes the name value in SISO Enumeration, Section 4.3 Entity Appearance (General and Specific), joined by the underscore character (_) if the name is multiple words, as its value. Note that the following names (frozen_status, power_plant_status, and state) are preceded by an abbreviation for their domain as follows - gm_ for guided munitions and lf_ for life forms. Refer to Table 4.3.5.	This keyword is placed in a Null node in the parent model as well as in a Null node in any externally referenced models. For more information, please see "20.5.4 Entity Appearance" on page 345.
Example: `@dis switch hatch`			
state	Determines the actual state of the appearance attribute with a SISO Enumeration appearance state code.	Takes the numeric value defined in SISO Enumeration, Section 4.3 Entity Appearance (General and Specific) Purpose field, as its value. Refer to Table 4.3.5 and Section 4.3.5 for a description of a value list for this keyword.	This keyword is placed in the child of a Null node in the parent model as well as in a child of a Null node in any externally referenced models. For more information, please see "20.5.4 Entity Appearance" on page 345.
Example: `@dis state -1,1`			
inherit	Identifies an instance where texture, material, or LODs should be inherited from the parent model.	Takes one of the following reserved names as its value: LOD, material, or texture.	This keyword is placed in the state's comment node as appropriate.

Table 43. SAL keywords and values (continued)

Keyword	Description	Values	Implementation Details
Example: `@dis inherit texture`			
weapon_effect	Identifies the location of a weapon fire special effect.	Takes the numeric values in SISO Enumeration, Section 4.2.3.2 Munitions separated by colons (:) as its value. Note that more than one munition entity type maybe specified in a comma separated list.	This keyword is placed in a NULL node that identifies the special effect location. For more information, please see "20.6. Tagging the Location of Weapon Fire Special Effects" on page 348.
Example: `@dis weapon_effect 2:2:225:2:13:1:-1:-1 # Munition, Anti-Armor, US, Ballistic, 120 mm, any type of round`			
animation	Identifies that animation instructions follow.	Takes num_frames_per_ second, which is the rate at which the frames should be displayed and keyword skip or noskip to indicate whether the elapsed time should be used to choose the next frame, or whether at most one frame should be advanced each frame.	This keyword is placed in a node that contains an animation sequence. For more information, please see "20.5.5 Animation Sequences" on page 346.
Example: `@dis animation 2 skip`			
loopCount	Identifies the animation-specific number of times though the animation sequence.	Takes numeric value count, which is the number of times to loop through the animation. Default: -1 (loop forever). 0 means do not loop.	This keyword is placed in the same node as animation keyword. For more information, please see "20.5.5 Animation Sequences" on page 346.

Table 43. SAL keywords and values (continued)

Keyword	Description	Values	Implementation Details
Example: `@dis animation 2 skip @dis loopCount 3`			
lastFrameTime	Specifies how long, in seconds, to hold the last frame of the animation.	Takes numeric value time, which specifies how long, in seconds, to hold the last frame of the animation.	This keyword is placed in the same node as animation keyword. For more information, please see "20.5.5 Animation Sequences" on page 346.
Example: `@dis animation 2 skip @dis loopCount 3 @dis lastFrameTime 11.5`			

20.4. Model Coordinate System

This section describes the orientation and origin of the model coordinates systems used.

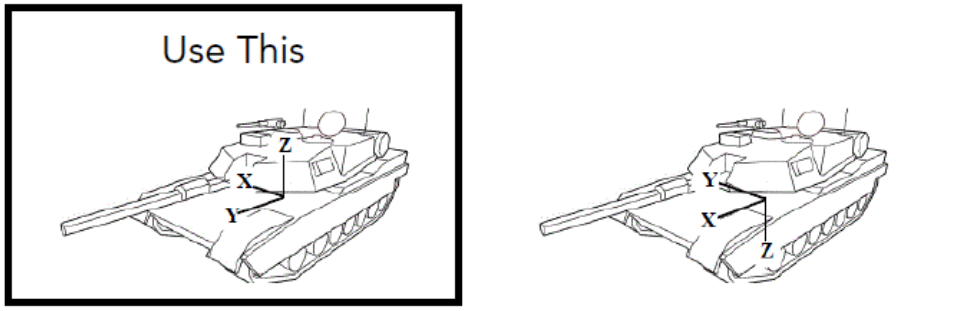
20.4.1 Orientation of the Model Coordinate System

The DIS standard uses a North, East, Down (NED) for X, Y, and Z (X forward, Y right, and Z down, but the WRM spec was initially designed for OpenFlight, which used an East, North, Up (ENU) for X, Y, and Z (X right, Y forward, and Z up).

The WRM standard chose an intentional, specific deviation from the DIS standard intended to make a modeler's use of OpenFlight (and this also applies to FBX) intuitive with respect to a model's coordinate system:

Construct models using the East, North, Up (ENU) coordinate orientations (X right, Y forward, and Z up), with a one (1) meter unit length.

Figure 117. Model coordinate systems



East, North Up (ENU) Coordinate System

North, East Down (NED) Coordinate System

The real-time software performs the translation:

- xmodel = yDIS
- ymodel = xDIS
- zmodel = -zDIS

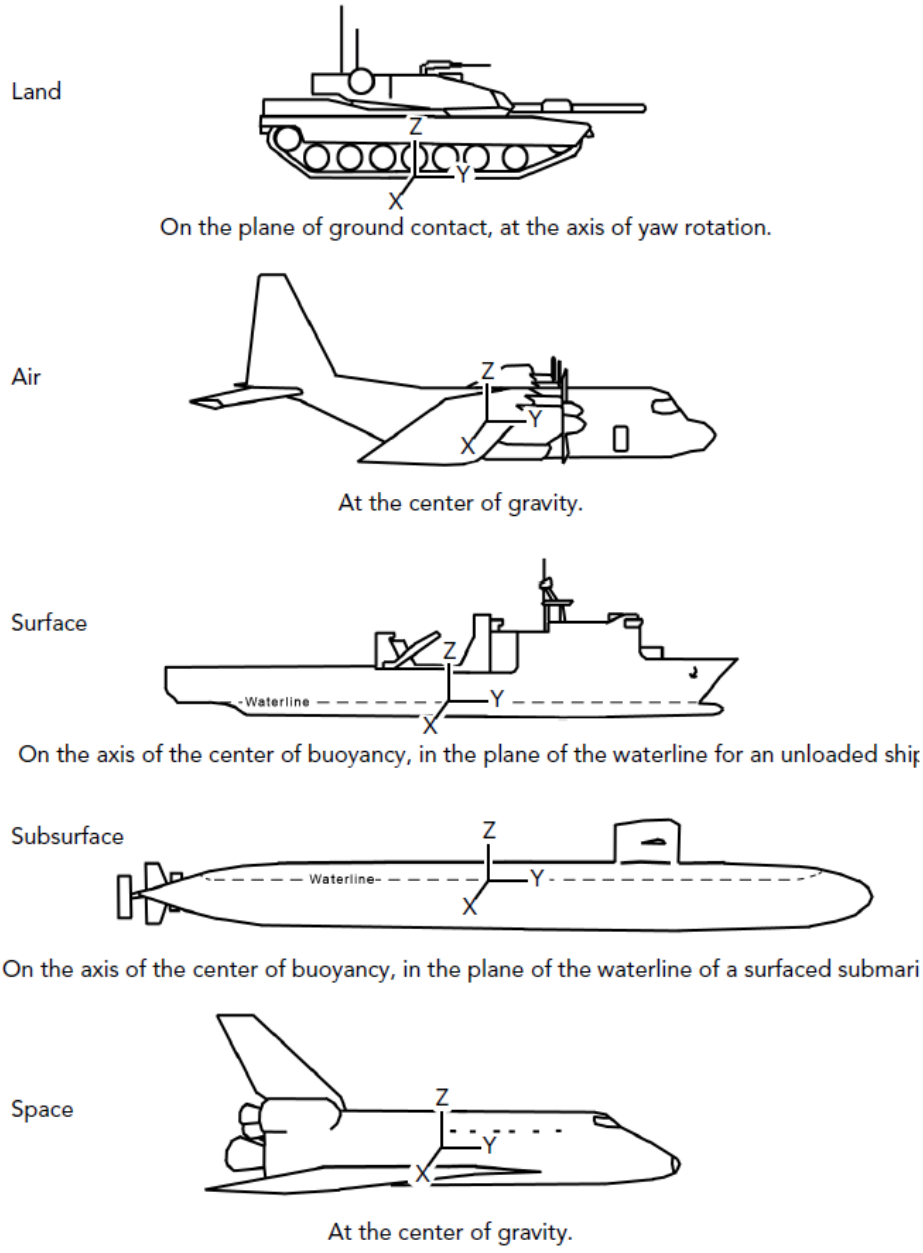
20.4.2 Origin of the Model Coordinate System

Table 44 describes the location of the model origin by domain. Figure 118 illustrates the locations of model origins.

Table 44. Location of model origin by domain

Domain	Location of model origin
Land	On the plane of ground contact, at the axis of yaw rotation.
Air	At the center of gravity. The center of gravity is the point at which weight is evenly dispersed and all sides are in balance and is the point from which the weight of a body or system may be considered to act.
Surface	On the axis of the center of buoyancy, in the plane of the waterline for an unloaded ship.
Subsurface	On the axis of the center of buoyancy, in the plane of the waterline of a surfaced submarine.
Space	At the center of gravity.

Figure 118. Location of origin by domain



20.5. Modeling Best Practices for Efficient Mapping

This section describes how to map models efficiently.

20.5.1 Comment the Parent Model Header



This is not directly used by VR-Vantage but can provide guidance to other possible editors of the model.

The parent model is the base geometry without any external references. The parent model header is comprised of two sequential comments. The first comment is:

```
@dis model_spec_version value
```

where *value* is the version number of this document to which the model was designed.

The second comment is:

```
@dis entity_type code
```

where *code* is found in the SISO Enumerations, Section 4.2.3 Comprehensive Entity-Type tables. Each field in the entity type is separated by a colon. Entity is the DIS terminology for a model.

For example, comment the parent model header of an M88A1 as follows:

```
@dis model_spec_version 001
@dis entity_type 1:1:225:3:1:1:0
```

The entity type is derived from the SISO Enumerations for Simulation Interoperability: For example, the fields in the entity type in the previous example have the following meanings:

- Entity kind (1) = Platform
- Domain (1) = Land
- Country (225) = USA
- Category (3) = Armored utility vehicle
- Sub-category (1) = M88 Medium Recovery Vehicle
- Specific (1) = M88A1
- Extra (0) = other.

20.5.2 Attached Parts



VR-Vantage does not currently support attached parts.

In a simulation environment, an entity may have one or more attached parts. Attached parts are separate entities that are "attached" to another entity. For instance, a missile launcher may have an attached missile prior to launch, an aircraft may have bombs attached prior to its bombing run, and so on.

Comment Attached Parts Stations in Primary Models

An attached part is placed at a particular attachment station on the primary entity. One may think of an attachment station as a hanger, from which an attached part may be hung.

When modeling an entity which may have other entities attached to it, (e.g. a missile launcher), one must indicate where the attachment stations are in the entity's geometry, and at what orientation the attached part model should be placed.

Stations are numbered sequentially starting with one. The order of station numbering is from top to bottom, then back to front, then left to right, except for aircraft wing stations, which are numbered from inboard to outboard. The attached part stations are defined in SISO Enumerations, Section 4.7.1 Attached Parts.

Attachment station information is encoded in the primary model as follows:

1. Place a Null node in the primary model for each attachment point.
2. Create a minimal polygon at 0, 0, 0.
3. Translate this polygon to the attachment station's x, y, z offset from the origin of the model.
4. Rotate the polygon in the yaw, pitch, and roll axes such that if any geometry were to be placed below this Null node, it would have the proper orientation for the attached part.
5. Comment the Null node in the primary model for each attachment point as:

   `@dis attachment_station` *`value list`*

   where *`value`* list is of the form {number[, number]}, where number is the four-digit code found in the SISO Enumerations, Section 4.7.1 Attached Parts for the station you are identifying. If the optional default specifier -1 follows an `attachment_station` number, it will be used as the attachment station for any attached part that does not have an exact match.

For example, comment the Null node for the fuselage centerline attachment point as follows:

```
@dis attachment_station 512, -1 # fuselage centerline station; also
use for anything else that doesn't match
```

For a missile launcher, the point on the launcher's geometry where a missile can be attached has a Null node with the following comment:

```
@dis attachment_station 1 # station 1
```

A comment of the form:

`@dis attach_entity` *`code`*

where *`code`* is found in the SISO Enumerations, Section 4.2.3 Comprehensive Entity-Type tables, is placed below the `@dis attachment_station` comment in the Null node. Additional codes may be placed in a comma separated list. Each field in the entity type is separated by a colon. For the aircraft example, comment the Null node:

```
@dis attach_entity 2:9:225:1:15:3:0 #Mk-84 bomb PAVEWAY II laser-
guided
```

For the missile launcher:

```
@dis attach_entity 2:1:225:1:15:1:0, 2:1:225:1:16:0:0 #FIM-92 Stinger
A/B/C or MIM-104 Patriot
```

Comment the Attachment Point in Attachable Models

When modeling an entity that can be attached to another entity, you must indicate what point on its geometry should coincide with the location of the attachment station on the entity to which this entity may attach. Think of this point as the point to be

hung from the hanger (attachment station) defined in "Comment Attached Parts Stations in Primary Models" on page 341.

The attached part will by default be oriented parallel to the main model's longitudinal (y) axis. If any rotation or transformation is required to attach the attaching point to the attachment point station, it will be done by the primary model.

The location of the attachment point is specified by a comment in the root node of the model in a line following the `@dis entity_type ...` comment of the following format:

```
@dis attachment_point x y z
```

where x, y, and z are the offsets from the model's origin that uniquely defines the attachment point. For the aircraft example, comment the root node in the bomb model:

```
@dis attachment_point 0 0 0.75 # bomb attachment point x = 0m, y = 0m, z = 3/4m
```

For the missile launcher, comment the root node in the missile model:

```
@dis attachment_point 0 0 -0.5 #missile attaching point
```

This indicates that when this missile type is attached to a primary model, the point of attachment is half a meter below the entity's origin.

20.5.3 Articulated Parts

In order for VR-Vantage to properly depict the state of articulated parts, models must conform to the following specifications:

- There must be one Null node for each articulated part.
- Each articulated part Null node must have all the geometry and other nodes associated with the part underneath it. This Null 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, a turret's Null node must be commented with:

  ```
  @dis articulated_part 4096
  ```

  A gun's Null node 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.
- Do not include static offsets in the Null node attributes. That is, when you set all the Null node parameters (X, Y, Z, heading, pitch, roll) to 0, the model should appear in its neutral position.

Embedding Simulation Articulation Data in the Model

The SISO Enumeration specific model features are identified by comments in the Null node as follows:

The articulated part's Group node comment field may consist of several lines, each starting with the `@dis` keyword.

- The first line is an Articulated Part Type Specification.
- The additional lines are Articulated Part Motion Type Specifications, each assigned to a single line.
- There must be an Articulated Part Motion Type Specification for each motion that is allowable for the articulated part.

The SISO Enumerations Articulated Part Type Specification takes the form `@dis articulated_part` *`type`*, where *`type`* is the four-digit enumeration for the Articulated part. Refer to SISO Enumerations, Section 4.7.2 Articulated Parts for type values.

For example, comment the Null node for the turret of an M1 tank:

```
@dis articulated_part 4096 # primary turret
```

A DIS Articulated Part Motion Type Specification takes the form "@dis motion kind min max", where kind is the type of motion allowed for an articulated part (see Table 45); min is the minimum position/orientation deviation from the neutral position/orientation; and max is the maximum position/orientation deviation from the neutral position/orientation. The min/max constraints on translational parameters are in units of meters, and they are in units of degrees for rotational parameters. Refer to the following examples:

- Example 1:

```
  @dis articulated_part 4096 # primary turret M1 tank
  "@dis motion azimuth 0.0 360.0 # primary turret can only rotate
in azimuth;# within the range of 0 to 360 degrees
```

- Example 2:

```
  @dis articulated_part 4416 # primary gun M1 tank
  @dis motion elevation -30.0 90.0 # gun may change its elevation
within a range of -30 to +90 degrees only
  @dis motion Y -0.6 0.3 # gun may change its position along the
  # Y axis for recoil within a range of -0.6 to +0.3 meters only
```

- Example 3:

```
  @dis articulated_part 3072 # left main landing gear F15E
  @dis motion Y 0.0 1.0 # gear may change its Y, X, and Z
positions
  @dis motion X -1.0 1.0 # with the indicated constraints
  @dis motion Z 0.0 2.0 # in meters
```

Table 45 lists the kind values for articulated parts.

Table 45. Simulation Articulated Part Motion Type Specification Kind Values

Kind	Value	DIS cross reference Description
Y	DIS X	Forward / Backward motion (Model Y axis).
X	DIS Y	Left / Right motion (Model X axis).
Z	-DIS Z	Up / Down motion (Model Z axis).
Azimuth	-Azimuth	Heading (rotation with respect to Model Z axis).
Elevation	Elevation	Pitch (rotation with respect to Model Y axis).
Rotation	Rotation	Roll (rotation with respect to Model X axis).
Extension	DIS X	Left / Right motion (Model Y axis)



The DIS coordinate system differs from the model coordinate system such that in DIS, x is forward, y is right, and z is down.

20.5.4 Entity Appearance

VR-Vantage supports switching between different pieces of a model's geometry based on appearance attributes. For instance, a tank's turret should look different if the hatch is open or closed. An entire tank should look different if it is damaged or destroyed.

Switch nodes are used to represent a choice between several different children, based on a particular appearance attribute.

Define a switch node as a Null node with a comment of the form:

```
@dis switch attribute
```

where attribute is the name of the attribute that should be used to select a child. (For example, damage, smoke, hatch, and so on.)

The children of the switch node represent the different portions of a model among which VR-Vantage will select, based on the appearance attribute. If a switch node is commented with:

```
@dis switch hatch
```

It may, for example, have three children containing geometry for a closed hatch, popped hatch, and open hatch.

These children of the switch node must be commented so that VR-Vantage knows which child matches which values of the appearance attribute. The comment is of the form:

```
@dis state value-list
```

value-list is of the form *m[,m]+* where *m* is one of the following:

- An integer representing the enumeration of a value that an appearance attribute may take on, for example, 1 for hatch closed.

- Two integers, a and b, separated by a hyphen, representing all enumerations from a through b, for example, 2-5 for all states whose enumerations are 2, 3, 4, or 5.
- The integer -1, which indicates that this child is the default child, to be used when the attribute state matches no values in any children.

For instance, the child of the hatch switch node that represents a closed hatch might be used when the hatch attribute value is either 0 (not-applicable), 1 (closed), or any value other than those enumerated in other children. This child would be commented:

```
@dis state 0,1,-1
```

The child with the popped hatch geometry should be used when the value is 2 (popped) or 3 (popped with person visible) and is commented:

```
@dis state 2,3
```

The child that contains the open hatch geometry should be used when the value is 4 (open) or 5 (open with person visible) and is commented:

```
@dis state 4,5
```

There may be many switch nodes for the same attribute. For example, there may be several pieces of a tank's geometry that are affected by the destroyed attribute.

The WRM Model Specification also discusses the inheritance of appearance attributes from ancestor nodes. VR-Vantage does not support this functionality. To tell VR-Vantage to, for example, display a camouflaged tank when the paint scheme attribute is set to camouflage, and a normal tank otherwise, you must use the switch node method described previously.

In Figure 119 on page 348, the turret Null node is attached to the body Null node. The barrel, gun, and hatch Null nodes are attached to the turret. This allows the turret to be positioned in the coordinate system of the body, and the barrel, gun, and hatch to be positioned in the coordinate system of the turret.

20.5.5 Animation Sequences

MAK extended the original WRM specification to support visualizing animation sequences defined in the model.

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

The comment syntax is as follows:

```
@dis animation num_frames_per_second [ skip | noskip ]
```

where:

- `animation` tells the visualization system that animation instructions follow.
- *`num_frames_per_second`* is the rate at which the visualization system should display the frames.
- `skip` and `noskip` indicate whether the elapsed time should be used to choose the next frame, or whether at most one frame should be advanced each frame.

You can also specify how many times to loop through an animation and how long to hold the last frame with the following comments:

`@dis loopCount` *`count`*

where *`count`* is the number of times to loop through the animation. Default: -1 (loop forever). 0 means do not loop.

`@dis lastFrameTime` *`time`*

where *`time`* specifies how long, in seconds, to hold the last frame of the animation.

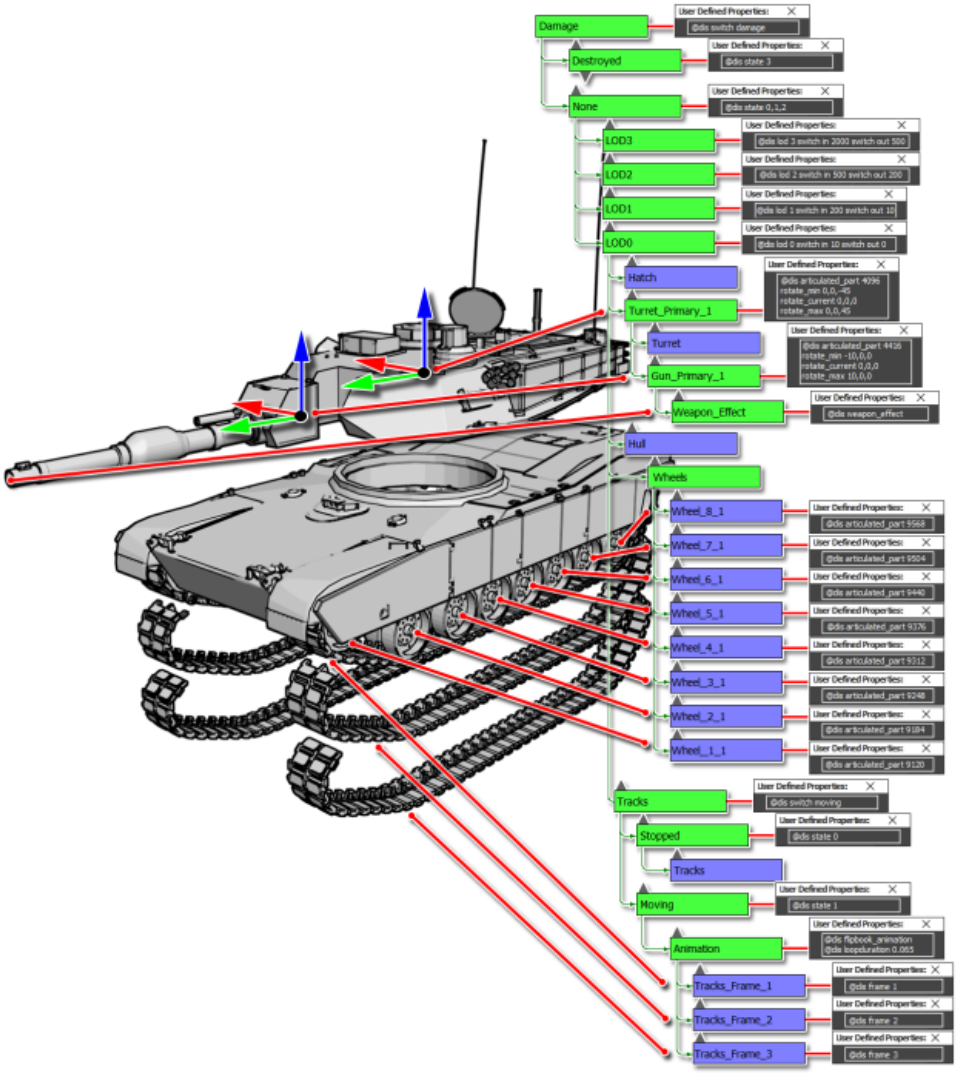
The `loopCount` and `lastFrameTime` comments only work with non-instanced models. You must use them with an animation comment, for example:

```
@dis animation 2 skip @dis loopCount 3 @dis lastFrameTime 11.5
```

The model in Figure 119 also contains a switch node that is flagged with animation. The comment on this node is `@dis switch moving`. This switches between two Null nodes – Stopped and Moving. Under the Stopped node there is a single mesh of the treads of the tank. Under the Moving node there is a Null node, with the animation comment. This creates a flipbook animation of all nodes below this Group node. There are three meshes in different states of moving: Frame 1, Frame 2 and Frame 3. At runtime VR-Vantage animates these meshes when the velocity is not 0. If multiple color schemes are needed to represent camouflage, winter, and desert versions, another switch is needed.

Figure 119 illustrates a 3D model structure.

Figure 119. 3D model structure



20.6. Tagging the Location of Weapon Fire Special Effects

> VR-Vantage does not currently support weapon effects using this approach.

Place a Null node at the location of the weapon fire special effect. Comment the Null node @dis weapon_effect that is a descendant of the Null node `@dis weapon_effect` *`code list`*, where *`code list`* consists of the SISO Enumerations munition IDs fired from this location.

For example:

- Place a NULL node at the end of the 50 caliber machine gun barrel on the turret of an M1A2, and label the descendant Null node:

```
@dis weapon_effect 2:2:225:2:1:-1:-1, 2:8:225:2:5:-1,-1 # Munition, Anti-Armor, US, Ballistic, 50 cal, any type of round; also Munition, Anti-Personnel, US, Ballistic, 50 cal, any type of round
```

- Place a Null node at the end of the main gun barrel, and label the node `@dis weapon_effect`. Label the descendant Null node

```
@dis weapon_effect 2:2:225:2:13:1:-1:-1 # Munition, Anti-Armor, US, Ballistic, 120 mm, any type of round
```

20.7. Encoding Simulation Details for Non-Entity Dynamic Objects

In some cases, MAK has made extensions to the original WRM spec. This might include additional keywords not found in (or modifications to keywords found in) the SISO Enumerations. In other cases, the WRM spec did not cover the specifics of the feature or the comment field-based approach was not necessarily needed to support the feature.

20.7.1 Specific Appearance Keyword Values used by MAK

The tables below list the values that you can use for the `@dis switch` keyword by kind and domain. For any `switch` keyword, the supported state values (`@dis state [value 1],[value 1]...`) are listed in the State column. The Bit column lists the appearance bit for setting DIS appearances. (Some bit numbers are repeated because they are valid only for a particular platform.)

The specific tables are:

- Table 46. DIS appearance and states — platform kind, land domain on the next page
- Table 47. DIS appearance and states — platform kind, air domain on page 352
- Table 48. DIS appearance and states — platform kind, surface domain on page 355
- Table 49. DIS appearance and states — platform kind, subsurface domain on page 356
- Table 50. DIS appearance and states — platform kind, space domain on page 357
- Table 51. DIS appearance and states — lifeform kind on page 357
- Table 52. DIS appearance and states — munition kind on page 360
- Table 53. DIS appearance and states — environmental kind on page 361
- Table 54. DIS appearance and states — cultural feature kind on page 362
- Table 55. DIS appearance and states — sensor kind on page 362
- Table 56. DIS appearance and states — supply kind on page 364
- Table 57. DIS appearance and states — expendable kind on page 365

In general, the keywords mostly match the SISO enumerations Entity Appearance Name values but are lower case and have replaced " " and "/" with a "_" between words. In some cases, the keywords have been shortened for easier readability although the underscore-separated versions should also work. For example, SISO Enumeration Paint Scheme becomes paint_scheme.

Table 46 lists DIS appearance and states for the land domain.

Table 46. DIS appearance and states — platform kind, land domain

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
2	fire_power_killed fire_power	Characteristics of firepower kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	trailing_dust_cloud (trailing_effects)	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
9-11	primary_hatch (hatch)	State of the primary hatch.	0 = Not Applicable 1 = Closed 2 = Popped 3 = Popped and Person Visible 4 = Open 5 = Open and Person Visible 6 = Custom1 7 = Custom2
12	head_lights_on (head_lights)	State of headlights.	0 = No 1 = Yes

Table 46. DIS appearance and states — platform kind, land domain (continued)

Bit	Values	Description	State/Description
13	tail_lights_on (tail_lights)	State of tail lights.	0 = Off 1 = On
14	brake_lights_on brake_lights	State of brake lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	launcher/operational (launcher)	State of the platform's primary launcher.	0 = Not Raised 1 = Raised
17-18	camouflage_type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused
19	concealed_position concealed	Type of concealment.	0 = Not Concealed 1 = In Concealed Position
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_status)	State of the entity’s power-plant.	0 = Off 1 = On
24	tent_extended (tent)	State of a tent extension.	0 = Not Extended 1 = Extended
25	ramp_extended (ramp)	State of a ramp.	0 = Down 1 = Up
26	blackout_lights_on blackout_lights	State of blackout lights.	0 = Off 1 = On
27	blackout_brake_lights _on (blackout_brake_ lights)	State of blackout brake lights.	0 = Off 1 = On
28	spot/search_light _#1_on (spot_lights)	Describes whether spotlights are on or off.	0 = Off 1 = On
29	interior_lights _(forward)_on (interior_lights)	State of interior lights.	0 = Off 1 = On

Table 46. DIS appearance and states — platform kind, land domain (continued)

Bit	Values	Description	State/Description
30	occupants_ surrendered	Describes whether the vehicle's occupants have surrendered.	0 = No 1 = Yes
31	masked/cloaked (masked_cloaked)	Masked or cloaked state.	0 = No 1 = Yes

Table 47 lists DIS appearance and states for the air domain.

Table 47. DIS appearance and states — platform kind, air domain

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	propulsion_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
2	nvg_mode	Describes whether air platform lighting is in covert or overt mode.	0 = Overt Lighting 1 = Covert Lighting (Night Vision Goggles)
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large

Table 47. DIS appearance and states — platform kind, air domain (continued)

Bit	Values	Description	State/Description
9-11	canopy/troop_door (hatch)	Describes the state of the canopy or troop door.	0 = Not Applicable 1 = Single Canopy/Single Troop Door Closed 2 = Front and Rear Canopy/Left and Right Troop Door Closed 3 = Front Canopy/Left Troop Door Open 4 = Single Canopy/Single Troop Door Open 5 = Rear Canopy/Right Troop Door Open 6 = Front and Rear Canopy/Left and Right Troop Door Open
12	landing_lights_on (landing_lights)	State of landing lights.	0 = Off 1 = On
13	navigation_lights_on (navigation_lights)	State of navigation lights.	0 = Off 1 = On
14	anti-collision_lights_on (anticollision_lights_on) (anticollision_lights)	State of anticollision lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	afterburner_on (afterburner)	State of an air platform's afterburner.	0 = Off 1 = On
17	lower_anti-collision_lights_on (lower_anti_collision_lights_on)	Describes whether the lower anticollision lights are on or off.	0 = Off 1 = On
18	upper_anti-collision_lights_on (upper_anti_collision_lights_on)	Describes whether the upper anticollision lights are on or off.	0 = Off 1 = On

Table 47. DIS appearance and states — platform kind, air domain (continued)

Bit	Values	Description	State/Description
19	anti-collision_lights_ day/night (anti_collision_ lights_day/night)	Day/night status of the anticollision lights.	0 = Day 1 = Night
20	is_blinking	Indicates whether any lights are blinking.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On
23	state	Describes whether the entity is active.	0 = Deactivated 1 = Active
24	formation_lights	State of formation lights.	0 = Off 1 = On
25	landing_gear _extended	Describes whether the landing gear is wholly retracted or extended.	0 = Retracted 1 = Extended
26	cargo_doors_opened	Describes whether the cargo doors (main door) are closed or open.	0 = Closed 1 = Open
27	navigation/position _brightness	Describes the brightness of the navigation/position lights.	0 = Dim 1 = Bright
28	spot/search_light_#1 _on (spot_lights)	Describes whether spotlights are on or off.	0 = Off 1 = On
29	interior_lights_on (interior_lights)	State of interior lights.	0 = Off 1 = On
30	reverse_thrust _engaged	Describes whether the air platform has engaged reverse thrust.	0 = No 1 = Yes
31	weight-on-wheels	Describes whether the air platform has weight on its main landing gear.	0 = No 1 = Yes

Table 48 lists DIS appearance and states for the surface domain.

Table 48. DIS appearance and states — platform kind, surface domain

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	wake_size (trailing_ effects)	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
12	running_lights_ on (running_lights)	State of running lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On
23	state	Describes whether the entity is active.	0 = Deactivated 1 = Active
28	spot_lights_on (spot_lights)	Describes whether spotlights are on or off.	0 = Off 1 = On
29	interior_lights_ on (interior_lights)	State of interior lights.	0 = Off 1 = On

Table 49 lists DIS appearance and states for the subsurface domain.

Table 49. DIS appearance and states — platform kind, subsurface domain

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_ killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
9-11	hatch	State of the primary hatch.	0 = Not Applicable 1 = Closed 2 = Popped 3 = Popped and Person Visible 4 = Open 5 = Open and Person Visible 6 = Custom1 7 = Custom2
12	running_ lights_on (running_ lights)	State of running lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_ status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_ on (power_plant_ status)	State of the entity’s power-plant.	0 = Off 1 = On

Table 50 lists DIS appearance and states for the space domain.

Table 50. DIS appearance and states — platform kind, space domain

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke or vapor is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_ status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_ on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On

Table 51 lists DIS appearance and states for the lifeform kind.

Table 51. DIS appearance and states — lifeform kind

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed

Table 51. DIS appearance and states — lifeform kind (continued)

Bit	Values	Description	State/Description
5-8	compliance_status (compliance)	Compliance of lifeform.	0 = Lifeform None 1 = Detained 2 = Surrender 3 = Using Fists 4 = Verbal Abuse 1 5 = Verbal Abuse 2 6 = Verbal Abuse 3 7 = Passive Resistance 1 8 = Passive Resistance 2 9 = Passive Resistance 3 10 = Non-Lethal Weapon 1 11 = Non-Lethal Weapon 2 12 = Non-Lethal Weapon 3 13 = Non-Lethal Weapon 4 14 = Non-Lethal Weapon 5 15 = Non-Lethal Weapon 6
11	signal_smoke_in_use	Describes whether signal smoke is being used.	0 = No 1= Yes
12	flash_lights_on (flashlight)	State of lights.	0 = Off 1 = On
13	signal_mirror_in_use	Describes whether a signal mirror is being used.	0 = No 1= Yes
14	ir_stobe_on	Describes whether an IR strobe is on or off.	0 = Off 1 = On
15	ir_illuminator_on	Describes whether an IR illuminator (flare) is on or off.	0 = Off 1 = On

Table 51. DIS appearance and states — lifeform kind (continued)

Bit	Values	Description	State/Description
16-19	life_form_posture (life_form_state)	Lifeform action.	0 = None 1 = Upright Still 2 = Upright Walking 3 = Upright Running 4 = Kneeling 5 = Prone 6 = Crawling 7 = Swimming 8 = Parachuting 9 = Jumping 10 = Sitting 11 = Squatting 12 = Crouching 13 = Wading 14 = Surrender 15 = Detained
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	mounted/hoisted _status	Describes whether the life form is mounted or hoisted on a platform such as a troop transport or helicopter hoist.	0 = Not Mounted/Hoisted 1 = Mounted/Hoisted
24-25	weapon/implement_1 (weapon_1)	State of primary weapon.	0 = None 1 = Stowed 2 = Deployed 3 = Firing Position
26-27	weapon/implement_2 (weapon_2)	State of secondary weapon.	0 = None 1 = Stowed 2 = Deployed 3 = Firing Position
28-29	camouflage_type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused

Table 51. DIS appearance and states — lifeform kind (continued)

Bit	Values	Description	State/Description
30	concealed_stationary (concealed)	Describes whether the life form is in a concealed position.	0 = Not Concealed 1 = Prepared Concealed Position
31	concealed_movement	Describes whether the life form uses concealment during movement.	0 = Open Movement 1 = Rushes Between Covered Positions

Table 52 lists DIS appearance and states for the munition kind.

Table 52. DIS appearance and states — munition kind

Bit	Values	Description	State/Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke or vapor is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	trailing_effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	launch_flash_ present (launch_flash)	Describes the presence of a guided munitions launch flash.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On

Table 52. DIS appearance and states — munition kind (continued)

Bit	Values	Description	State/Description
24-25	cover/shroud	Status of the cover or shroud.	0 = Closed 1 = Opening 2 = Blown/Detached 3 = Open/Attached
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 53 lists DIS appearance and states for the environmental kind.

Table 53. DIS appearance and states — environmental kind

Bit	Values	Description	State/ Description
16-19	density	Density of the environment (visibility).	0 = Clear 1 = Hazy 2 = Dense 3 = Very Dense 4 = Opaque 5 = Unused
21	is_frozen (frozen_status)	Frozen status of an entity.	2 = Not Frozen 3 = Frozen
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 54 lists DIS appearance and states for the cultural feature kind.

Table 54. DIS appearance and states — cultural feature kind

Bit	Values	Description	State/Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5-6	is_smoke_ emanating (smoke)	State or location of smoke and vapor.	0 = None 1 = Plume 2 = Engine 3 = Engine and Plume
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	internal_heat_on (internal_heat_ status)	Describes whether the internal heat is on or off.	0 = Off 1 = On
28	exterior_lights_ on (exterior_lights)	Describes whether the exterior lights are on or off.	0 = Off 1 = On
29	interior_lights_ on (interior_lights)	State of interior lights.	0 = Off 1 = On
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 55 lists DIS appearance and states for the sensor kind.

Table 55. DIS appearance and states — sensor kind

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
1	mobility_ killed (mobility)	Characteristics of mobility kill.	0 = No 1 = Yes
2	mission_ killed (mission)	Ability to carry out mission.	0 = No 1 = Yes

Table 55. DIS appearance and states — sensor kind (continued)

Bit	Values	Description	State/Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_ emanating (smoke)	Describes whether smoke or vapor is emanating from the entity.	0 = No 1 = Yes
6	is_engine_ emitting _smoke (smoke)	Describes whether the engine is emitting smoke.	0 = No 1 = Yes
7-8	trailing_ effects	Size of the trailing effect for the entity.	0 = None 1 = Small 2 = Medium 3 = Large
12	lights_on (lights)	State of lights.	0 = Off 1 = On
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
16	antenna_ raised (antenna)	Describes whether the antenna is raised or not	0 = Lowered 1 = Raised
17-18	camouflage_ type	Type of camouflage.	0 = Desert 1 = Winter 2 = Forest 3 = Unused
19	concealed_ position (concealed)	Type of concealment.	0 = Not Concealed 1 = In Concealed Position
21	is_frozen (frozen_ status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_ on (power_ plant_status)	State of the entity’s power-plant.	0 = Off 1 = On

Table 55. DIS appearance and states — sensor kind (continued)

Bit	Values	Description	State/Description
24	tent_extended (tent)	State of a tent extension.	0 = Not Extended 1 = Extended
26	blackout_ lights_on (blackout_ lights)	State of blackout lights.	0 = Off 1 = On
29	interior_ lights_on (interior_ lights)	State of interior lights.	0 = Off 1 = On

Table 56 lists DIS appearance and states for the supply kind.

Table 56. DIS appearance and states — supply kind

Bit	Values	Description	State/Description
0	paint_scheme	Paint scheme of an entity.	0 = Uniform 1 = Camouflage
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
7-8	parachute_status (exp_parachute)	Status of a supply entity's parachute.	0 = None 1 = Deployed 2 = Collapsed 3 = Malfunction/Streamer
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On

Table 56. DIS appearance and states — supply kind (continued)

Bit	Values	Description	State/Description
24-25	deployed_status	Describes the deployment status of a supply.	0 = Not Applicable 1 = Stowed 2 = Deployed 3 = Deployed and Active
31	masked/cloaked (masked_cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

Table 57 lists DIS appearance and states for the expendable kind.

Table 57. DIS appearance and states — expendable kind

Bit	Values	Description	State/Description
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed
5	is_smoke_emanating (smoke)	State of smoke.	0 = Off 1 = On
7-8	parachute_status (exp_parachute)	Status of an expendable entity's parachute.	0 = None 1 = Deployed 2 = Collapsed 3 = Malfunction/Streamer
9-11	flare/smoke_color (exp_color)	The color of a flare's light output or smoke emanating from a smoke expendable.	0 = White 1 = Red 2 = Green 3 = IR 4-7 = unspecified
15	is_flaming (flaming)	State of flames rising from an entity.	0 = No 1 = Yes
17-18	flare/smoke_status	Status of a flare or smoke expendable.	0 = Not Ignited 1 = Burning 2 = Burned Out
21	is_frozen (frozen_status)	Frozen status of an entity.	0 = Not Frozen 1 = Frozen

Table 57. DIS appearance and states — expendable kind (continued)

Bit	Values	Description	State/Description
22	power_plant_on (power_plant_ status)	State of the entity's power-plant.	0 = Off 1 = On
24-25	spot_chaff_status	Describes the status of spot chaff.	0 = None 1 = Deployed 2 = Malfunction
31	masked/cloaked (masked_ cloaked)	Describes whether the entity is masked or cloaked.	0 = No 1 = Yes

20.7.2 Specific Articulation Keyword Values used by MAK

Table 58 lists the supported values for the `@dis articulated_part` keyword. String values are part of the WRM specification, but are not currently supported by MAK.

Table 58. DIS articulations

Value	String	Description
1024	rudder	Rudder
1056	left_flap	Left flap
1088	right_flap	Right flap
1120	left_aileron	Left aileron
1152	right_aileron	Right aileron
1184	helicopter_main_rotor	Describes main rotor rotation
1216	helicopter_tail_rotor	Describes tail rotor rotation
1248	other_aircraft_control_ surfaces_defined_as_needed	Other articulated parts
2048	periscope	Periscope on subsurface watercraft
2080	generic_antenna	Antenna
2112	snorkel	Snorkel
2144	other_extendible_parts_ defined_as_needed	Other extendible parts
3072	landing_gear	Landing gear on a plane
3104	tail_hook	Tail hook on a plane for aircraft carrier landings
3136	speed_brake	Speed brake for aircraft
3168	left_weapon_bay_door	Left door to weapon bay
3200	right_weapon_bay_doors	Right door to the weapon bay

Table 58. DIS articulations (continued)

Value	String	Description
3296	bridge_launcher	Bridge launcher
3328	bridge_section_1	Bridge section 1
3232	tank_or_apc_hatch	Hatch on a tank or APC
3264	wingsweep	Identifies the angle that an aircraft's wings are moved forward or back
3360	bridge_section_2	Bridge section 2
3392	bridge_section_3	Bridge section 3
3424	primary_blade_1	Primary blade 1
3456	primary_blade_2	Primary blade 2
3488	primary_boom	Primary boom
3520	launcher_arm_primary	Primary launcher arm
3552	other_fixed_position_ parts_ defined_as_needed	Other fixed position parts
4096	primary_turret_number_1	Primary turret number 1
4128	primary_turret_number_2	Primary turret number 2
4160	primary_turret_number_3	Primary turret number 3
4192	primary_turret_number_4	Primary turret number 4
4224	primary_turret_number_5	Primary turret number 5
4256	primary_turret_number_6	Primary turret number 6
4288	primary_turret_number_7	Primary turret number 7
4320	primary_turret_number_8	Primary turret number 8
4352	primary_turret_number_9	Primary turret number 9
4384	primary_turret_number_10	Primary turret number 10
4416	primary_gun_number_1	Primary gun number 1
4448	primary_gun_number_2	Primary gun number 2
4480	primary_gun_number_3	Primary gun number 3
4512	primary_gun_number_4	Primary gun number 4
4544	primary_gun_number_5	Primary gun number 5
4576	primary_gun_number_6	Primary gun number 6
4608	primary_gun_number_7	Primary gun number 7
4640	primary_gun_number_8	Primary gun number 8
4672	primary_gun_number_9	Primary gun number 9
4704	primary_gun_number_10	Primary gun number 10
4736	primary_launcher_1	Primary launcher 1

Table 58. DIS articulations (continued)

Value	String	Description
4768	primary_launcher_2	Primary launcher 2
4800	primary_launcher_3	Primary launcher 3
4832	primary_launcher_4	Primary launcher 4
4864	primary_launcher_5	Primary launcher 5
4896	primary_launcher_6	Primary launcher 6
4928	primary_launcher_7	Primary launcher 7
4960	primary_launcher_8	Primary launcher 8
4992	primary_launcher_9	Primary launcher 9
5024	primary_launcher_10	Primary launcher 10
5056	primary_defense_systems_1	Primary defense systems 1
5088	primary_defense_systems_2	Primary defense systems 2
5120	primary_defense_systems_3	Primary defense systems 3
5152	primary_defense_systems_4	Primary defense systems 4
5184	primary_defense_systems_5	Primary defense systems 5
5216	primary_defense_systems_6	Primary defense systems 6
5248	primary_defense_systems_7	Primary defense systems 7
5280	primary_defense_systems_8	Primary defense systems 8
5312	primary_defense_systems_9	Primary defense systems 9
5344	primary_defense_systems_10	Primary defense systems 10
5376	primary_radar_1	Primary radar 1
5408	primary_radar_2	Primary radar 2
5440	primary_radar_3	Primary radar 3
5472	primary_radar_4	Primary radar 4
5504	primary_radar_5	Primary radar 5
5536	primary_radar_6	Primary radar 6
5568	primary_radar_7	Primary radar 7
5600	primary_radar_8	Primary radar 8
5632	primary_radar_9	Primary radar 9
5664	primary_radar_10	Primary radar 10
5696	secondary_turret_number_1	Secondary turret number 1
5728	secondary_turret_number_2	Secondary turret number 2
5760	secondary_turret_number_3	Secondary turret number 3
5792	secondary_turret_number_4	Secondary turret number 4

Table 58. DIS articulations (continued)

Value	String	Description
5824	secondary_turret_number_5	Secondary turret number 5
5856	secondary_turret_number_6	Secondary turret number 6
5888	secondary_turret_number_7	Secondary turret number 7
5920	secondary_turret_number_8	Secondary turret number 8
5952	secondary_turret_number_9	Secondary turret number 9
5984	secondary_turret_number_10	Secondary turret number 10
6016	secondary_gun_number_1	Secondary gun number 1
6048	secondary_gun_number_2	Secondary gun number 2
6080	secondary_gun_number_3	Secondary gun number 3
6112	secondary_gun_number_4	Secondary gun number 4
6144	secondary_gun_number_5	Secondary gun number 5
6176	secondary_gun_number_6	Secondary gun number 6
6208	secondary_gun_number_7	Secondary gun number 7
6240	secondary_gun_number_8	Secondary gun number 8
6272	secondary_gun_number_9	Secondary gun number 9
6304	secondary_gun_number_10	Secondary gun number 10
6336	secondary_launcher_1	Secondary launcher 1
6368	secondary_launcher_2	Secondary launcher 2
6400	secondary_launcher_3	Secondary launcher 3
6432	secondary_launcher_4	Secondary launcher 4
6464	secondary_launcher_5	Secondary launcher 5
6496	secondary_launcher_6	Secondary launcher 6
6528	secondary_launcher_7	Secondary launcher 7
6560	secondary_launcher_8	Secondary launcher 8
6592	secondary_launcher_9	Secondary launcher 9
6624	secondary_launcher_10	Secondary launcher 10
6656	secondary_defense_systems_1	Secondary defense systems 1
6688	secondary_defense_systems_2	Secondary defense systems 2
6720	secondary_defense_systems_3	Secondary defense systems 3
6752	secondary_defense_systems_4	Secondary defense systems 4
6784	secondary_defense_systems_5	Secondary defense systems 5
6816	secondary_defense_systems_6	Secondary defense systems 6
6848	secondary_defense_systems_7	Secondary defense systems 7

Table 58. DIS articulations (continued)

Value	String	Description
6880	secondary_defense_systems_8	Secondary defense systems 8
6912	secondary_defense_systems_9	Secondary defense systems 9
6944	secondary_defense_systems_10	Secondary defense systems 10
6976	secondary_radar_1	Secondary radar 1
7008	secondary_radar_2	Secondary radar 2
7040	secondary_radar_3	Secondary radar 3
7072	secondary_radar_4	Secondary radar 4
7104	secondary_radar_5	Secondary radar 5
7136	secondary_radar_6	Secondary radar 6
7168	secondary_radar_7	Secondary radar 7
7200	secondary_radar_8	Secondary radar 8
7232	secondary_radar_9	Secondary radar 9
7264	secondary_radar_10	Secondary radar 10
7296	deck_Elevator_1	Deck elevator 1
7328	deck_Elevator_2	Deck elevator 2
7360	catapult_1	Catapult 1
7392	catapult_2	Catapult 2
7424	jet_blast_deflector_1	Jet blast deflector 1
7456	jet_blast_deflector_2	Jet blast deflector 2
7488	arrestor_wires_1	Arrestor wires 1
7520	arrestor_wires_2	Arrestor wires 2
7552	arrestor_wires_3	Arrestor wires 3
7584	wing_(or rotor)_fold	Fold on plane’s wings or helicopter’s rotors
7616	fuselage_fold	Fold on plane’s body
32704	wind_direction	Parts that rotate into the wind
32736	wind_blades	Parts that rotate based on the wind speed, such as turbine blades
9120	first_left_wheel	Left wheel 1
9152	first_right_wheel	Right wheel 1
9184	second_left_wheel	Left wheel 2
9216	second_right_wheel	Right wheel 2
9248	third_left_wheel	Left wheel 3

Table 58. DIS articulations (continued)

Value	String	Description
9280	third_right_wheel	Right wheel 3
9312	fourth_left_wheel	Left wheel 4
9344	fourth_right_wheel	Right wheel 4
9376	fifth_left_wheel	Left wheel 5
9408	fifth_right_wheel	Right wheel 5
9440	sixth_left_wheel	Left wheel 6
9472	sixth_right_wheel	Right wheel 6
9504	seventh_left_wheel	Left wheel 7
9536	seventh_right_wheel	Right wheel 7
9568	eighth_left_wheel	Left wheel 8
9600	eighth_right_wheel	Right wheel 8
9632	ninth_left_wheel	Left wheel 9
9664	ninth_right_wheel	Right wheel 9
9696	tenth_left_wheel	Left wheel 10
9728	tenth_right_wheel	Right wheel 10

20.7.3 Shader-based Effects Maps in FBX

The MAK FBX Specification supports use of shader-based effects maps on models the same way that VR-Vantage applications support them on OpenFlight models. For details, see "11.4. Using Shader-based Effect Maps" on page 159 and "11.4.1 Composite Effects Maps" on page 161.

20.7.4 Using Texture Atlases, Metafiles, and Markings

The MAK FBX Specification supports use of texture atlases, metafiles, and dynamic markings the same way that VR-Vantage applications support them on OpenFlight models. For details, please see "18.10. Using Texture Atlases to Skin Models" on page 278 and "18.11. Adding Dynamically Changeable Markings to Models" on page 280.

20.7.5 Using Material-Classified (Sensor) Textures

SensorFX is a plug-in to VR-Vantage that lets it use physics-based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies. SensorFX requires material-classified textures to be included with each on the models.

SensorFX uses the following types of files to load the materially classified version of a model:

- Material classification file (emat2.dds and emat3.dds).
- Material system file (.ms).

- Intensity modulation file (.imod) (optional).
- Active regions file (.ar) (optional).

The .dds, .ms, and .imod files are generated by the GenesisMC tool. The .ar file is created by the model developer or someone who understands the location of hotspots on the model. SensorFX uses the names of the textures and models to look for the material classified configuration files.

See the GenesisMC and SensorFX documentation in the *VR-Vantage Users Guide* for the specific details about creating, converting, and storing material-classified textures for each model.

20.8. Encoding Damage to Non-Entity Dynamic Objects

VR-Forces supports damage to buildings as the result of munition impact. It also supports the ability to open and close doors, windows, and other features. This feature depends on the presence of switch nodes and appropriate model states in the models that can be damaged or changed.

Table 59. Switch nodes for dynamic buildings

Type	Allowable Values	Description
damage	none, slight, moderate, destroyed	General terrain damage.
damage_ structure	none, slight, moderate, destroyed	Damage specific to houses and buildings.
door	open, closed	The state of a door.
window	open, closed	The state of a window.



Other entity-based comments can also be used. For example, turning on or off the state of lights in or on man-made structures.

20.9. Mapping FBX Model Nodes to DIS Notes

If an FBX model does not follow the MAK FBX specification, MAK ONE applications must interpret the nodes in the FBX file so that it can visualize LODs and articulated parts.

For LODs, the MAK convention roughly follows the conventions used by the Unity engine. For articulated parts, MAK follows the conventions used by Aegis Corp.

Mapping of FBX nodes to DIS notes depends on using articulated part names for the nodes in the models. You can do this if you are using 3DS Max.

If you are creating FBX models for use in MAK ONE applications, following the MAK convention will allow you to import the model into a MAK ONE application with a minimum of difficulty. If you do not follow the conventions in naming nodes, you will need to map the nodes in your FBX files to DIS notes by hand as described in this section.

FBX nodes are mapped in the file *./appData/importConfig/FBX_NodeNameMap.csv*. The following example shows a portion of this file:

```
  Node,Comment,Type,Description
  turret_turn,@dis articulated_part 4096,comment,Primary Turret Number
1
  main_cannon_elev,@dis articulated_part 4416,comment,Primary Gun
Number 1
  m2_group_turn,@dis articulated_part 4160,comment,Primary Turret
Number 3
  m2_group_elev,@dis articulated_part 4480,comment,Primary Gun Number 3
  m240_group_elev,@dis articulated_part 4448,comment,Primary Gun Number
2
  wheel_1_1_group,@dis articulated_part 9120,comment,1st Left Wheel
  wheel_1_2_group,@dis articulated_part 9152,comment,1st Right Wheel
  wheel_10_1_group,@dis articulated_part 9696,comment,10th Left Wheel
  wheel_10_2_group,@dis articulated_part 9728,comment,10th Right Wheel
  turret_trav_group,@dis articulated_part 4096,comment,Primary Turret
Number 1
  cannon_elev_group,@dis articulated_part 4416,comment,Primary Gun
Number 1
  SimpleVehicle,Wreck,DamageState,Match node name to NORMAL/DAMAGE
state
  High,@dis lod 0 switch_in 50 switch_out 0,,
  Med,@dis lod 1 switch_in 200 switch_out 50,,
  Low,@dis lod 2 switch_in 1000 switch_out 200,
```

You can edit this file in a text editor or spreadsheet application. There are four columns, as follows:

- Node. The name of the node in the FBX file.
- Comment. The DIS note to apply to this node, as specified in the WRM specification.
- Type. The type of DIS note, for example a comment or an LOD switch.
- Description. A description of what the node represents.

When you create an FBX model, you can refer to this file and use the mappings that MAK has created for the FBX models that it includes in the application distribution. Or, you can update the file to map the nodes in your FBX models to the appropriate DIS notes. You can also use a combination of these approaches.

20.9.1 Specifying the FBX Texture Path

Most of the models provided with MAK ONE applications have their textures in the same directory as the model files. Imported models might not follow this convention. Therefore, MAK provides a way to specify search paths for textures. To specify an alternate texture directory, you can edit the file *./appData/importConfig/FBX_TexturePath.csv*.



21. Special Effects (Particle Systems)

This chapter describes how to maintain special effects using the Particle System Editor.

21.1. Particle Systems 376
21.2. Adding a New Particle System 376
21.3. Editing Particle Systems 377
21.3.1 Editing the Properties of Keyframes 378
21.4. Creating a Model Definition for a Particle System 379
21.5. Saving and Reverting Particle Systems 379
21.6. Previewing a Particle System 380
21.6.1 Viewing a Particle System With or Without Textures 382
21.6.2 Viewing a Particle System's Wireframes 384
21.6.3 Viewing Debug Visuals 385
21.7. Particle System Properties 387
21.7.1 Preview Parameters for Particle Systems 392
21.8. Particle System Collision Objects 393

21.1. Particle Systems

MAK ONE applications support special effects, such as smoke, fire, explosions, debris, dust trails, and weather effects. These effects are created using a powerful and versatile particle systems engine. MAK ONE applications are delivered with predefined special effects. In the Particle System Editor, you can modify the existing special effects to create new ones.

As long as a particle system is active, it emits particles. When the system is finished, it stops emitting new particles but the particles it previously emitted are still visible, for example, drifting smoke. Particle systems do freeze when the parent entity is frozen, such as when a scenario is paused.

A large set of parameters determines the look and behavior of the particle systems. They are described in "21.7. Particle System Properties" on page 387.

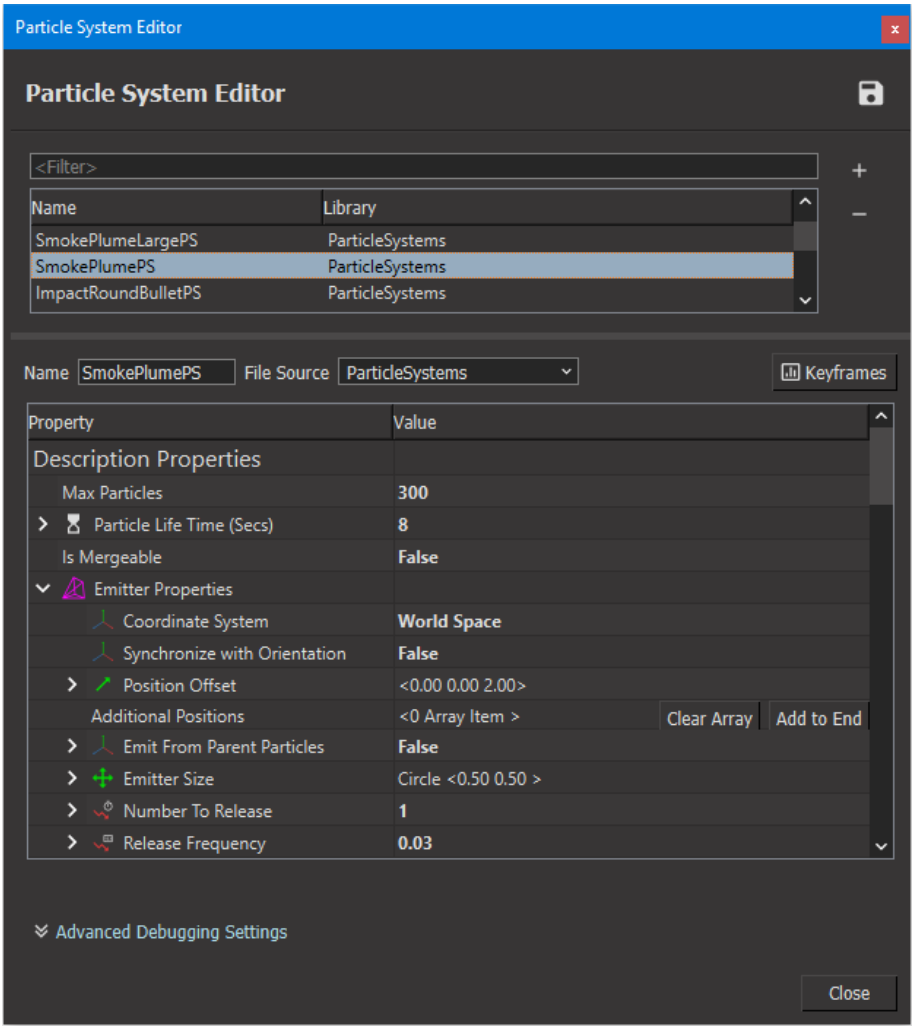
21.2. Adding a New Particle System

Particle systems have many parameters. You can edit the particle systems provided with MAK ONE applications. However, if you want to change a particle system, it is best to add a new system based on an existing one and edit the new system. This preserves the original particle system.

➤ **To add a new particle system:**

1. Choose **Settings > Particle System Editor**. The Particle System Editor opens (Figure 120).

Figure 120. Particle System Editor



2. In the list of particle systems, select a system of the type that you want to add.
3. Click the Add button (+). The New Name dialog box opens. The default name is the system you selected with a 0 appended to the end.
4. Optionally, edit the name for the new system.
5. Click OK. The new system is added to the list of systems.
6. Edit the parameters in the property sheet. For details about the parameters, see "21.7. Particle System Properties" on page 387.

21.3. Editing Particle Systems

You can edit the properties of a particle system. For details on the properties, see "21.7. Particle System Properties" on page 387.

➤ **To edit the properties of a particle system:**

1. Choose **Settings > Particle System Editor**. The Particle System Editor opens (Figure 120 on the previous page).
2. Select the particle system whose properties you want to edit.
3. Change values in the properties table as desired.
4. To save the changes, click Save Particle Descriptions.

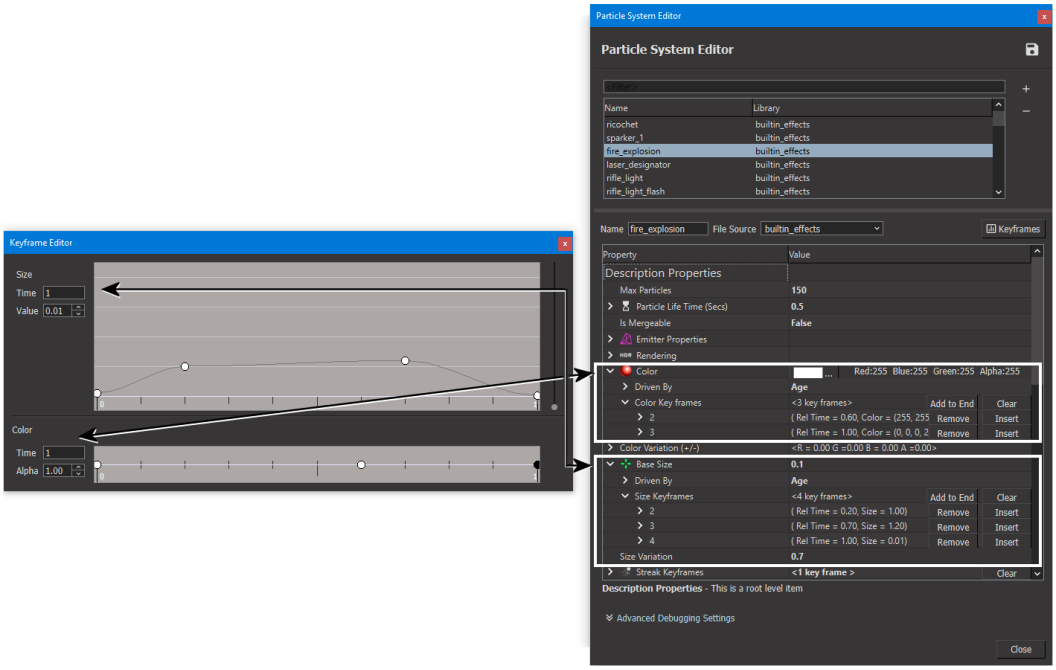
21.3.1 Editing the Properties of Keyframes

Keyframes define the beginning and ending points of transitions. You can set the properties of several types of keyframes in the properties table. (For descriptions of the keyframes, please see Table 60 on page 387.) You can also set the properties for some keyframes in the Keyframe Editor.

➤ **To edit keyframes in the Keyframe Editor:**

1. Select the particle system whose properties you want to edit.
2. Click the Keyframes button. The Keyframe Editor opens (Figure 121).

Figure 121. Keyframe Editor and keyframe properties



3. Change the values as desired.
4. To save the changes, click Save .

21.4. Creating a Model Definition for a Particle System

You can quickly create a model definition for a particle system. You can then add the model definition to an element definition.

When you create a model definition for a particle system, it uses the Particle System schema. It has the fxSystem parameter with a value that matches the name of the particle system.

> When you create a model definition for a particle system, if you want to save it, you must open the Visual Models Editor and follow the procedures for saving model definitions as described in "Chapter 15. Model Definitions" on page 217. Saving is easier if you load the model definition file to which you want to save the new model definition before you create the new model definition.

➤ **To create a model definition for a particle system:**

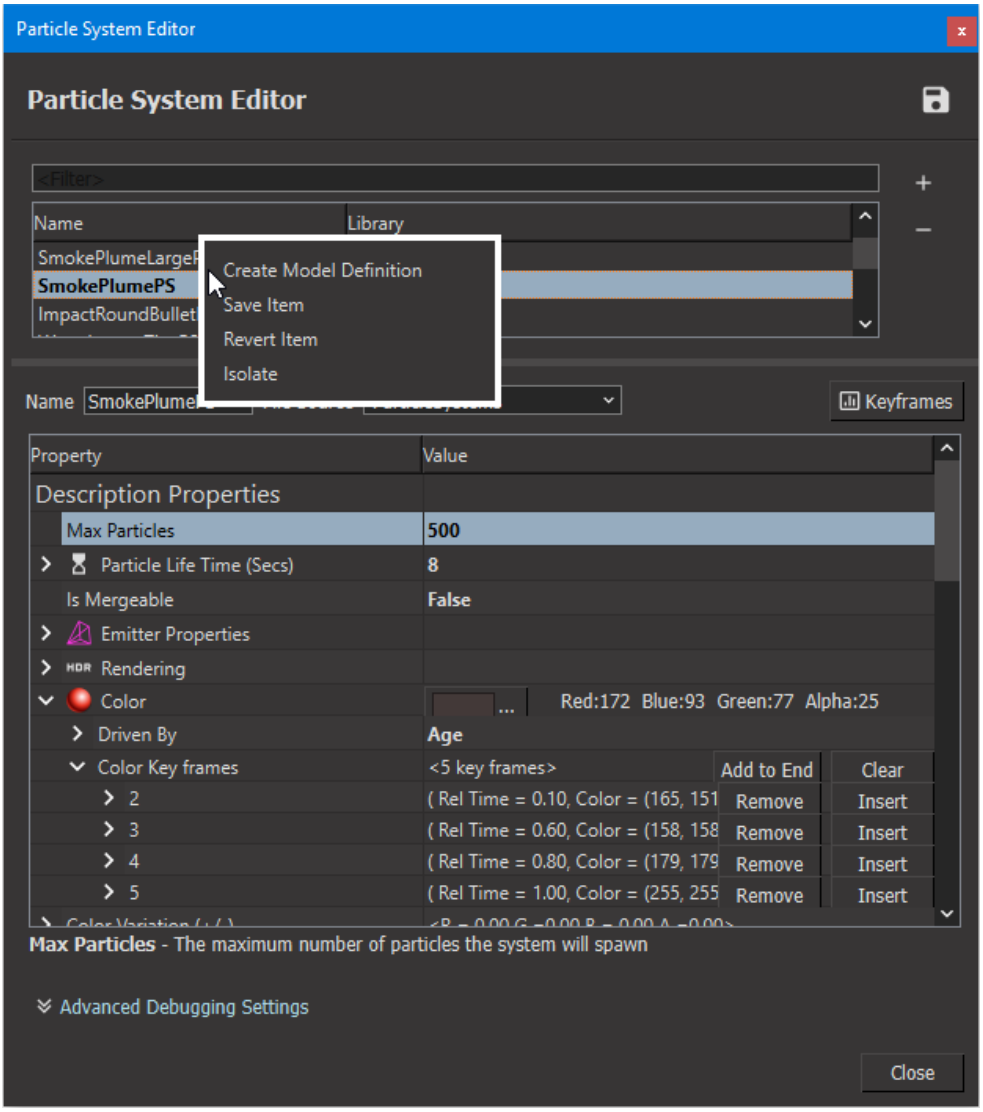
1. In the Particle System Editor, select the particle system for which you want to create a model definition.
2. Right-click and then choose Create Model Definition. The New Name dialog box opens.
3. Type a name for the model definition (or accept the default name).
4. Click OK. The new model definition is created.
5. Optionally, open the Visual Model Editors dialog box to confirm creation of the particle system and add any parameters that you want. In most cases, editing particle system properties in the Particle System Editor is preferable to editing its model definition.

21.5. Saving and Reverting Particle Systems

You can save and revert your changes. You can only revert the changes that you have made since the last time you saved.

➤ To save all particle systems, click Save ( ) (Figure 122).

Figure 122. Save Particle Systems



- To save changes to one particle system, right-click the particle system and choose **Save Item**.
- To revert changes to a particle system, right-click the particle system and choose **Revert Item**.

21.6. Previewing a Particle System

You can preview a particle system in the terrain (Figure 123). You can only test a particle system that has been saved.

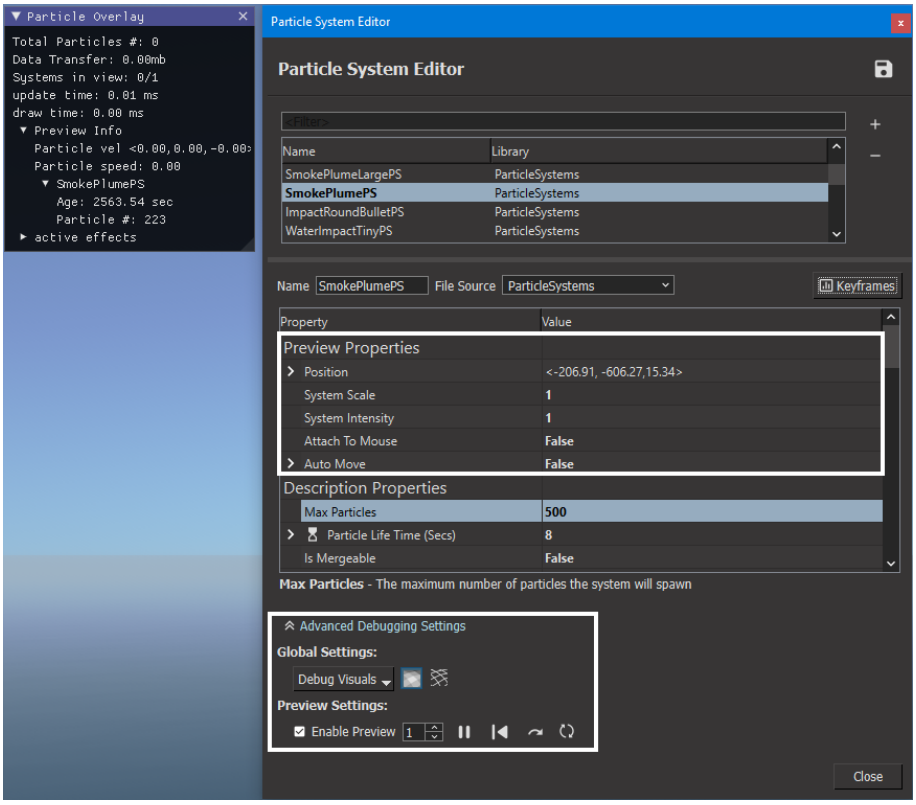
Figure 123. Preview of a smoke plume



➤ **To preview a particle system:**

1. Open a terrain. Depending on the size of the particle system, you should adjust the view so that you are close to the terrain or simulation object where you want to test the particle system.
2. Choose **Systems > Particle System Editor**. The Particle System Editor opens.
3. Select the particle system.
4. Click Advanced Debugging Settings to access the preview options.
5. Select the Enable Preview check box. A set of preview properties is added to the top of the properties list and the Particle Overlay statistics displays (Figure 124).

Figure 124. Particle System Editor, Advanced Debugging Settings



6. Optionally, enter the number of test particles to create.
7. If you want the preview to start again after it completes, click the Automatically Loop icon ( ). This is useful for previewing particle systems where the emitter's spawn lifetime is finite.
8. If desired, enter a name for the particle system.
9. Click in the terrain.
10. Use the preview buttons to view the particle system effects:

    / (Play/Pause). Start or stop the animation.

    (Jump to Start). Rewind the animation to its beginning.

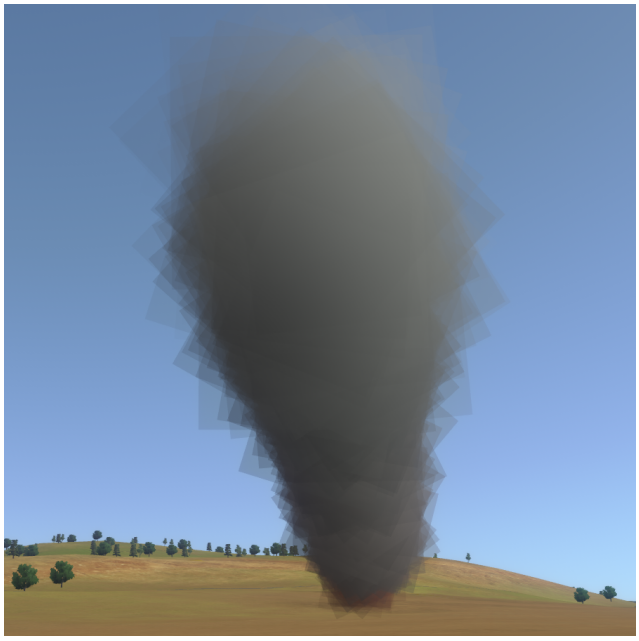
    (Step). Move step-by-step through the animation.

    (Automatically Loop). Loop the animation.

21.6.1 Viewing a Particle System With or Without Textures

While previewing a particle system, you can turn the textures on or off (Figure 125).

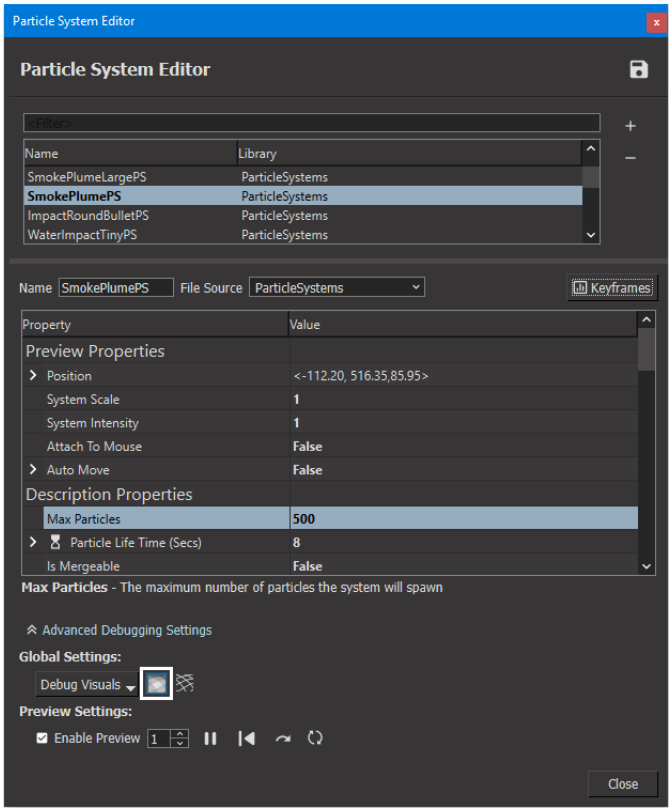
Figure 125. Preview of a smoke plume without textures



➤ **To enable or disable textures when you preview a particle system:**

1. In the Particle System Editor, select the particle system you want to preview.
2. Select the Enable Preview check box.
3. If you want the preview to start again after it completes, select the Automatically Loop check box. This is useful for previewing particle systems where the emitter's spawn lifetime is finite.
4. Click the Textures button () (Figure 126) to toggle the textures on or off.

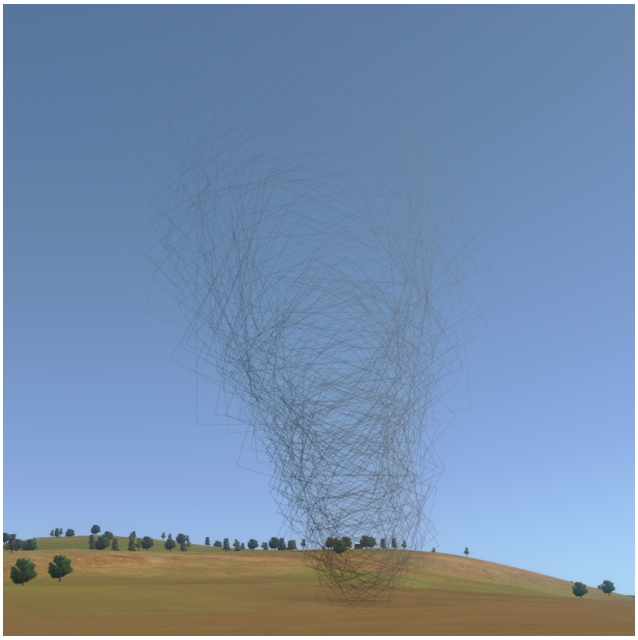
Figure 126. Particle System Editor - Textures



21.6.2 Viewing a Particle System's Wireframes

While viewing a particle system, you can choose to look at the wireframes for the particles (Figure 127).

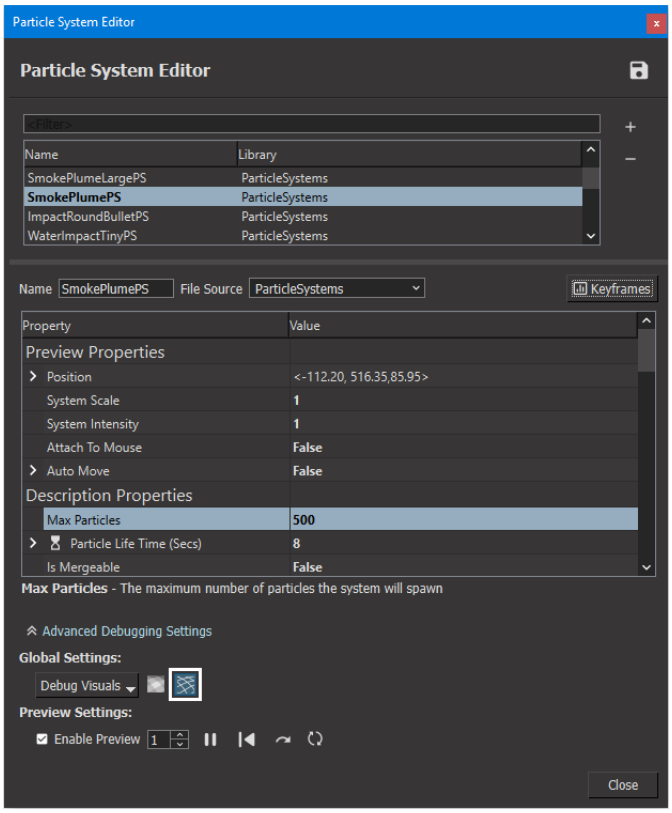
Figure 127. Preview of a smoke plume as wireframes



➤ **To view a particle system's wireframes:**

1. In the Particle System Editor, select the particle system you want to preview.
2. Select the Enable Preview check box.
3. If you want the preview to start again after it completes, select the Automatically Loop check box. This is useful for previewing particle systems where the emitter's spawn lifetime is finite.
4. Click the Wireframe Mode button ( ) (Figure 128) to toggle wireframes on or off.

Figure 128. Particle System Editor - Wireframes



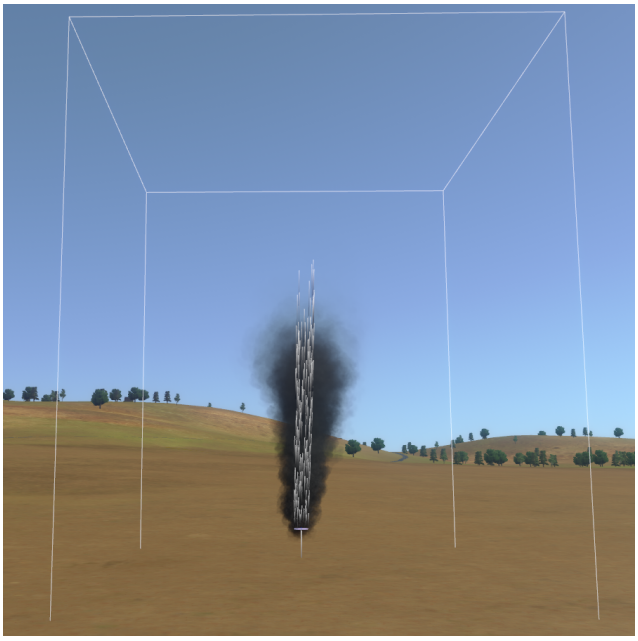
21.6.3 Viewing Debug Visuals

You can visualize the following aspects of a particle system:

- Lights
- Bounding boxes
- Emitters
- Forces
- Velocities

They are calculated for each frame.

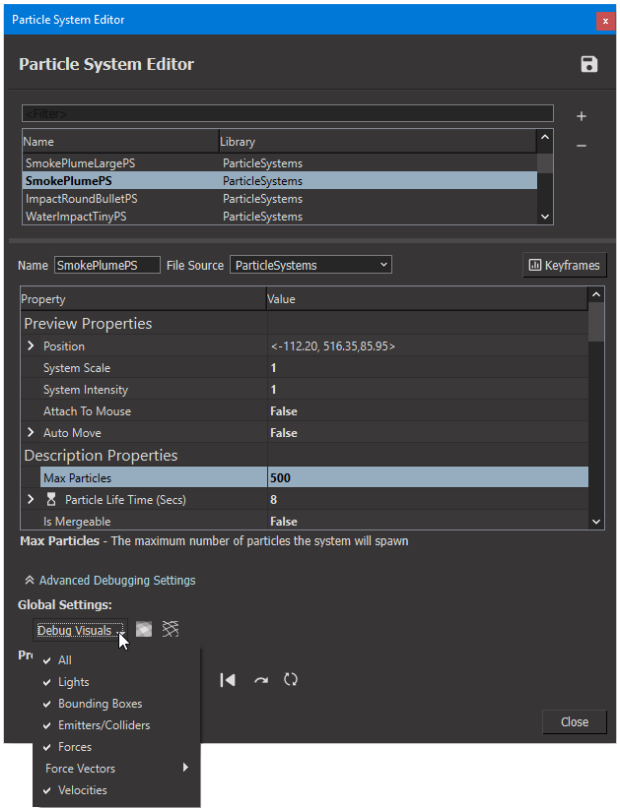
Figure 129. Preview of a smoke plume with all debug visuals enabled



➤ **To show the debugging visuals:**

1. In the Particle System Editor, select the particle system you want to preview.
2. Select the Enable Preview check box.
3. If you want the preview to start again after it completes, select the Automatically Loop check box. This is useful for previewing particle systems where the emitter's spawn lifetime is finite.
4. Select an option from the Debug Visuals list (Figure 130). You can toggle individual visuals on and off and can toggle all visuals on and off.

Figure 130. Particle System Editor - Debug Visuals



21.7. Particle System Properties

When you select a particle system, its properties are displayed in a table. When you select a property, a brief description appears at the bottom of the Particle System Editor. Table 60 describes the properties.

Table 60. Particle system properties

Parameter	Description
Max Particles	The maximum number of particles that this system creates. Once the maximum is reached, the system waits until particles die before emitting new particles.
Particle Life Time	How long, in seconds, each particle lives.
Is Mergeable	Indicates whether the system is used to represent a short-lived or transient effect, such as smoke from a gun barrel.
Base Size*	The size of particles. The size linearly interpolates between keyframes.
Size Variation	Applied to each particle when it is emitted. It is a fudge factor in addition to the size keyframes for varying particle size.

Table 60. Particle system properties (continued)

Parameter	Description
Streak Keyframes[1]	The amount the particles should stretch back toward the previous location.
Velocity Keyframes[1]	An impulse to apply to a particle as it hits the keyframe time.
Velocity Variation	A random velocity to apply to a particle as it hits the keyframe time.
Initial Velocity Scale	Indicates whether the particle inherits velocity from its parent.
Rotation	Indicates the initial variation of the rotation and also how quickly the particle rotates, in degrees per second.
Child Systems	Additional particle systems that are created and destroyed along with this system.
Emitter Properties	
Coordinate System	Controls whether particles move along with the emitter that created them. Particles in local space move along with the emitter as the emitter moves; they can also be scaled. Particles in world space move independently of the emitter and they can be made to collide with world geometry.
Synchronize with Orientation	If the effect is parented, use the orientation of the parent link.
Position Offset	The offset from the emitter.
Additional Positions	Other locations for emission.
Emit from Parent Particles	Indicates that particles should emit only from the root position of the parent's particles.
Emitter Size	The size and shape of the emitter: • Shape. The shape or volume to use. • Min. The minimum noise value. • Max. The maximum noise value.

[1]Keyframes let you vary the parameters of a particle system as a function of time. Each keyframe is associated with a time that is a percentage of the particle's lifetime. Most of the keyframe groupings require at least one keyframe, usually at time 0. Keyframes must be listed in time order. The user interface tries to maintain this requirement.

Table 60. Particle system properties (continued)

Parameter	Description
Number To Release	The number of new particles to emit when the system is updated. This is bounded by the release interval.
Release Frequency	How frequently the system attempts to emit new particles. (Bounded by the update rate.)
System Warm up time	How long, in seconds, the system should pre-update when the system starts.
Spawn Lifetime	The length of time, in seconds, that the system should emit. -1 indicates that the system should always be emitting particles.
Rendering	
Alpha Threshold	Controls the clipping of textures based on an alpha threshold. It is useful for controlling the shape of solid particles.
Material (SensorFX)	These settings are used for sensorization. Select the option that best matches the system to ensure that it displays correctly when viewed through sensorization.
HDR Gain	Multiplier for high dynamic range.
Emissivity Factor	This multiplier indicates how much color this system emits.
Specularity Factor	This multiplier indicates the particle system's smoothness and shininess, if any.
Metallicness Factor	This multiplier indicates the particle system's metallic appearance, if any.
Use Lighting	Indicates whether to use lighting calculations.
Particle Style	Indicates how the graphics are drawn. Typically, this is set to Billboard. • Billboard. Uses textures polygons. • Sparks. Uses line graphics instead of billboards.
Cast Shadow	If true, the particle system casts a shadow.
Depth Softness Factor	This multiplier indicates how much edges should blend when the particle system is near an object.
Texture Map	
Texture Map	The texture map to use on the particles. To use your own, add the file to the *./SharedData/16/latest/Particles* directory.

Table 60. Particle system properties (continued)

Parameter	Description
Number of X, Y Frames	This allows the texture map to be divided into smaller subtextures, which is useful in the following cases: • If animation speed is set to zero, a new particle is randomly assigned a subtexture. In this way you can make the same emitter create very different looking particles using the same texture. • If animation speed is set to a number other than zero the particles play through the subtextures like the pages of a flip book, allowing for animated particles.
Animation Speed	How quickly the animation changes, in frames per second.
Normalized Animation Time	If true, then an Animation Speed of 1 indicates that the animation plays once during the life of a particle.
Overall Blend Control	
Blend	Controls particles being blended with the background. (Controls if source and destination parameters work.)
Blend Src Function	OpenGL parameter describing the source transfer method, usually GL_ONE.
Blend Dest Function	OpenGL parameter describing the destination transfer method, usually GL_ONE.
Depth Write	Controls whether the particles write to the depth buffer. Turn this on and turn blending off for more solid-looking particles.
Color	
Driven By	This indicates whether the color is based on the age or the speed of the particle system.
Color Keyframes[1]	The color and alpha value to draw the particle with; colors linearly interpolate between keyframes.
Color Variation	Applied to each particle when it is emitted. This can give the particles random variations in color.
Forces	
Affected by Global Wind	If true, global wind affects the particle system.
Particle Mass	The mass of each particle impacts how wind and other forces impact the particle.
Air Resistance	This indicates how much wind and other forces affect the particle system.

Table 60. Particle system properties (continued)

Parameter	Description
Gravity	A constant acceleration force applied every frame. <0,0,0> is good for lighter-than-air particles. <0,0,-9.8> mimics normal earth gravity and is good for solid objects like debris.
Collision Objects	An array of planes that particles can interact with. Each plane has a position and orientation as well as a collision behavior. The behaviors are bounce, kill, and stick. If bounce is selected the bounce factor affects how much energy the particles retain after hitting the plane.
Local Space Collision Planes	Determines whether collision planes track with the object if the object moves.
Additional Forces	An array of forces that can affect the particles.
Max Particle Velocity	The maximum velocity a particle can have.
Emit Light	If true, the particle system emits light. This requires dynamic lights.
Advanced Scripts	
Scripts	You can specify custom scripts for: • Lifetime • Release Rate • Size • Initial Velocity • Streak • Color
Global Forces	
Force Type	The type of force: • Repulsor 2D. • Repulsor 3D. • Attractor 2D. • Attractor 3D. • Spinner 2D. • Wind Volume. • Lift Vortex.
Global Force	Indicates whether the force is global.

Table 60. Particle system properties (continued)

Parameter	Description
Force	Specifies the strength of the force.
Falloff Type	The type of falloff: • None. Force falls off 1/distance. • Constant. Force falls off 1/distance*distance. • RadiusSquared. The square of the specified Radius, which is derived.
Radius	The radius of the force if the Falloff Type is set to None.
Position	The position of the force.
Force Script	The script for the force.
Collision Objects	
Object Type	The type of the collision objects: either plane, sphere, or ellipsoid.
Bounce Factor	The amount by which to multiply the velocity (usually reduces velocity) as a result of the collision.
Collision Behavior	What to do when the particle strikes a collision plane. • Bounce. The particle bounces, but it maintains its life. • Bounce and die. The particle bounces, then it dies. • Stick. The particle maintains its position and its life. • Kill. The particle is immediately terminated.

21.7.1 Preview Parameters for Particle Systems

While previewing a particle system, you can change some properties. These changes do not affect the particle system itself, only the preview.

Table 61. Preview properties

Parameter	Description
Position	The particle system's coordinates in the terrain.
System Scale	The size of the particles for the preview.
System Intensity	The intensity of the particle system.
Attach to Mouse	If this is False, the preview displays where you click in the terrain. If this is True, the preview displays at the point of the mouse cursor. As you move the mouse, the particle system moves.
Auto Move	The speed and radius at which the particle should move automatically around the clicked location.

21.8. Particle System Collision Objects

By default, particle systems do not know anything about the entities and are not blocked by the superstructure of vehicles. As a result, if the eyepoint is inside an object that has interior geometry, particles may be visible inside the vehicle. This is a particular issue for rotor wash particle systems. To prevent particles from entering a vehicle, you can add a Particle Collision Object visualizer to an element definition.

Particle Collision Objects have the following parameters:

- **collisionobjecttype**. The supported object types are **sphere** and **ellipsoid**.
- **collisionobjectoffset**. The offset of the collision object from the center of the object specified as three comma-separated values that represent front offset, right offset, and down offset.
- **collisionellipsoidradiixyz**. The radius for an ellipsoid particle collision object specified as three comma-separated values that represent the width, length, and height.
- **collisionsphereradius**.The radius for a sphere particle collision object.
- **collisionorientationoffset**. An offset in degrees to the orientation of the collision object as (yaw, pitch, roll) in the direction (clockwise, counterclockwise, clockwise) respectively.



IV. DI-Guy

VR-Forces, VR-Vantage, and VR-Engage use DI-Guy to provide realistic human and animal characters. These characters are developed using the DI-Guy SDK and third-party graphics tools that are not included with MAK ONE applications.

MAK customers may want to add new DI-Guy characters or edit existing ones. There are four main aspects to a character: the geometric mesh, the textures, the rig (or skeleton), and the configuration files.

The textured geometric mesh of a character is what you see when a character is rendered. It is created using texture files that are UV-mapped to the geometric mesh that the character uses. A geometric mesh can be skinned to any skeleton to make use of that skeleton's various animation libraries, creating a large, varied set of characters. The geometric meshes provided by MAK are all encrypted. You cannot edit them. However, you could create your own texture files and overwrite them to the geometric meshes provided by MAK.

The rig determines how character joints are weighted. DI-Guy entities are categorized as Appearances within various Character Types. If you look at different DI-Guy character types in the Character Viewer, or explore the animations available to them in VR-Forces, you will see that different character types have different animations available to them. This is because each character type has its own set of animations. For example, the Police character type has animations specific to that character's needed motions, such as making an arrest . If you want to edit which animations are available for various character types, or to create an entirely new character type, you must use the DI-Guy SDK.

This section of the *Adding Content to MAK ONE Applications Reference Manual* describes the files that must be created or edited to add or modify DI-Guy characters. It also explains how to use the Character Viewer. The Character Viewer is included with MAK's 3D applications and lets you examine all the DI-Guy characters included in these products.

As noted previously, creating and editing DI-Guy characters requires additional tools and is a task for people with advanced graphics skills. Although teaching these skills is beyond the scope of this manual, we also include a brief tutorial that shows how to edit a texture file.



22. DI-Guy Content

This chapter is an introduction to customizing DI-Guy content.

22.1. Customizing DI-Guy Content 398
22.1.1 Encrypt Your Final Customized Geometry 398
22.1.2 Compressing Geometry Files 398
22.1.3 Where to Place Customized Files 398
22.1.4 The DI-Guy Data Directory Structure 399
22.1.5 Understanding DI-Guy Configuration Files 399
22.1.6 Autoload Configuration Files 402
22.2. Customizing Motion 402
22.3. Customizing Texture 402
22.4. Customizing Sound 402

22.1. Customizing DI-Guy Content

DI-Guy characters are highly customizable. Motions, geometry, textures, characters, and special effects (that is, DI-Guy content) can all be customized. Be sure to use the *./custom* directory when you modify DI-Guy characters. This segregates your custom characters from the installed characters and lets you easily migrate them to future releases of your MAK ONE application.



When editing configuration files, do not use Notepad. Use a text editor that preserves end-of-line information.

22.1.1 Encrypt Your Final Customized Geometry

When customizing *.flt* and *.dae* geometry files, always encrypt your geometry to protect DI-Guy intellectual property. This is required by the DI-Guy Model Use Agreement.

➤ To encrypt your customized geometry, open a command prompt and then enter:
`diguyEncryptGeometryFile` *`filename`*

22.1.2 Compressing Geometry Files

As an alternative to encrypting geometry files, you can compress them. This allows you to distribute them without distributing source files. The geometry processor compresses all uncompressed geometry in the custom directory.

➤ To compress your customized geometry, open a command prompt and then enter:
`DIGuyGeometryProcessor`

22.1.3 Where to Place Customized Files

When a MAK ONE application needs to access DI-Guy data, it looks in the *./custom* directory before looking in the default location for the file. The *./custom* directory acts as a shadow or mask directory to the default DI-Guy content directories. Therefore the files in it should use the exact same directory layout as the default directories. For example, if the application normally loads *./geometry/flt/model.flt*, it will first search for the file *./custom/geometry/flt/model.flt* before loading *./geometry/flt/model.flt*.



We strongly recommend that you leave all files installed by your MAK ONE application unchanged and place all your modified and new files in *./custom/*. This will allow your customizations to be portable to other installs and allow you to upgrade to new versions of your application with minimal or no changes to your custom content.

22.1.4 The DI-Guy Data Directory Structure

Here is a list of custom directories whose files mask identically named files in the standard directory:

- Geometry *./data/Lifeforms/DI-Guy/custom/geometry/*
- Textures *./data/Lifeforms/DI-Guy/custom/geometry/rgb/*, *./data/Lifeforms/DI-Guy/custom/geometry/png/*
- OpenFlight Models *./data/Lifeforms/DI-Guy/custom/geometry/flt/*
- Collada Files *./data/Lifeforms/DI-Guy/custom/geometry/dae/*
- Configuration Files *./data/Lifeforms/DI-Guy/custom/config/*
- Motions *./data/Lifeforms/DI-Guy/custom/motions/*

22.1.5 Understanding DI-Guy Configuration Files

DI-Guy content is declared in and loaded using configuration files. By default, the configuration file *./config/diguy.cfg* defines content. This file **includes** other configuration files, which also may **include** more nested configuration files, and so on.

If you want to better understand how DI-Guy organizes its data, we encourage you to examine, trace through, and modify files using the custom directory shadow technique. All of the configuration files are in ASCII text format, so they are easy to read and understand.

The following concepts and terminology are used in the DI-Guy configuration file system:

- Actor. The flesh-and-blood actor whose performance was captured using motion capture. Motion files reference the original actor. The DI-Guy motion engine often performs mathematics to scale motions to fit the end dimensions of the virtual character's appearance.
- Appearance. The specific geometry and textures that describe the visual look of the character. DI-Guy appearances can be skinned (deformable mesh) or segmented (rigid interpenetrated polyhedral). Many skinned appearances include equipment that may be segmented.
- Character_type. The combination of the set of actions a character can perform with the set of visual appearances the character can use.
- Joint. Describes the way two rigid links are joined together. Popular joints include pin, frozen, 6DOF (degrees-of-freedom), ball, 3DOF, and slider.
- Link. A link is a joint plus an offset. Often shapes are associated with links. Links can be traversed and drawn. Sometimes links are referred to as "bones".
- Shape. A mesh or polyhedral associated with a link. Unlike the link, which is conceptual in nature, shapes are actual drawables.
- Shapeset. A collection of one or more shapes.

The entries in *diguy.cfg* are organized into the following sections:

- Preface. This section defines very basic things about the content.
- Main Section. This section defines the bulk of DI-Guy content, often by pointing at lists of Level Two include file content.
- Level Two File. Common files using common syntax to describe DI-Guy content.



Paths in *diguy.cfg* are relative to the *./config* directory.

The Preface Section

The preface section of *diguy.cfg* defines very basic things. It is not usually modified except by very advanced DI-Guy users.

- geometry_set_priority_list. Order of operation for geometry file search.
- data_version. Specifies the DI-Guy version of the data, include the configuration files themselves.
- required_sdk_version. Specifies the version of DI-Guy software that this data is compatible with.
- *common/loaders.cfg*. Motion data loaders. Specifies the exact name of every variable for a particular DI-Guy character_type. Each loader has a name, and that name is referenced in the character_type definitions.
- *common/shader_matrix_index_tables.cfg*. A list of different object types and bones that a DAE file has, needed to map a skinned character to a DI-Guy skeleton.
- *common/common_links.cfg*. The variables for popular links, as well as the joint type and some skeletal hierarchical information.
- *common/shape_and_joint_data.cfg*. The name of a shape is joined to the name of the joint. Therefore you can determine what shapes will be drawn when a joint is drawn.
- *common/common_actors.cfg*. The actor object is described by its shapesets and links.
- *common/common_characters.cfg*. Various DI-Guy primitive characters, such as lines, are described in this file. Character definitions usually have a graphics_ class, actor, link list, and default_equipment.
- *diguy/required_characters.cfg*. Defines the character waypoint.

The Main Section

This section of *diguy.cfg* defines most of the DI-Guy content. If you want to customize DI-Guy content, you must understand this section and the Level Two content it references.

- *diguy/diguy_actors.cfg*. Lists individual actor configuration files. See *diguy/actor_<actor_name>.cfg* for more information.

- *diguy/diguy_characters.cfg*. Contains a list of character_type definitions. See the *diguy/char_<character_type>.cfg* definition for more information. The file also includes *diguy/diguy_character_type_maps.cfg*, which associates a single word (for example, soldier) with a character_type and appearance combination to support generic requests for a type of character where DI-Guy responds with the best match.
- *diguy/character_types_list.cfg*. Defines the set of character_types that match simple descriptions. For example, more than a dozen character types are deemed relevant when querying which character_types are available as male pedestrians. Like *diguy/diguy_character_type_maps.cfg*, this list supports wizard-type queries into the DI-Guy content database for best matches to user search criteria.
- *diguy/exface_shapesets.cfg*. Defines expressive face appearances and geometry available to particular actors.
- *diguy/gestures.cfg*. Defines lists of gesture motions available.
- *diguy/motex_arcs.cfg*. Defines a list of available motion textures, used to add visual noise to character motions.
- *diguy/default_particle_descriptions.cfg*. Defines default particles and the parameterizations that define them.
- *diguy/appearances.cfg*. Lists of DI-Guy appearance category include files. See *diguy/appearances_<appearance_category>.cfg* for more information.
- *diguy/default_appearance_effects.cfg*. Appearance effects are particle effects that augment appearances, particularly of DI-Guy vehicles. Appearance effects also are used to support incoming DIS/HLA designations such as “smoking” or “on fire”.

The Level Two Files Section

- *diguy/actor_<actor_name>.cfg*. Each actor configuration file defines the shapesets available to the character, the links that compose the actor, and an actor definition. The actor definition includes the standing_hip_height used to scale actor motions to end appearance dimensions. The actor definition also includes *common/links_<link category>.cfg*, which tells DI-Guy what variables will be found in the motion file. Finally, the actor definition includes a shader_matrix_index_table assignment.
- *diguy/actor_<actor_name>_shapesets.cfg*. Defines the geometry and textures that will be made available to character appearances that are supported by the actor.
- *diguy/actor_<actor_name>_links.cfg*. Defines the skeletal hierarchy of the actor as a set of links, each with a name, offset, associated joint, and DOF variable names.
- *diguy/appearances_<appearance_category>.cfg*. Defines an appearance. An appearance consists of its name, the actor with which is associated, a number of “items” which designate shapesets used by this appearance, and information that helps categorize the appearance for wizard-like queries.

22.1.6 Autoload Configuration Files

Any files placed in the *./custom/config* directory override their counterpart in the *./config* directory.

In addition, any configuration file in *./custom/config/* that begins with autoload_ is read automatically during DI-Guy configuration initialization.

22.2. Customizing Motion

You can customize motion using the DI-Guy Motion Editor. Please see *DI-Guy Motion Editor Users Guide* for details. (You must have the DI-Guy SDK to use the Motion Editor. It is not included with the MAK simulation and visualization applications.)

22.3. Customizing Texture

Textures use the RGB, RGBA, and PNG format.

➤ To customize textures, edit an existing texture and save it with the same name in *./custom/geometry/rgb* or *./custom/geometry/png*.

Alternatively, you can create a new appearance that uses your new texture instead of overriding the existing appearance.

22.4. Customizing Sound

➤ To customize sounds, edit an existing sound file and save it with the same name in *./custom/sfx*. You can also add new sounds to this directory.



23. Using the Character Viewer

This chapter explains how to use the Character Viewer to view characters, appearances, and actions.

23.1. Introduction to the DI-Guy Character Viewer 404
23.2. Viewing Characters, Appearances, Gestures, and Actions 405
23.2.1 Filtering the Character Type and Appearance Lists 406
23.2.2 Changing the Viewing Angle for a Character 406
23.2.3 Changing the Quality of the Character 407
23.2.4 Demonstrating a Character's Aim and Locomotion Capability 408
23.2.5 Viewing Different LOD Levels 408
23.2.6 Changing How a Character Is Rendered 409
23.2.7 Viewing Multiple Appearances at One Time (Army Mode) 411
23.2.8 Advanced Character Editing 413
23.2.9 Reloading Characters that Have Been Edited 414
23.2.10 Controlling Character Movement 415
23.3. Improving Performance 415
23.3.1 Displaying the Performance Profiler 416
23.4. Running a Lua Script 416
23.5. Viewing Character Geometry Details 416
23.6. The Appearance Editor 417
23.6.1 Visualizing Character Structure 419
23.6.2 Editing Configuration File Text 419
23.6.3 Editing Appearance Effects 419
23.7. Using the PBR Material Tester 420
23.7.1 Reloading Shaders 421
23.8. The Character Viewer Log Window 421
23.8.1 Setting the Notification Level 421
23.9. Broadcasting the Character Viewer over the Network 422
23.9.1 Joining a DIS Exercise 422
23.9.2 Joining an HLA Exercise 423

23.1. Introduction to the DI-Guy Character Viewer

The DI-Guy Character Viewer is a utility program that allows you to view and edit all of the DI-Guy characters provided with your MAK application. It is also a testing tool for VR-Vantage customers who create their own models. The Character Viewer has the following windows:

- 3D window. Displays the selected character, appearance, and action.
- Content Library. The Content Library lets you select the character appearances, gestures, and actions you want to view. It also has advanced features for experimenting with lighting and geometry.
- Appearance Editor. The Appearance Editor is a tool for advanced users who want to edit the parameters of character models.
- Geometry Viewer. The Geometry Viewer gathers all of the information for the selected character into one window so you can easily understand its details.
- Log. Displays errors, warning, and informational messages.

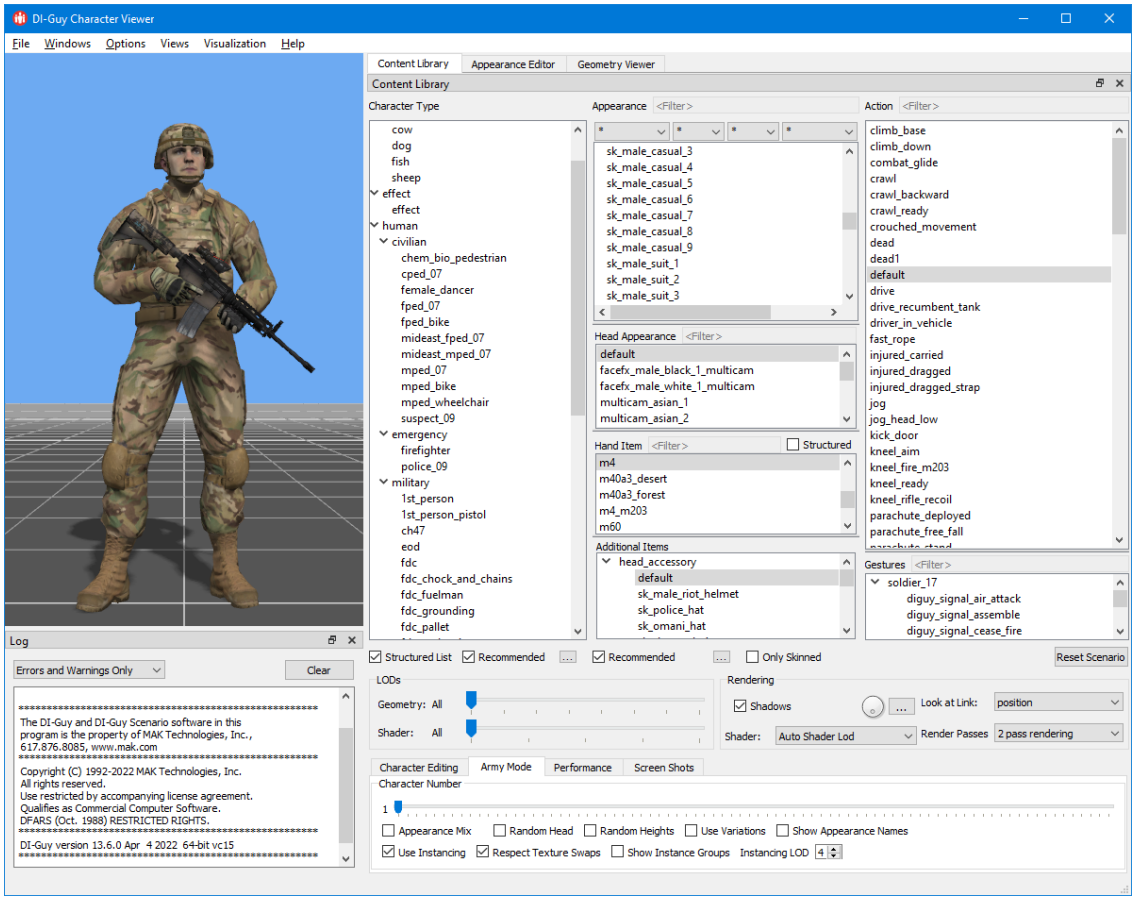
You can undock and resize the windows just as you would windows in other applications.



The Character Viewer is a Windows-only application.

➤ To open the Character Viewer, on the Start menu, choose **MAK Technologies > DI-Guy Character Viewer (64-bit)**. The Character Viewer opens (Figure 131).

Figure 131. Character Viewer



23.2. Viewing Characters, Appearances, Gestures, and Actions

The Character Viewer lists all of the DI-Guy characters available in your application. For each character, it lists all of the appearances, gestures, and actions that are available for that character. You can view any combination of character, appearance, gesture, and action. You can filter the Appearance list and you can change the perspective from which you view the characters.

➤ **To view a character and its appearances, actions, and gestures:**

1. Select the Content Library tab.
2. In the Character Type list, expand the list for the type of character you want to view.
3. Select the character that you want to view. It displays in the 3D window.
4. Optionally, to see only skinned characters in the Appearance list, select the Only Skinned check box (below the list).
5. In the Appearance list, select the appearance that you want to see for this character.

If the selected character and appearance has head appearances, they are listed in the Head Appearance list. If the selected character and appearance has hand items, they are listed in the Hand Item list.

6. Optionally, select a head appearance.
7. Optionally, select a hand item.
8. In the Action list, select the action that you want to view for this character. The character starts the action. If it is a traveling action, the character repeats it continuously.
9. In the Gestures list, select the gesture that you want the character to use.

23.2.1 Filtering the Character Type and Appearance Lists

You can filter the Character Type and Appearance Lists in the following ways:

- Organize the Character Type list by type of character or alphabetically.
- Display character types and appearances at or above a specified level of quality.

> The quality of a character or appearance is specified in its **quality_bias** parameter. The quality is a somewhat arbitrary value, assigned by the model developer, based on their opinion of the quality of the character relative to other characters. The characters provided with DI-Guy have quality values ranging from 1 to 10.

- Display only skinned appearances.
- Filter appearances by country, gender, age, and ethnicity. The filter options change based on the character type.

➤ To organize the Character Type list by category of character, select the Structured List check box. To organize it alphabetically, clear the check box.

➤ To display character types, appearances, or both that are at or above a specified quality level threshold, select the Recommended check box under the list.

➤ To filter the Appearance list by country, gender, age, or ethnicity, select an option from one of the drop-down lists at the top of the Appearance column.

➤ **To specify the quality level threshold to use when the Recommended check box is selected:**

1. Click the Browse button under the list for which you want to specify a recommended level of quality. A Quality Threshold dialog box opens for the list (Set Character Type Quality Threshold or Set Appearance Quality Threshold.)
2. Set a quality threshold value. Higher values indicate a higher level of quality.
3. Click OK.

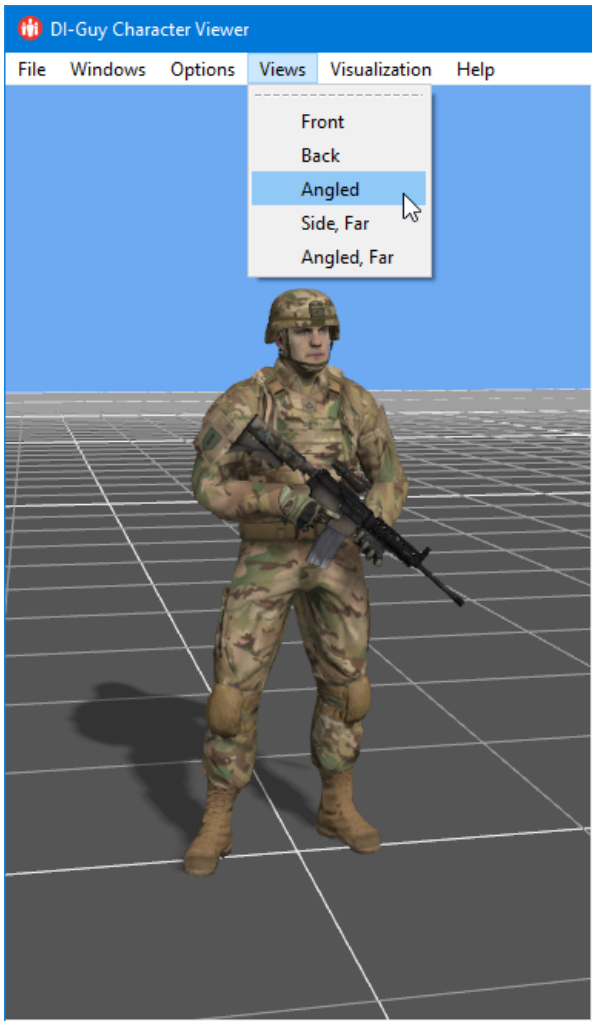
23.2.2 Changing the Viewing Angle for a Character

The default view of a character is from the front. You can quickly change to a predefined set of angles or move the character using your mouse or keyboard.

➤ **To change the viewing angle of a character, do one or more of the following:**

- Select an option from the View menu (Figure 132).

Figure 132. Views menu



- Zoom in and out using the left and right mouse buttons, or pressing **W** or **S**.
- Change the angle of view by holding down both mouse buttons and moving the mouse. You can also press **A**, **D**, or the arrow keys.

23.2.3 Changing the Quality of the Character

To optimize performance, DI-Guy characters have several quality levels. The closer a character is to the camera, the higher the quality that is used. If a character is farther away from the camera, it can be rendered at a lower quality, thereby improving performance. You can view characters at their various quality levels.

➤ **To change the quality of the character in the viewer:**

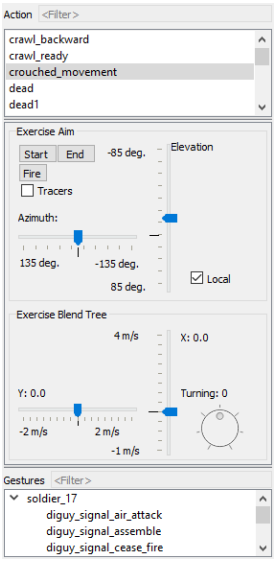
1. Click the browse button below the Character Type list. The Set Character Type Quality Threshold dialog box opens.

2. Select a quality level. The quality of the character in the viewer changes.
3. Click the browse button below the Appearance list. The Set Appearance Quality Threshold dialog box opens.
4. Select a quality level. The quality of the appearance in the viewer changes.

23.2.4 Demonstrating a Character's Aim and Locomotion Capability

When you select an action in the Action list, the Character Viewer displays the action in the 3D window. For some aim actions, the Character Viewer opens a panel where you can you vary the aim azimuth and elevation (Figure 133). For some movement actions, you can vary the blend tree settings.

Figure 133. Exercise Aim panel



➤ **To demonstrate aim and locomotion capability:**

1. Select an aim action or an action that uses a blend tree. A panel gets added to the Character Selector window between the Action list and the Gestures list. You may need to resize the window as a whole or the list windows to see all of the panel.
2. Adjust any of the sliders in the panel to see how they affect the character's movement and aiming.

23.2.5 Viewing Different LOD Levels

You can view a character at the different LOD levels available and you can view different quality of shaders.

➤ **To change the LOD level:**

1. In the LODs group box, adjust the Geometry slider to see the different LODs for visual quality.

2. In the LODs group box, adjust the Shader slider to see the different LODs for shaders.

23.2.6 Changing How a Character Is Rendered

The Rendering group box has options for lighting, shadows, and shaders.

Changing Lighting and Shadows

You can quickly make simple changes to the lighting and shadows applied to the character. You can also make more advanced changes.

➤ **To change lighting direction and shadows:**

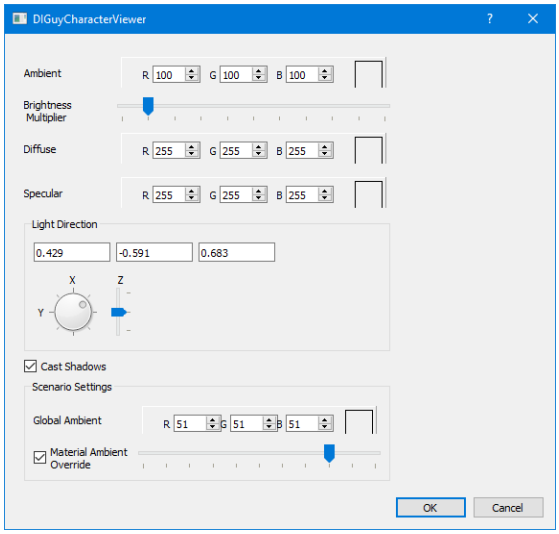
1. In the Rendering group box, select the Shadows check box to enable shadows.
2. Rotate the lighting widget to change the direction from which light is applied to the character (and therefore, the direction of the shadow).

Configuring Advanced Lighting Options

➤ **To make advanced changes to the lighting:**

1. In the Rendering group box, click the Browse button next to the lighting widget. The Lighting dialog box opens (Figure 134).

Figure 134. Lighting dialog box



2. Change the color of ambient, global ambient, diffuse, or specular light by changing any of the RGB values or by clicking the color swatch and selecting a different color.
3. In the Light Direction group change both the direction and altitude of the light source.
4. Adjust the Brightness Multiplier slider to increase the brightness of the diffuse and specular light.

5. To enable or disable shadows, select or clear the Cast Shadows check box. This check box is linked to the Shadows check box in the Rendering group box.
6. Click OK.

Viewing a Character's Structure

You can view a character's wireframe and skeleton.

➤ To view a character's wireframe, in the Rendering group box, select the Wireframe check box.

➤ To view a character's skeleton, in the Rendering group box, select the Wireframe check box.

Using the Shader Debugger

The shader debugger allows you to visualize the different texture maps that are used as source data. DI-Guy provides some shader debugging capabilities within its lighting shaders. The shaders are instrumented such that setting different flags (through uniforms) changes the behavior of the shader. Typically it changes fragment colors based on some parameter or parameters. Figure 135 shows two different shader settings.

Figure 135. Shader debugging



Auto Shader Lod | Normal Map

➤ To change the shader being used, in the Rendering group box, select an option from the Shader list.

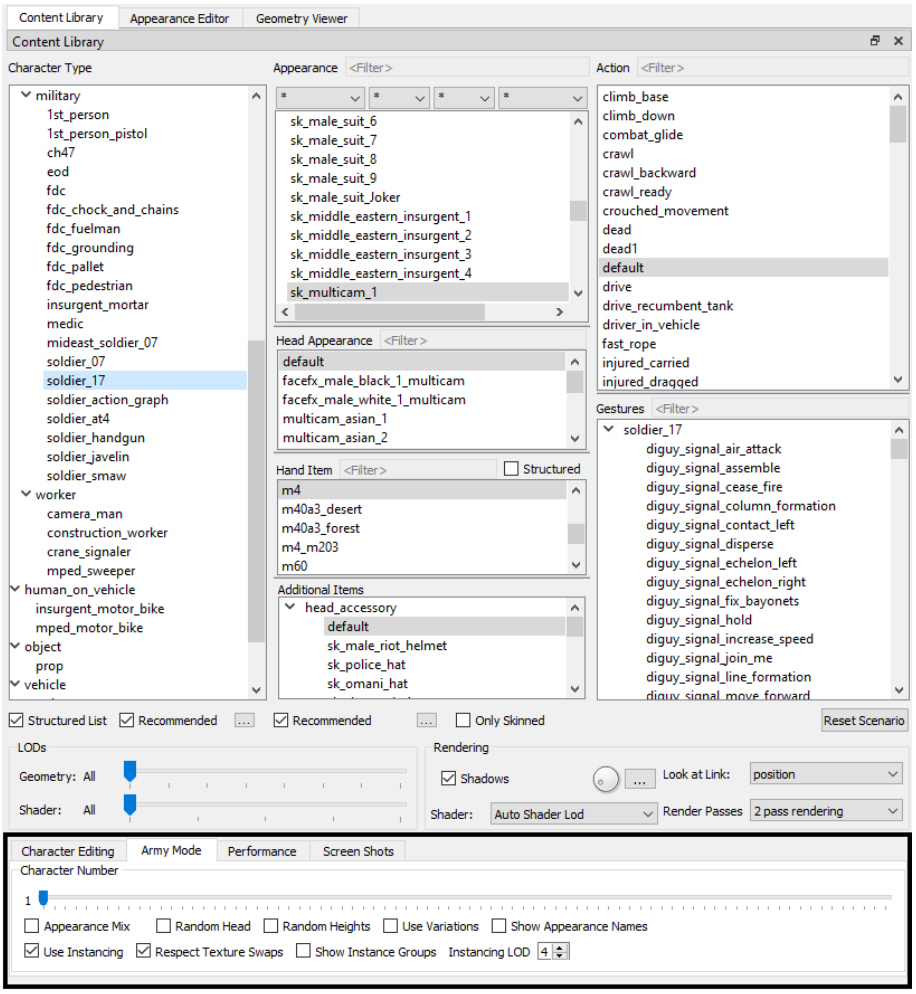
23.2.7 Viewing Multiple Appearances at One Time (Army Mode)

You can display many instances of a character at one time using different appearances.

➤ **To view multiple appearances for a character:**

1. At the bottom of the Content Library, select the Army Mode tab (Figure 136).

Figure 136. Army mode



2. Drag the Character Number slider to increase the number of characters displayed in the viewer window (Figure 137).

Figure 137. Multiple character instances (army mode)



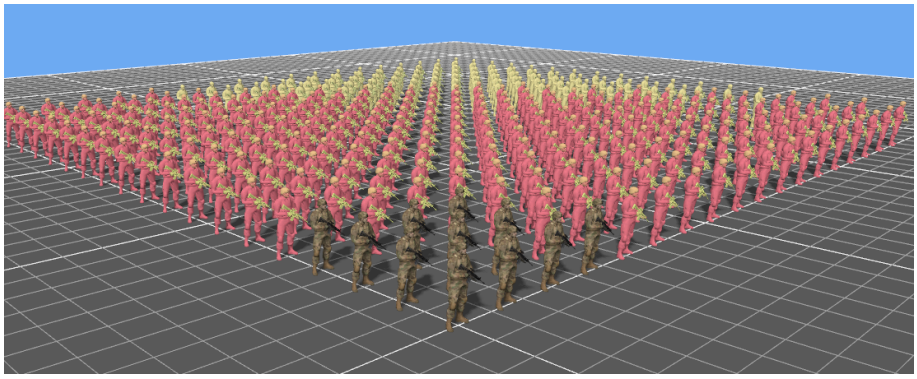
3. Select the desired check boxes to change the variations in appearance of the characters in the viewer.

Instancing Options in Army Mode

The Army Mode tab has several options for testing instancing. When you use instancing, characters are lower quality, but display more quickly and use less memory. You can see the result of using instancing by turning on the profiler. (For information about the profiler, see "23.3.1 Displaying the Performance Profiler" on page 416.)This option is useful only if you are viewing multiple characters in army mode. When the Use Instancing check box is selected, additional check boxes are enabled:

- Respect Texture Swaps. Models have a base mesh that has different texture maps applied to it to create the different appearances for a character. For example, a character that wears a suit, might have several different color suits. Replacing the base texture for different appearances is called a texture swap. If this option is selected, the Character Viewer displays the different textures. However, texture swaps affect performance. You can choose to not use texture swaps.
- Show Instance Groups. When you enable instance groups, the characters are colored to show the distance at which LOD changes take effect (Figure 138).

Figure 138. Instance groups with Instancing LOD set to 4



- Instancing LOD. Specifies the LOD at which instancing turns on.

➤ **To enable instancing:**

1. Click the Advanced button at the bottom of the page. Two tabs are added to the bottom of the Character Selector (Figure 136 on page 411).
2. Select the Army Mode tab.
3. Select Use Instancing.

23.2.8 Advanced Character Editing

The 3D window can display the bounding volume (Figure 139), inverse kinematics (Figure 140), aim vectors (Figure 141), LODs, and more. You can also see how a character looks with different rending approaches.

Figure 139. Character bounds

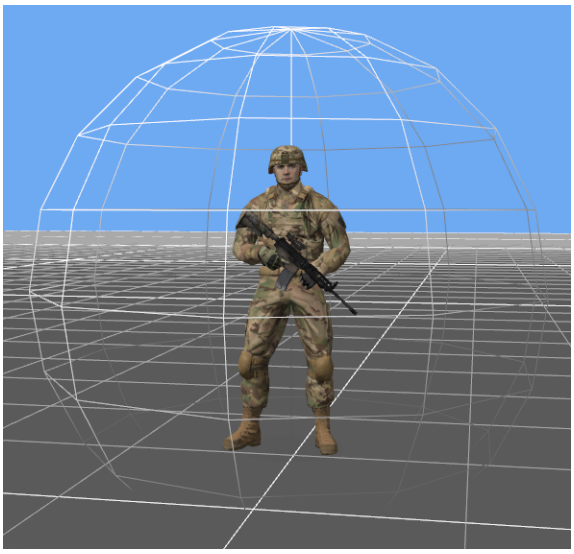


Figure 140. Inverse kinematics

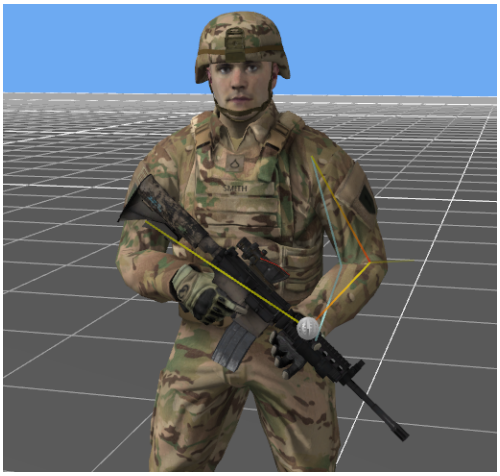


Figure 141. Aim vector



- To view a character's bounding volume, choose **Visualization > Character Bounds**.
- To see inverse kinematics, choose **Visualization > Inverse Kinematics**. Not all characters display inverse kinematics. This depends on the hand item selected.
- To see aim vectors, choose **Visualization > Aim Vectors**. This displays the vectors only for characters that have a weapon that is configured for aim vectors and an action that involves aiming.

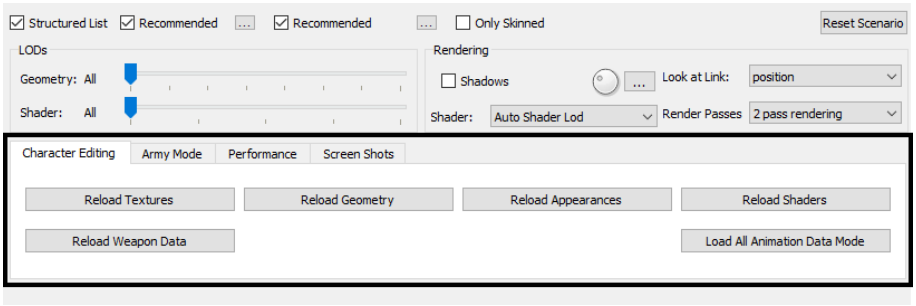
23.2.9 Reloading Characters that Have Been Edited

If you are editing some aspect of a character in another application, such as Photoshop, you can load the updated files without exiting the Character Viewer. You can reload textures, geometry, appearances, shaders, weapon data, and all animation data.

➤ **To reload the files for a character:**

1. At the bottom of the Content Library, select the Character Editing tab (Figure 142).

Figure 142. Character Editing Tab



2. Click the button for the type of data that you want to reload.

23.2.10 Controlling Character Movement

If you select a character action that involves movement, such as walking, or firing, the character in the viewing window displays that action. You can start and stop the action with the playback controls.

The Playback Controls button bar has a full set of playback controls. Only the Start, Fast Forward, and Stop buttons are used.

➤ **To use the playback controls:**

1. Choose **Window > Playback Control**. The Playback Controls are added to the Character Viewer.
2. Click the Forward button to make the character move.
3. Click the Fast Forward button to make the character move faster.
4. Click the Stop button to stop movement.

23.3. Improving Performance

The Character Viewer has some settings that can improve performance. They are:

- Use Multithreaded Pipeline. Use multithreading for the general animation system.
- Use Asynchronous Texture Loader. Load textures in the background.
- Use Delayed Motion Loading. Does not load motions until the first time they are played.
- Freeze Characters. Characters cannot move.
- Animation Load Management. Grants access to the Load Manager Settings in the ImGui Debug overlay.

➤ **To enable of disable performance options:**

1. At the bottom of the Content Library, select the Performance tab.
2. Select or clear the check boxes to enable or disable performance options.

23.3.1 Displaying the Performance Profiler

The Character Viewer can display performance statistics for the characters that you are viewing. It displays frame rate statistics, a histogram, and geometry statistics.

➤ **To display the performance profiler:**

1. Click the character display window to make it active.
2. Press **Ctrl**+**P**. Continue pressing **Ctrl**+**P** to cycle through the performance displays.

23.4. Running a Lua Script

You can run Lua scripts to debug characters. You can access most of the *diguyCharacter* API as well as *diguyScenario* and *diguyApp* functions. You can also load and save scripts.

➤ **To run a Lua script:**

1. Choose **Window > Script Editor**. The Code Editor dialog box opens.
2. Type the code you want to run.
3. Click Trigger.

➤ **To load a Lua script:**

1. In the Code Editor, click Load. The Load Script window opens.
2. Navigate to the file that you want to load.
3. Click Open.

➤ **To save a Lua script:**

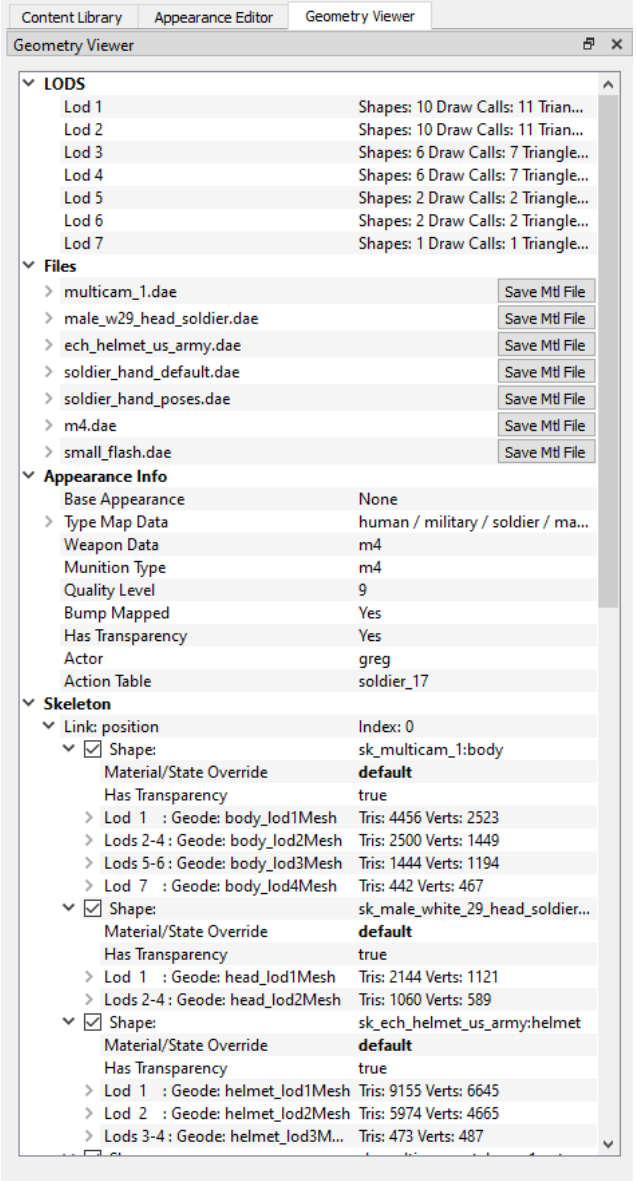
1. Click the Save button. The Save Script window opens.
2. Navigate to the folder where you want to save the file.
3. Click Save.

23.5. Viewing Character Geometry Details

DI-Guy characters are rendered based on information in many different configuration files. Trying to understand a character's structure and functioning by examining all these files can be a tedious process. The Geometry Overview window gathers all of the information for the selected character into one window, helping you easily understand its details.

➤ To display the Geometry Viewer, select the Geometry Viewer tab (Figure 143).

Figure 143. Geometry Overview



23.6. The Appearance Editor

The Appearance Editor is a graphical user interface that lets you edit the character appearance files. By using the Appearance Editor rather than editing the files in a text editor, you can avoid syntax errors and invalid information. And since the Appearance Editor shows the configuration for the character selected in the Character Selector, you do not have to figure out which configuration file to edit for that character.

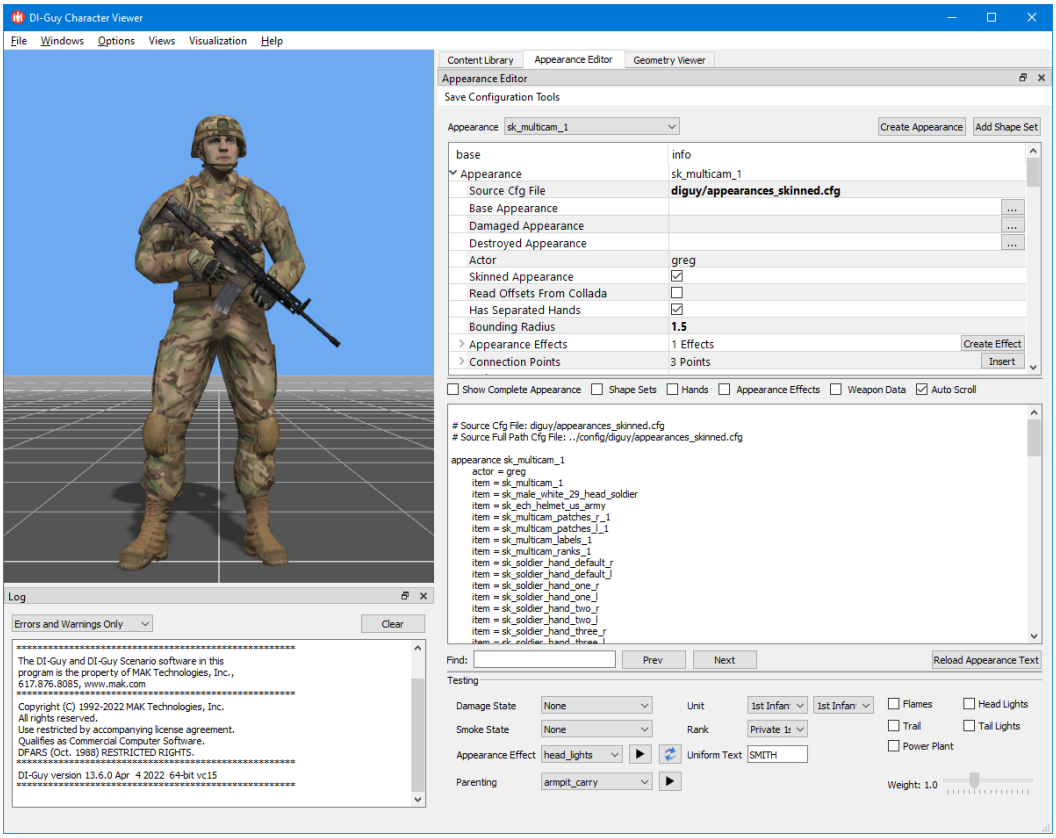
Although the Appearance Editor lets you edit configuration files without touching the actual files, it also displays the contents of the configuration files, so that you can edit them directly if you want to. Sometimes it is quicker to edit the file, particularly if you want to delete a large section of text.

The Appearance Editor is a tool for experienced modelers. It is beyond the scope of this manual to teach all the aspects of modeling needed to use it effectively.

➤ **To use the Appearance Editor:**

1. In the Content Library window, select the character whose appearance you want to edit.
2. Select the Appearance Editor tab (Figure 144).

Figure 144. Appearance Editor



3. In the Appearance list, select the appearance that you want to edit.
4. In the list of parameters, edit the parameters as desired. You edit parameters by:
   - Choosing options from a list.
   - Select or clearing check boxes.
   - Typing new values.
   - Clicking buttons to add, insert, and delete new parameters.

23.6.1 Visualizing Character Structure

You can display the following aspects of character structure:

- Character Bounds. Display the bounding volume.
- Skeleton. Display the skeleton.
- Link Names. Display the names of the skeleton's links.
- Connection Names. Display the names of nodes that have been added to the model for use at runtime. These nodes might not have a joint.
- Vehicle Footprint. Display the areas where the vehicle intersects the terrain.

➤ To visualize character structure, select or clear the options in the Visualization menu.

23.6.2 Editing Configuration File Text

The Appearance Editor displays the text of the configuration file that is being edited in the editor. You can edit the configuration file text directly. If you edit the configuration file text directly, you can refresh the editing GUI. You can control the amount of detail shown in the editing window, as follows:

- Display partial or complete appearance. By default, the window only shows the leaf-level parameters for the appearance. To show the parameters that the character inherits from the base appearance, show the complete appearance.
- Shape sets. By default, shape sets are not displayed in the editing window.

➤ To show the complete appearance, select the Show Complete Appearance check box.

➤ To show shape sets, select the Show Shape Sets check box.

➤ To refresh the editing GUI, click Reload Appearance Text.

23.6.3 Editing Appearance Effects

You can add effects to an appearance, such as dust, brake lights, flames, and rotor wash. The Appearance Editor lets you edit the supplied appearance effects and add more. You can then add them in the Appearance Effects line of the Appearance section of the editor.

➤ **To edit appearance effects:**

1. Select the Appearance Effects check box. The editor displays a list of shared appearance effects.
2. Edit the effects. The GUI works the same way as it does for editing an appearance.

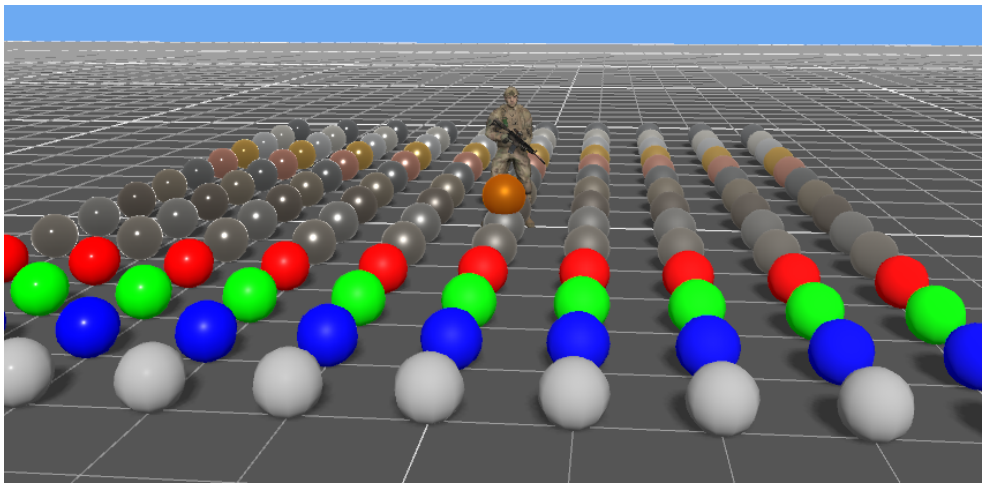
➤ To return to editing the appearance, clear the Appearance Effects check box.

23.7. Using the PBR Material Tester

The PBR Material Tester is an experimental tool for manipulating PBR shaders. (Experimental tools are not as fully tested as other features.) It adds a grid of cylindrical or spherical shapes to the Character Viewer window. You can adjust the following shader characteristics:

- Color
- Metalness
- Specular reflection
- Reflectivity
- Diffuse
- Shininess
- Ambient
- Emission
- Alpha.

Figure 145. PBR Material Tester

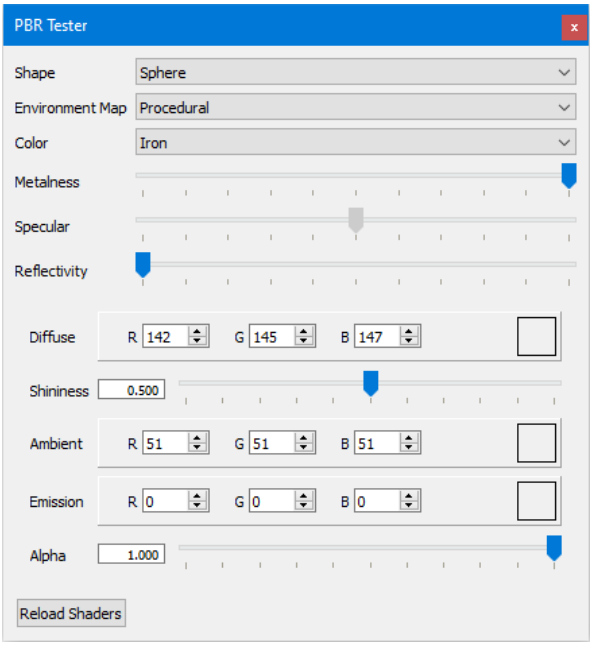


If you change shaders offline, you can reload them from the PBR Tester dialog box.

➤ **To use the PBR Material Tester:**

1. Choose **Window > PBR Material Tester**. A grid of spheres is added to the window and the PBR Tester dialog box opens (Figure 146).

Figure 146. PBR Tester dialog box



2. Adjust settings as desired.

23.7.1 Reloading Shaders

If you edit shaders, you can reload them without restarting the Character Viewer.

➤ **To reload shaders.**

1. Choose **Window > PBR Material Tester**. A grid of spheres is added to the window and the PBR Tester dialog box opens (Figure 146 above).
2. Click Reload Shaders.

23.8. The Character Viewer Log Window

The Character Viewer has a log that records system messages while you are using it.

➤ To open the Log window, choose **Window > Log**.

23.8.1 Setting the Notification Level

The Log window displays messages at the notification level that you specify. The levels are:

- Errors only.
- Errors and warnings.
- Informational messages and higher.
- Debug messages and higher.
- Everything.

➤ To set the notification level, choose an option from the list in the Log window.

23.9. Broadcasting the Character Viewer over the Network

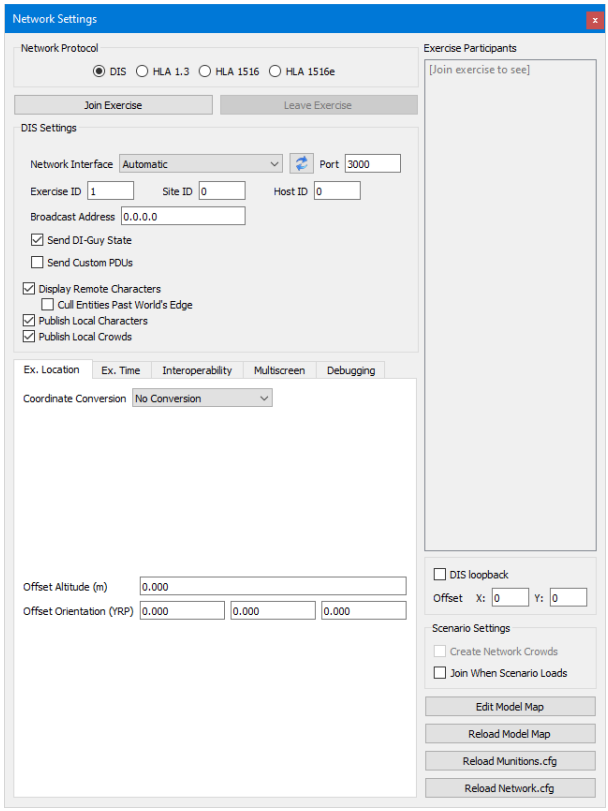
You can broadcast the state of the character in the Character Viewer over DIS or HLA. This lets you view the character within the context of the terrain in an image generator or similar 3D viewing application.

23.9.1 Joining a DIS Exercise

➤ **To join an exercise using DIS:**

1. Choose **Window > Network Settings**. The Networking Settings dialog box opens.
2. In the Network Protocol group box, select DIS (Figure 147).

Figure 147. Networking Settings dialog box for DIS



3. In the Network Interface list, select the network interface address that matches the one you are using for your IG's DIS connection.
4. In the Exercise ID box, type the number of the exercise you want to join. Exercise 0 connects you to all DIS traffic on the network. Exercise 1 or above connects you to exercises that use the same exercise number.
5. In the Port box, set the port to match the port you are using for your IG.
6. Select the Publish Local Characters to Network check box.
7. Select the Ex. Location tab.
8. In the Coordinate Conversion list, select Flat Earth.

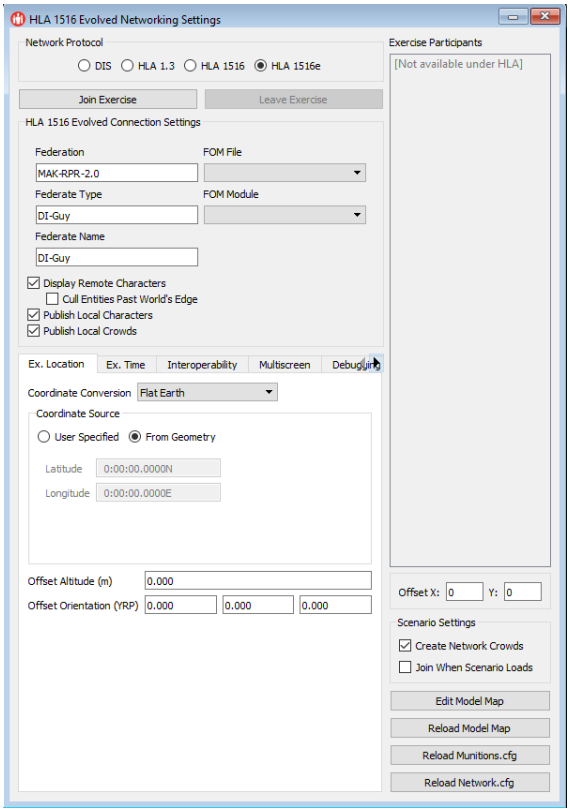
9. In the Coordinate Source group box, select User Specified.
10. Enter the latitude and longitude at which you want to view the character. (If you set latitude, longitude, and altitude to 0, the character stands in the ocean on the equator, just west of Africa.)
11. In the Offset Altitude box, specify an altitude for the character.
12. At the top of the dialog box, click Join Exercise.

23.9.2 Joining an HLA Exercise

➤ **To join an exercise using HLA:**

1. Choose **Window > Network Settings**. The Networking Settings dialog box opens.
2. In the Network Protocol group box, select one of the HLA options (Figure 148).

Figure 148. Networking Settings dialog box for HLA Evolved



3. In the HLA Connection Settings group box, set the Federation to match the Federation value in the connection you are using in your IG.
4. In the FED File list, select that same FED file as the one you are using in your IG.
5. Select the Publish Local Characters to Network check box.
6. Select the Ex. Location tab.
7. In the Coordinate Conversion list, select Flat Earth.

8. In the Coordinate Source group box, select User Specified.
9. Enter the latitude and longitude at which you want to view the character. (If you set latitude, longitude, and altitude to 0, the character stands in the ocean on the equator, just west of Africa.)
10. In the Offset Altitude box, specify an altitude for the character.
11. At the top of the dialog box, click Join Exercise.





24. Adding DI-Guy Models

This chapter explains how to take DI-Guy models created with the DI-Guy SDK and integrate them into VR-Forces and VR-Vantage.

24.1. How MAK ONE Applications Map DI-Guy Models 426
24.2. Adding New DI-Guy Models to VR-Forces 426
24.2.1 Copy DI-Guy Files to the Data Directory 426
24.2.2 Add a New DI-Guy Entity in the Simulation Object Editor 427
24.2.3 DI-Guy Configuration Files 427
24.3. Adding New DI-Guy Models to VR-Vantage 428
24.3.1 Copy DI-Guy Files to the Data Directory 429
24.3.2 Add Model Definitions and Element Definitions 429

24.1. How MAK ONE Applications Map DI-Guy Models

When VR-Vantage discovers a lifeform entity on the DIS or HLA networks, it chooses the animations to play for a DI-Guy based on the lifeform's speed, position, and similar attributes. However, if VR-Vantage receives a special DI-Guy state message (PDU) that dictates the lifeform's character, appearance, or animation, it will no longer try to guess and will expect to be told how to render that lifeform for the remainder of the simulation.

Ordinarily, VR-Forces chooses animations to play for a DI-Guy based on the lifeform's speed, position, and similar attributes. However, VR-Forces can specify the DI-Guy state and appearance with the Set DI-Guy Appearance set data request and the Set DI-Guy Animation task. Once you explicitly set the DI-Guy appearance or animation, VR-Forces no longer tries to choose animations and expects to be told how to render that lifeform for the remainder of the simulation.

24.2. Adding New DI-Guy Models to VR-Forces



VR-Forces supports DI-Guy characters and includes many lifeform characters. However, MAK customers sometimes want to add their own DI-Guy characters. The general process for adding a new character is as follows:

1. Create the new character using DI-Guy software and copy the required files to the *./data* directory.
2. Add a model definition and element definition for the new character.
3. If the character has animations, add them to *character_animations_table.mtl*.
4. Create a new character in the Simulation Object Editor.

24.2.1 Copy DI-Guy Files to the Data Directory

VR-Forces stores files for DI-Guy characters in *./data/Lifeforms/DIGuy* and its subdirectories. After you create a new character using DI-Guy software, you must copy the files for this character to these directories.

DI-Guy documentation recommends that all customer content be placed in *$(DIGUY)/custom*. To follow this convention, create a directory called *./data/Lifeforms/DIGuy/custom* and put your files in the appropriate subdirectories under it.

VR-Forces can visualize custom DI-Guy characters based on DIS enumerations mapped to element definitions or by recognizing custom DI-Guy PDUs.

- If you want to map your custom characters to element definitions, then you must add new simulation models in the Simulation Model Editor. This will automatically create the complementary model and element definitions and object type mappings.
- If you want to visualize characters based on custom DI-Guy PDUs without creating element definitions or object type mappings, place the DI-Guy files in the *./data/Lifeforms/DIGuy/custom* directory as described in the previous paragraph. No further configuration is needed.

24.2.2 Add a New DI-Guy Entity in the Simulation Object Editor

Add a new entity in the Simulation Object Editor. When you specify the 3D model, the list of character names is drawn from the **display-name** attribute in the entries in *./appData/settings/vrfSim/character_animations_table.mtl*. Therefore, it must have been updated before you can specify the 3D model in the Simulation Object Editor.

24.2.3 DI-Guy Configuration Files

VR-Forces can specify the character, appearance, and animation values for DI-Guy models used in 3D visualization applications such as the VR-Forces front-end 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, see "DI-Guy Animation", "DI-Guy Characteristics (Appearance, Head, Body Weight)", and "DI-Guy Hand Item" in the *VR-Forces Users Guide*.

You can specify the default character, appearance, and animation for a lifeform entity in the Simulation Object Editor. (For details, see "DI-Guy Animation" and "Editing a Lifeform's Visual Model and Animation" in the *VR-Forces Users Guide*.) The DI-Guy mappings are stored in the entity's *.entity* file.

The DI-Guy animations available to lifeforms are controlled by *./appData/settings/vrfSim/character_animations_table.mtl*. This information is used to populate the lists in the Simulation Object Editor. You can edit this file to add additional DI-Guy characters to those already configured for VR-Forces. The syntax of entries is described in the file.



Any characters you add to *character_animations_table.mtl* must be valid DI-Guy characters. You cannot arbitrarily make up new types of characters. If you create new characters using a supported version of the DI-Guy SDK, MAK ONE applications will support these characters.

Adding a New DI-Guy Animation

This section explains how to add a new DI-Guy animation to VR-Forces. It assumes that you have the DI-Guy software and have created the new animation.



The new animation must be made with the same version of DI-Guy supported by VR-Forces. Please see the *VR-Forces Release Notes* for the currently supported version.

➤ **To add a new DI-Guy animation:**

1. Create the new animation in DI-Guy. You will create an animation and add it to a character's action table. Make sure that the name of the animation file matches the name of the action you added to the action table. For example, if the name of the animation is layEgg, but you call the action lay_egg, it will not work.

   There is a 1-1 relationship between characters and their action tables. You cannot use an animation based on a chicken character in a cow character's action table.

2. Add the *motion_name*.bdm file for the animation to *./data/Lifeforms/DIGuy/motions*.
3. Add the *action_table*.cfg file to *./data/Lifeforms/DIGuy/config/diguy*.
4. Add the new animation to *./appData/settings/vrfSim/character_animations_table.mtl*. This table determines whether an animation can be assigned to a character using the DI-Guy Animation task (**is-taskable-animation True**) or whether it is used based on the entity's behavior. For example, if a character is moving faster than a certain speed, it is given a running animation. An entry for our lay egg example would look like the following:

```
(chicken
    (display-name "Chicken")
    (animations
        (animation
            (animation-name "lay_egg")
            (name-to-be-displayed "Lay an egg")
            (is-taskable-animation  True)
            (animation-variant "normal")
        )
        (animation
            (animation-name "feeding")
            (name-to-be-displayed "Feed")
            (is-taskable-animation  True)
            (animation-variant "normal")
        )
...
```

24.3. Adding New DI-Guy Models to VR-Vantage



VR-Vantage supports DI-Guy characters and includes many lifeform characters. However, MAK customers sometimes want to add their own DI-Guy characters. The general process for adding a new character is as follows:

1. Create the new character using DI-Guy software and copy the required files to the *./data* directory.
2. Add a model definition and element definition for the new character.

24.3.1 Copy DI-Guy Files to the Data Directory

VR-Vantage stores files for DI-Guy characters in *./data/Lifeforms/DIGuy* and its subdirectories. After you create a new character using DI-Guy software, you must copy the files for this character to these directories.

DI-Guy documentation recommends that all customer content be placed in *$(DIGUY)/custom*. To follow this convention, create a directory called *./data/Lifeforms/DIGuy/custom* and put your files in the appropriate subdirectories under it.

VR-Vantage can visualize custom DI-Guy characters based on DIS enumerations mapped to element definitions or by recognizing custom DI-Guy PDUs.

- If you want to map your custom characters to element definitions, follow the procedures in this chapterthe topics in this section.
- If you want to visualize characters based on custom DI-Guy PDUs without creating element definitions or object type mappings, place the DI-Guy files in the *./data/Lifeforms/DIGuy/custom* directory as described in the previous paragraph. No further configuration is needed.



If you want to be able to create custom DI-Guy characters in a VR-Forces scenario, then you must add new simulation models in the Simulation Model Editor. This will also automatically create the complementary model and element definitions and object type mappings.

24.3.2 Add Model Definitions and Element Definitions

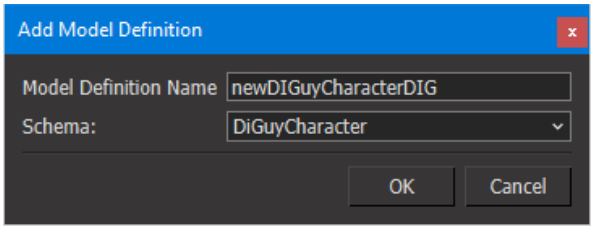
To visualize a DI-Guy character in MAK ONE applications, you must add a model definition and element definition for each character.

Create a DI-Guy Model Definition

➤ **To create a model definition for a DI-Guy:**

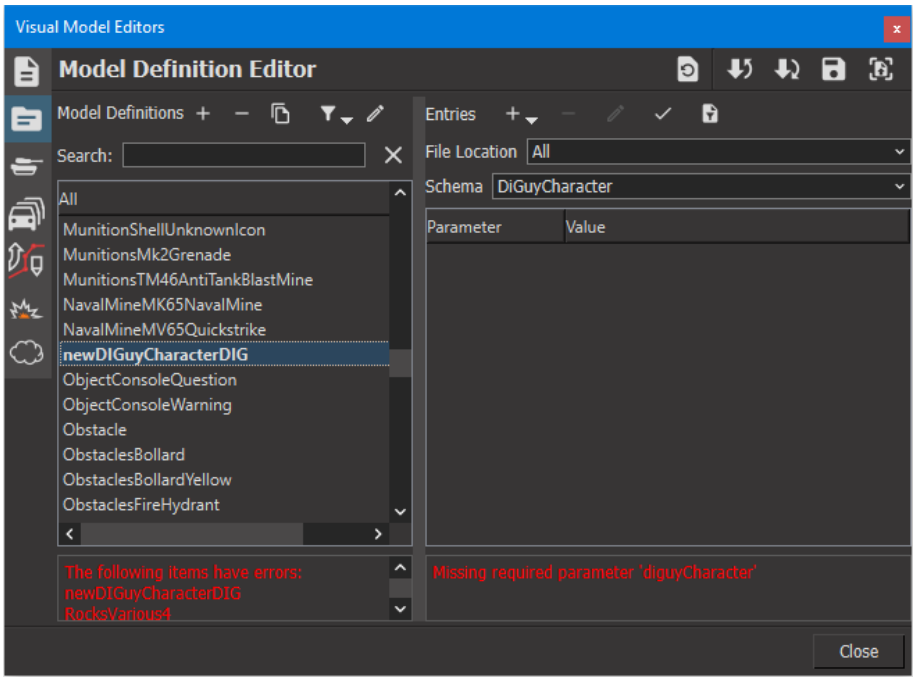
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 80 on page 218).
3. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens.
4. Type a name for the new model definition. It must end with the letters DIG.
5. In the Schema list, select DiGuyCharacter (Figure 149).

Figure 149. Add Model Definition dialog box



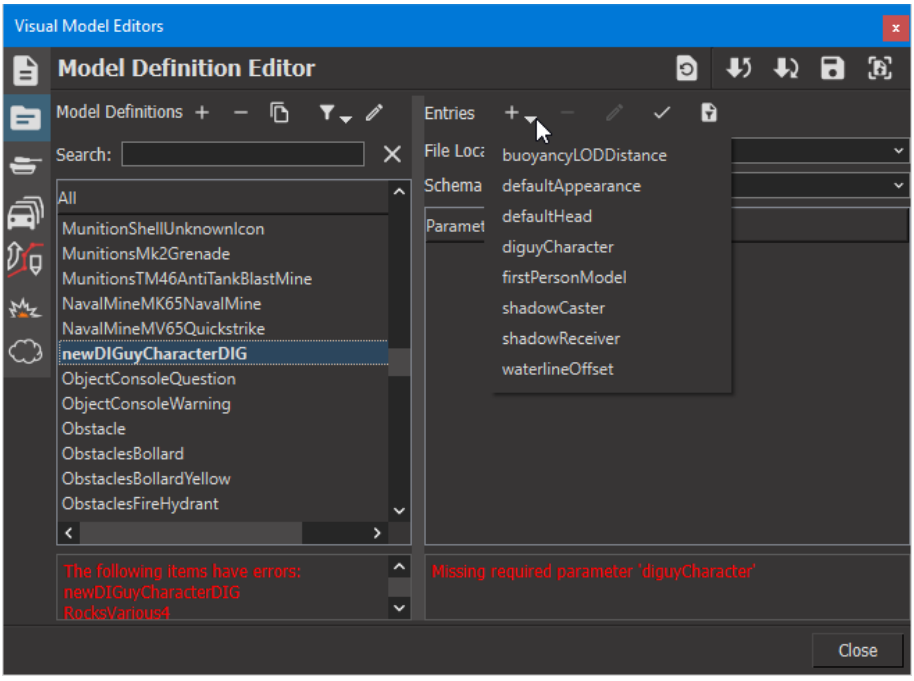
6. Click OK. The new model definition is added (Figure 150).

Figure 150. New model definition



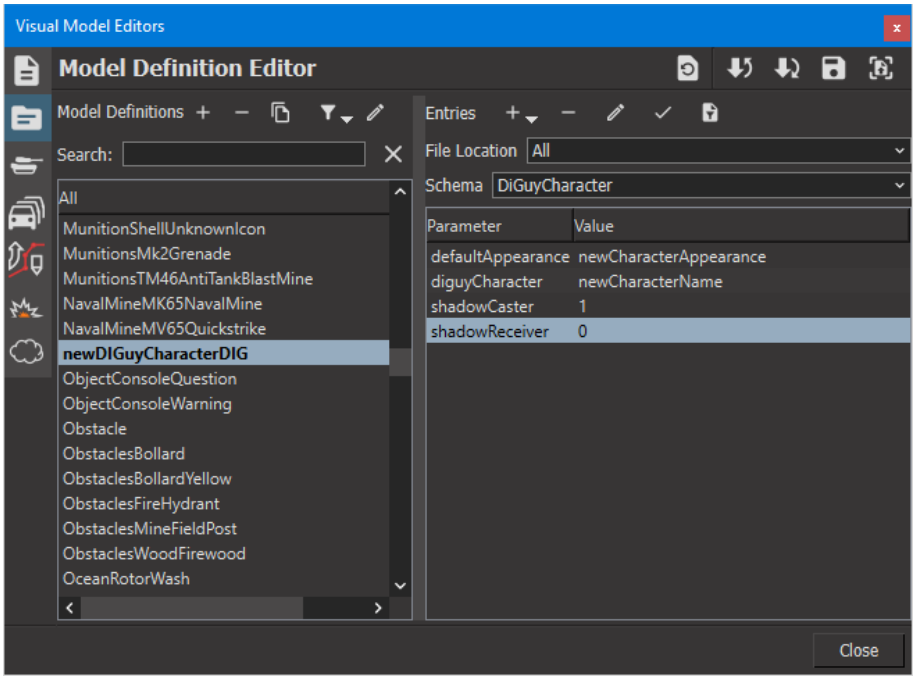
7. Click the Add Parameter button ( ). A list of parameters is displayed (Figure 151).

Figure 151. Add parameter list



8. Select defaultAppearance. It is added to the parameter list.
9. Click the value for the parameter. The field becomes editable.
10. Type the default appearance name for this character (Figure 152).
11. Add a **diguyCharacter** parameter.
12. Click the value for the parameter. The field becomes editable.
13. Type the character name for this character.
14. Add the **shadowCaster** and **shadowReceiver** parameters (Figure 152).
15. Export the new model definition to a new file.

Figure 152. DI-Guy character parameters

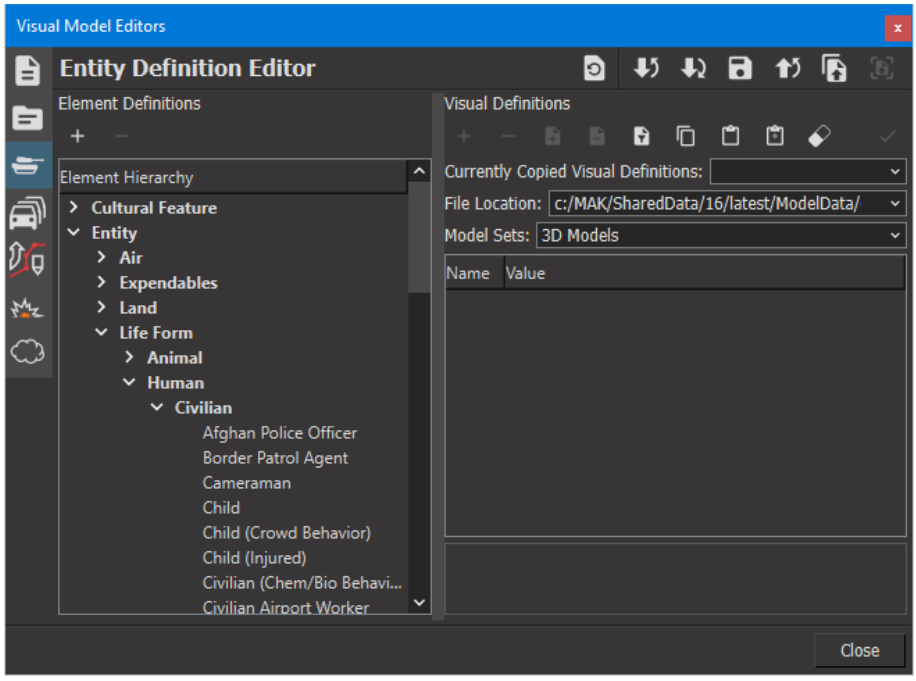


Add an Element Definition

➤ **To create an element definition:**

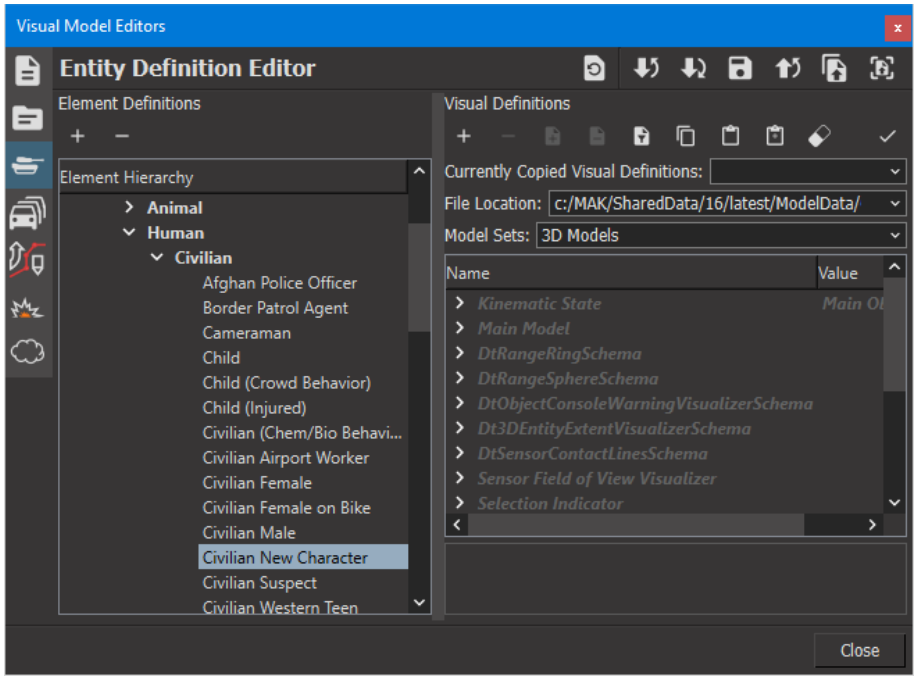
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page.
3. Expand the element definition hierarchy to the Life Form level and select the type of life form you are adding (Figure 153).

Figure 153. Entity Definition Editor page



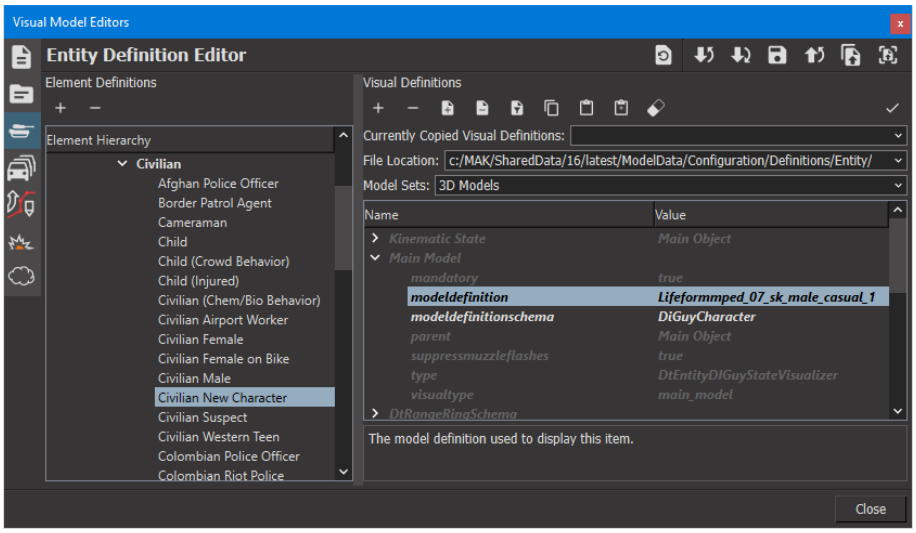
4. Click the Add button (+). The Create Element Definition dialog box opens.
5. Type a name for the element definition. (You cannot edit the name after you create it, so try to avoid typing errors.)
6. Click OK. The new element is added to the list (Figure 154).

Figure 154. New entity definition added



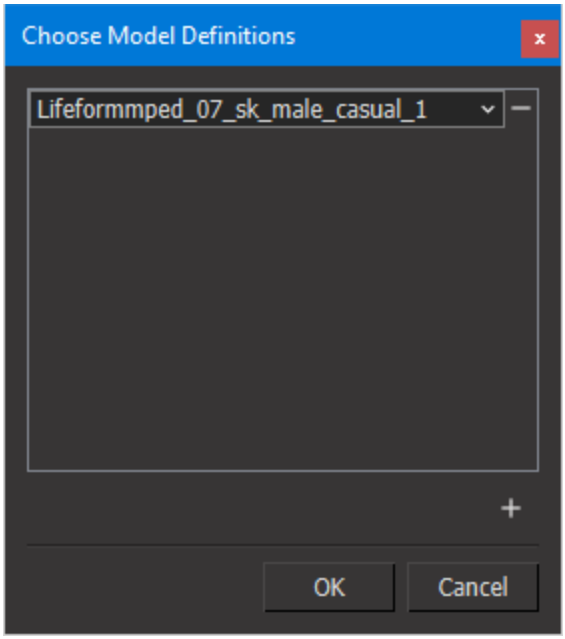
7. In the Visual Definitions list, select Main Model (Figure 155).

Figure 155. Main Model visual definition



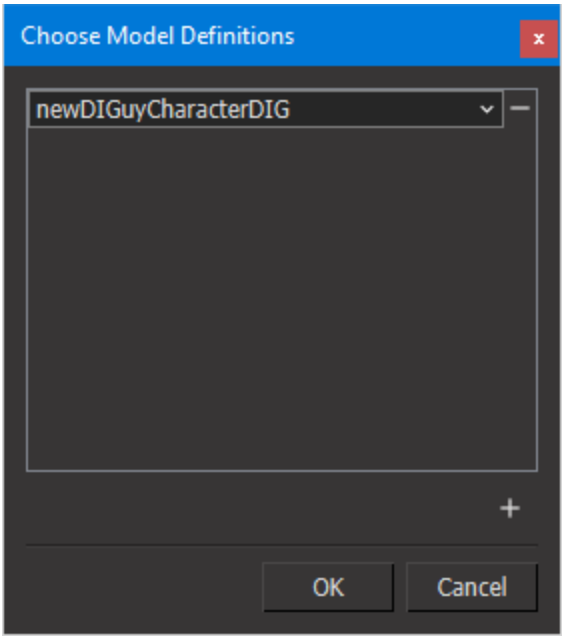
8. Click the Value for the **modeldefinition** attribute. The Choose Model Definition dialog box opens (Figure 156).

Figure 156. Choose Model Definition dialog box



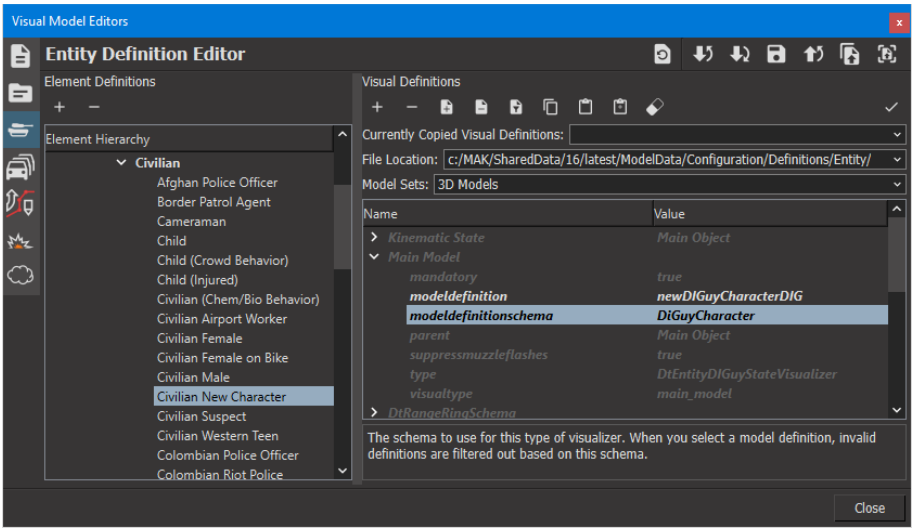
9. Select the model definition that you created for this character (Figure 157).

Figure 157. New model definition selected



10. Click OK. The model definition is updated (Figure 158).

Figure 158. Completed entity definition



11. Select the value column for the **modeldefinitionschema** attribute. The Choose Supported Model definition Schema dialog box opens.
12. Select DiGuyCharacter.
13. Click OK.
14. Close the dialog box.
15. Export the new element definition to a new individual leaf file.
16. Export the hierarchy.





25. DI-Guy Tutorials

This chapter has examples of customizing DI-Guy content.

25.1. Customizing Appearances Using Texture Swaps 438

25.1.1 Select an Appearance to Modify 439

25.1.2 Change the Texture Map 441

25.1.3 Create Your New Texture Swap and Declare Your New Appearance 442

25.1. Customizing Appearances Using Texture Swaps

Perhaps the easiest and most popular DI-Guy customization is to take an existing DI-Guy appearance and tweak its texture file without modifying geometry to create a new appearance. Uniforms can be changed to be specific to a country; clothing can match a target culture; sensorized appearances can be created to simulate UV or infrared visualization, and so on.

DI-Guy uses texture swapping to add variety to its character appearances while keeping the memory footprint small. Here are two mped_07 appearances, male_suit_1 and male_suit_2. Note how the polygons of the bodies are the same but the texture maps make it appear that the men are wearing different jackets, ties, and pants. The following sections guide you through the process of creating a new appearance.

Figure 159. Texture swapping to create different appearances



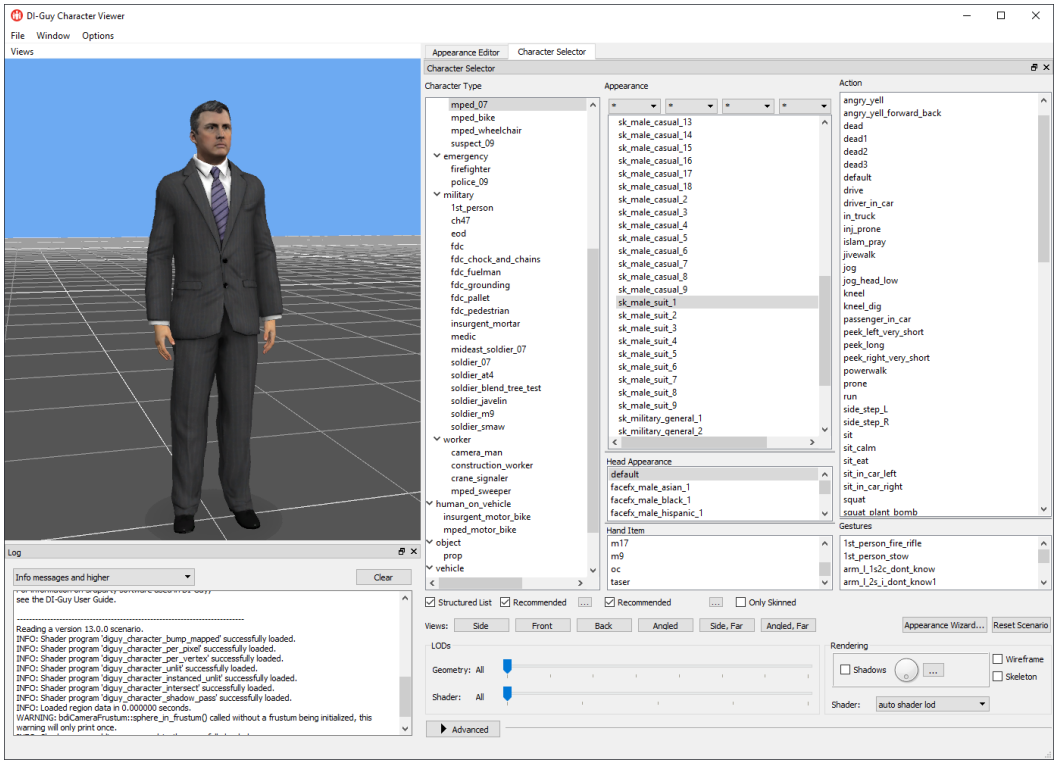
25.1.1 Select an Appearance to Modify



This tutorial references *.dae* files. The *.dae* files are in the DI-Guy CD3 installer. If you do not have the DI-Guy SDK and would like these files, please contact MAK support at support@mak.com.

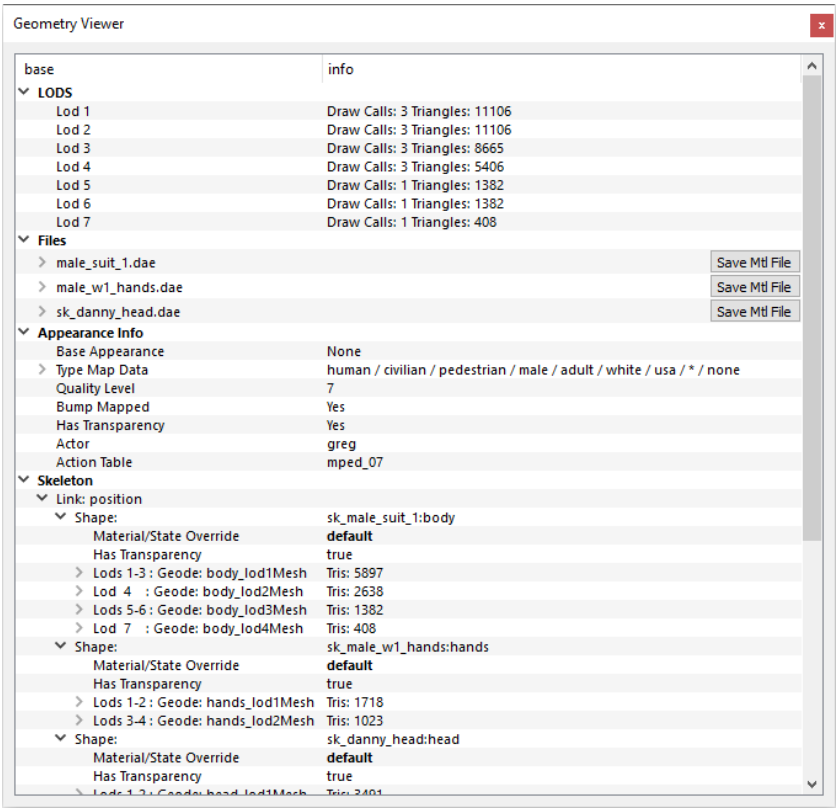
1. On the Start menu, choose **MAK Technologies** > **DI-Guy Character Viewer**. The Character Viewer opens.
2. In the Character Type list, select mped_07.
3. In the Appearance list, select sk_male_suit_1.
4. Hold down the left and right mouse buttons, and rotate the character so that it faces you (Figure 160).

Figure 160. Character viewer



5. Select the sk_male_suit_5 appearance to see the differences.
6. Choose **Window** > **Geometry Viewer**. The Geometry Viewer window opens (Figure 161).

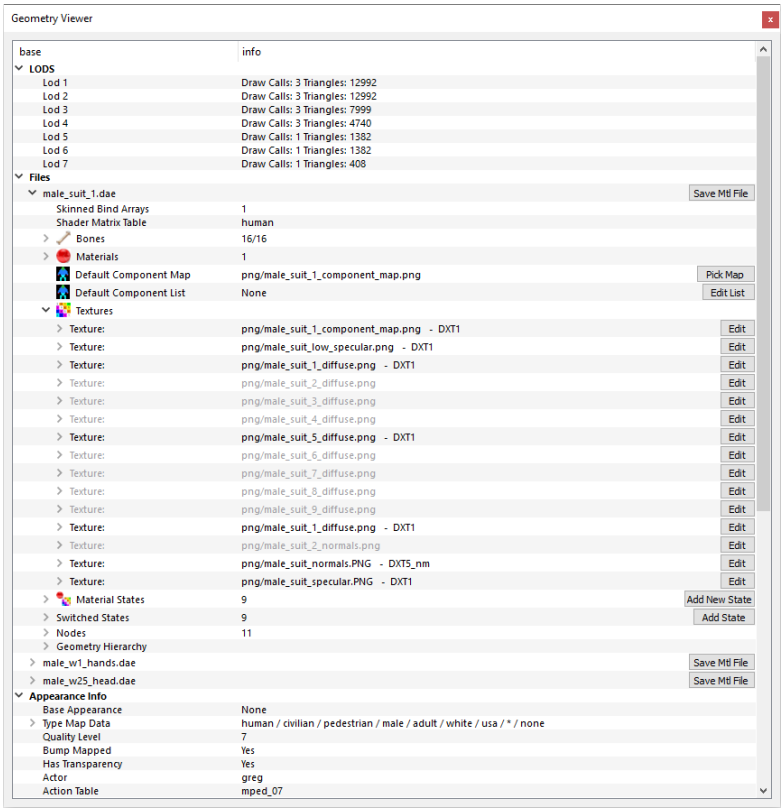
Figure 161. Geometry Overview window



The Files section lists three files. Their names indicate that they are for the suit, hands, and head. For our customization, we will make a simple change to male_suit_1. First we need to find the texture map that male_suit_1 uses.

7. Expand the *male_suit_1.dae* entry and then expand the textures entry. You can see there are many textures associated with the character. The textures in bold indicate textures already loaded into memory. In the image, *male_suit_1_diffuse.png* and *male_suit_5_diffuse.png* are also highlighted because they were visualized when you switched appearances. The naming conventions indicate how the textures are used. The convention _diffuse is used for the diffuse texture, _normals for the normal map, and _specular for the specular map.

Figure 162. male_suit_1 textures



8. We need to make a copy of the male_suit_1 configuration file. However, to do that, the *.dae* file must be in the *./data/Lifeforms/DI-Guy/custom* directory. (If this is the first time you have created a custom configuration, you may need to create the *custom* directory.)
    a. In *./data/Lifeforms/DI-Guy/custom/geometry*, create a new directory called *dae*.
    b. Copy *./data/Lifeforms/DI-Guy/geometry/dae/male_suit_1.dae.enc* to *./custom/geometry/dae/male_suit_1.dae.enc*.
9. In the Geometry Overview window, on the *male_sui_1.dae* line, click Save Mtl File. You are prompted to save to the custom directory and may receive informational prompts.
10. Click Yes. The file *./custom/geometry/dae/male_suit_1_mtl.cfg* gets created.
11. Use a text editor to open the file *male_suit_1_mtl.cfg*. Near the top of the file, are some texture entries. One of them is *male_suit_1_diffuse.png*. We will edit this file.

25.1.2 Change the Texture Map

To change the texture map, you need a paint program, such as Photoshop or Gimp.

1. In your paint program, open *./geometry/png/male_suit_1_diffuse.png* (Figure 163). The tie is in the upper right quadrant of the file.

Figure 163. *male_suit_1_diffuse.png*



2. Change the tie's stripes to yellow.

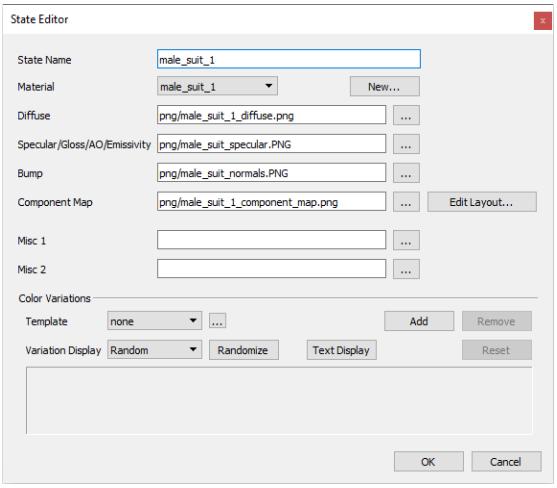
   Be careful not to change the topology. That is, do not move any of the individual images. The x,y coordinates of the image map to the u,v of the target mesh. Changing the topology would require a re-mapping.

3. Create a new directory *./custom /geometry/png*.
4. Save the texture as *male_suit_1_g_diffuse.png* in *./custom /geometry/png*.

25.1.3 Create Your New Texture Swap and Declare Your New Appearance

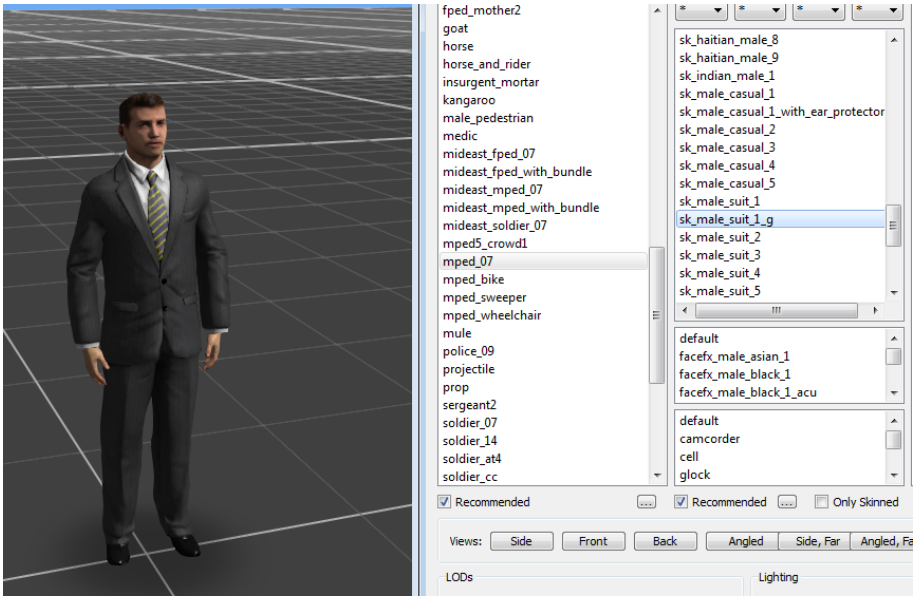
1. In the Geometry Overview, in the Material States line for male_suit_1.dae, click Add New State. The State Editor opens (Figure 164).

Figure 164. State Editor



2. Set the State Name to “male_suit_1_g” and browse *./custom /geometry/png/male_suit_1_g_diffuse.png*.
3. Click OK. The *male_suit_1.dae* is highlighted and has an asterisk.
4. Click Save Mtl File. You are prompted to save to the custom directory.
5. Click Yes.
6. Create a new appearance configuration entry for your character.
    a. In a text editor, open *./config/diguy/appearances_skinned.cfg*.
    b. Copy the multi-line entry for appearance sk_male_suit_2 to a new file.
    c. Save the new file as *./custom/config/autoload_my_custom_ appearances.cfg*.
    d. Change the name of that entry to sk_male_suit_1_g.
    e. Change the head back to sk_male_w1_head (or leave it as is if you like).
    f. Change the graphics_state_switch_map1 line to use male_suit_1_g.
    g. Delete the graphics_state_switch_map2 line.
7. Restart the Character Viewer. The new appearance should now be available with a beautiful new tie (Figure 165). Compare it to Figure 160 on page 439 .

Figure 165. Character with modified tie





MAK Products Glossary

This glossary defines terms used in MAK product documentation.

A

absolute dead reckoning

A method of dead-reckoning in which an application uses the time stamps contained within state updates if the sender is using absolute time stamping. Contrast with relative dead reckoning.

aggregate

The combination of individual entities, aggregates, or both into a single object — for example, an organizational group such as a platoon.

API

Application Programming Interface.

articulated part

A part of an entity that is capable of movement relative to the entity, such as a wheel, turret, or gun.

attached part

An object that is attached to another object, such as a rocket attached to a jet.

B

body coordinate frame

A coordinate framework centered on an entity. To represent the orientation of an entity, an application uses Euler angles or a rotation matrix that rotates from a reference frame to the body coordinate frame.

C

channel

A channel renders the view of an observer in a 3D visualization application.

clipping

A feature that prevents the display of terrain and objects or parts of objects, that are closer than or farther than specific distances from the observer.

clipping planes

The range of distances from the observer (near clipping and far clipping) in which an application clips (does not display) objects.

culling

The process of discarding database objects which are not within view, and therefore need not be rendered and displayed.

Cartesian coordinates

A system for indicating location by means of three planes intersecting at right angles to one another at the origin.

D

Data Distribution Management

The set of High-Level Architecture services used to specify the routing of data between publishers and subscribers. DDM adds greater control over the Declaration Management (DM) services that only specify filtering based on the class of the data (e.g., ground vehicle versus aircraft). DDM uses the data content to express regions of interest and regions of affect whose overlap establish a communication channel. An example is geographic filtering where a publisher indicates a region around the entity or effect and a subscriber indicates a region expressing its sensor range.

dead-reckoning

A process by which an application calculates the expected location of an entity during periods between state updates, based on velocity, acceleration, and rotational velocity. See also absolute dead reckoning and relative dead reckoning.

DDM

Data Distribution Management.

Defense Modeling and Simulation Office

The former name of the executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. Renamed Modeling and Simulation Coordination Office. Currently the MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness.

DIS

Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

display engine

An object in an application that can render 3D data.

Distributed Interactive Simulation

The DIS protocol is a set of standards that govern how participating applications share information about the virtual world. The protocol specifies a set of packets, called protocol data units (PDUs), that communicate this information. Each PDU identifies the sender and contains other information depending on the PDU type. The DIS protocol also specifies when and how frequently PDUs are sent.

The DIS protocol has been superseded by the High-Level Architecture for use by the Department of Defense.

DMSO

See Defense Modeling and Simulation Office.

E

element definition

In VR-Forces and VR-Vantage, an element definition specifies all of the visualizers that control the various aspects of visualizing a scene element. For example, an element definition includes the main model or military symbology, trailing effects, cockpit displays, and so on.

emitter

A simulated device on an entity that emits electromagnetic radiation, for example, radar.

entity

An element in a simulation, such as a vehicle or a person, that is represented in the simulation through the issuance of state messages.

entity maintenance

A process by which the MAK Data Logger compensates for discontinuities following a time jump. The Logger sends out interim state messages for entities that were present before the time jump so that they do not time out before the next update message arrives.

entity-type

A list of seven components as specified by the DIS protocol and the HLA RPR FOM. If none of the seven components contains a wildcard (-1) value, then the entity type refers to a specific, narrowly-defined entity. If some components contain a wildcard, then the entity type refers to a class of entities. A larger number of wildcards indicates a broader, more general class.

The components of an entity-type are:

- Entity kind
- Domain
- Country
- Category
- Subcategory
- Specific
- Extra.

environmental process

According to the IEEE 1278.1a specification (DIS), environmental process PDUs communicate simple environment variables, small scale environmental updates, and embedded processes.

Euler angles

A set of three angles used to describe the orientation of an entity as a set of three successive rotations about three different orthogonal axes (x, y, and z). These angles specify successive rotations needed to transform from a reference coordinate system

to the entity's body coordinate system.

event

An interaction between objects or between an object and the terrain, such as firing of a munition, or a collision of entities.

exercise

The DIS term for a one or more interacting simulation applications. Compare to federation execution in HLA.

exercise connection

The object (DtExerciseConn) through which a VR-Link application connects to the network. State messages are sent through the exercise connection and information about remote entities is received through the exercise connection. (All MAK products are VR-Link applications.)

exercise ID

A numeric identification for a DIS simulation exercise.

F

FED file

Federation Execution Data file. The Federation Execution Data is the subset of FOM data needed and read by the RTI. VR-Link applications also read the FED file.

federate

A connection to the RTI. Typically a single simulation application can be thought of as a federate.

federation

A group of HLA federates capable of playing in the same federation execution.

Federation Object Model

Defines the data content of a federation execution.

federation execution

The federation execution represents the actual operation, over time, of a subset of the federates and the RTI initialization data taken from a particular federation. A federation execution is the logical equivalent of a DIS simulation exercise.

field of view

Controls the perspective of the observer.

A wide field of view creates an effect like that of a wide-angle camera lens. Objects appear smaller and farther away from the observer, since the observer coverage spans a wider area. Depth become exaggerated.

A narrow field of view creates an effect like that of a telephoto lens. Objects appear larger and closer to the observer, and the overall scene depth appears flattened. The distances between objects appears compressed.

filter range

A setting that prevents distant entities from being processed while allowing distant terrain to appear normally.

FOM

See Federation Object Model.

FOM Module

In HLA Evolved, defines additional modular data content for a federation execution, usually extensions to an existing Federation Object Model. (Also supported by the HLA 1.3 and HLA 1516 versions of the MAK RTI. Per the HLA Evolved interface specification:

“A partial FOM (containing some or all OMT tables) that specifies a modular component of a FOM. A FOM module may contain classes not inherent to it but upon which the FOM module depends, i.e., superclasses to the modular components. These superclasses will be included in the FOM module either as complete or scaffolding definitions.”

frame rate

The rate at which the application displays updated images.

G

ground clamping

A process by which the a 3D visualization application keeps an entity anchored to the surface of its terrain database, regardless of the altitude data contained in its entity state message.

geocentric coordinates

A coordinate system calculated with respect to the earth’s center. The origin of the geocentric coordinate system is the center of the earth. The positive X-axis passes through the prime meridian at the equator; the positive Y-axis passes through 90 degrees east longitude at the equator; and the positive Z-axis passes though the north pole.

geodetic coordinates

A coordinate system in which position is determined relative to a reference ellipsoid, such as the surface of the earth at sea level. In MAK applications, geodetic coordinates consist of latitude and longitude in radians, and altitude in meters above the reference ellipsoid.

gridded data

Data that has been processed in a rectangular array of points, in X, Y or latitude/longitude, at which single data values define a two dimensional function. According to the IEEE 1278.1a specification, gridded data transmits information about large-scale or high-fidelity spatially and temporally varying ambient fields and about environmental processes and features.

guise

An alternative entity type used to display an object depending on the force ID. For example, a tank could look like an M1A1 to friendly forces and a T72 to hostile forces.

H

head-up display

A set of indicators and readouts superimposed onto a graphics display. Also called an overlay.

heartbeat

In DIS, the frequency with which current Protocol Data Units are sent to the network regardless of whether the entity's state has changed.

High-Level Architecture

The High Level Architecture (HLA) for simulations is a U.S. Department of Defense (DOD)-wide initiative to provide an architecture to support interoperability and reuse of simulations. The HLA supersedes DIS for the DOD.

HLA

See High-Level Architecture.

I

IG

image generator

interaction

A describing a simulation event. An interaction describes an event, it does not update an object's state.

L

Logger control PDUs

A set of DIS PDUs or HLA interactions that let you control the MAK Logger remotely.

logical range

A suite of TENA Resources, sharing a common object model, that work together for a given range event. Similar in concept to the HLA Federation.

Local RTI Component

The libraries on a computer that implement a Run-Time Infrastructure. A federate communicates with the HLA network through the LRC.

LRC

See Local RTI Component.

M

MAK Technologies Lisp

An adaptation of the Lisp language used in configuration files for MAK products.

message

A general term used to refer to DIS PDUs and HLA interactions and state updates. In TENA, a message is similar to an HLA interaction.

MIM

The MOM & Initialization Module (MIM) allows for extensions to be made to the High-Level Architecture standard MOM. It is a subset of the Federation Object Model that contains OMT tables that describe the HLA MOM. The MIM also contains additional predefined HLA constructs such as object and interaction roots, data types, transportation types and dimensions. HLA specifies a standard MIM that is incorporated into all FDDs automatically by the Run-Time Infrastructure. The standard MIM can be replaced with a user-supplied MIM containing the standard MIM plus extensions.

model definition

In VR-Forces and VR-Vantage, a model definition specifies the 3D model and other basic attributes of a model. It is referenced by an element definition.

Modeling and Simulation Coordination Office

The executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. The MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness. (Formerly DMSO)

MSCO

Modeling and Simulation Coordination Office. (Formerly DMSO.)

MTL

See MAK Technologies Lisp.

O

orientation clamping

Adjusts the pitch and roll of an entity so that it appears properly seated when the terrain is inclined. For example, if a tank is moving horizontally across the face of a hill, orientation clamping prevents the tank from appearing level, and therefore, partially embedded into the hillside. Used with ground clamping.

object

An element in a simulation that has persistence, as opposed to an interaction, which is a transient element.

object handle

An integer that an application uses to identify a particular object in Run-Time Infrastructure service calls. An object handle is meaningful only to a particular federate. The same object can be known to different federates by different object handles.

object name

A character string that can be used to identify an object. In HLA, the object name is known to the Run-Time Infrastructure, and the RTI provides functions to find out an object's name, given its handle, and vice versa. Object names can be chosen by applications that register the objects with the RTI, however if you do not want to choose names for objects, the RTI will assign names for you.

observer

In MAK 3D applications, the point of view into the scene. (Called the eyepoint in MAK Stealth 6.x and earlier.)

P

PDU

See Protocol Data Unit.

Protocol Data Unit

A unit of data message (packet) that is passed on a network between DIS simulation applications.

R

radio transmitter

A simulated device on an entity, capable of transmitting radio communications.

radio receiver

A simulated device on an entity, capable of receiving radio communications.

Realtime Platform Reference FOM

An HLA reference Federation Object Model based on the DIS protocol.

recording

A MAK Data Logger file that stores a history of the interactions in a simulation for playback.

reflected entity list

A list of entities simulated by remote applications (federates in HLA) and about which the local simulation has received information over the network.

reference FOM

A designed to be used as a whole, or with modification, by a wide variety of similar federation executions.

relative dead reckoning

A method of dead reckoning in which an application approximates the location of an object based on the local time of receipt of the last state update message. Contrast with absolute dead reckoning.

remote display engine

A display engine running in an application that is separate from the master application, and which is controlled by the master application.

RID file

See RTI Initialization Data.

RTI Initialization Data

Data required by a Run-Time Infrastructure during initialization, independent of the FOM being used. RID data is usually dependent on a specific implementation of the RTI.

RPR FOM

See Realtime Platform Reference FOM.

RTI

See Run-Time Infrastructure.

Run-Time Infrastructure

A library and other supporting software that implements the HLA interface specification. All federates communicate with one another in an HLA environment through RTI functions.

S

scene element

In VR-Vantage and VR-Forces, any object that is rendered in the scene, such as an entity model, interaction, or tactical graphic.

Simulation Interoperability Standards Organization

A group that seeks to promote modeling and simulation interoperability and reuse for the benefit of diverse M&S communities, including developers, procurers, and users, world-wide.

simulation time

In VR-Forces, simulation time is used in dead-reckoning of remote entities and thresholding of local entities. Typically, simulation time is set once during each iteration of the application's main simulation loop so that all entities are dead-reckoned based on the same value of current time.

SISO

See Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

A method of ensuring that transitions from an entity's dead-reckoned position to its actual position are not so abrupt as to be visually disconcerting.

state

The current status of an object, including location, direction of movement, extent of damage, and so on.

T

tape

An alternative term for a Logger recording. Not a physical magnetic tape.

terrain following

In a 3D visualization application, causes the observer (eyepoint) to maintain a constant distance above the terrain surface. The observer's height changes in tandem with the peaks and valleys as it passes over the geography.

timeout

The period of time in which an application continues to display an entity after that entity's update messages have stopped appearing on the network. Time-outs are usually not used in HLA because there is no set frequency (no heartbeat) for transmitting messages.

timescale

A factor by which time is accelerated or slowed during playback of a Data Logger recording.

topographic coordinates

A right-handed Cartesian coordinate system whose X-Y plane is tangent to the earth's surface at the origin, with the positive X axis pointing north, the positive Y axis pointing east, and the positive Z axis pointing down. There are an infinite number of topographic coordinate systems – one for each point on the earth's surface.

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

A method used by to smooth positional discontinuities that could occur when new state updates arrive.

U

UDP port

A network channel through which an application sends and receives data for DIS exercises.

Universal Transverse Mercator

In general, a non-cartesian coordinate system in which the X, Y, and Z correspond to easting (nearly east), northing (nearly north), and height above an ellipsoid that approximates the surface of the earth.

UTM

See Universal Transverse Mercator.

V

view control messages

A set of programmatic messages that let you control the view in VR-Vantage applications remotely.



Index

-

-- command-line option, 126
--file command-line option, 126
--generateFiles command-line option, 126
--help command-line option, 126
--ignore_rest command-line option, 126
--ignoreNonBaseTex command-line option, 290
--level command-line option, 126
--mergeHierarchyEntityDef command-line option, 245
--mergeHierarchyTacticalGraphicsDef command-line option, 245
--mergeHierarchyUnitDef command-line option, 245
--mergeLeafEntityDef command-line option, 245
--mergeLeafTacticalGraphicsDef command-line option, 245
--mergeLeafUnitDef command-line option, 245
--mergeModelDefs command-line option, 227
--missing command-line option, 126
--noAppDataEntityDefs command-line option, 245
--noAppDataModelDefs command-line option, 227
--noAppDataTacticalGraphicsDefs command-line option, 245
--noAppDataUnitDefs command-line option, 245
--processEmats command-line option, 126
--publishMftOnly command-line option, 127
--regenerateAll command-line option, 126
--removeHighestLod command-line option, 291
--row command-line option, 127
--scaleLod command-line option, 291
--searchDir command-line option, 126
--subDir command-line option, 127
--textureSize command-line option, 127
--threads command-line option, 126
--verbose command-line option, 291
--version command-line option, 127
--vtFile command-line option, 127
-A command-line option, 126
-b command-line option, 290
-c command-line option, 126
-e command-line option, 126
-f command-line option, 126
-g command-line option, 126
-h command-line option, 126
-l command-line option, 126
-m command-line option, 126
-p command-line option, 126
-r command-line option, 126, 291
-s command-line option, 127
-t command-line option, 127
-u command-line option, 127
-v command-line option, 127
-V command-line option, 291
-w command-line option, 127
-x command-line option, 127

@

@dis comment, 96
@dis navigation_lights, 96

2

2525B icon, 257

2D
  icon
    configuring, 257
    edit, 257
2D map, 131
2D View Layers
  overview, 132
  Test-2DViewLayers terrain, 131

3

3D tile
  configuring, 85

9

900_ExtImagery.xml, 121

A

action
  viewing, 405
action table, 427
actor, 399
  appearance, 401
  configuration file, 400-401
  skeletal hierarchy, 401
adding
  attribute
    visual definition, 237, 239
  category
    Props Palette, 268
  cockpit model, 264
  DI-Guy animation, 427
  DI-Guy character, 427
  entity bitmap image, 263
  feature layer, 39
  model mapping, 249
  ocean layer, 42, 44
  parameter
    model definition, 224
    schema, 214
  particle system, 376
  prop, 49
    from feature layer, 51
  terrain patch to terrain, 37
  terrain server, 60
  visual definition, 234
advanced lighting, 409
aerodrome
  runway lights
    setting from panel, 150
Aeronautical Charts terrain, 31
airport
  centering on, 150
  runway lights
    setting from panel, 150
    setting in plan, 152
  setting runway lights
    in plan, 152
Airport Settings Panel, 150
Ala Moana terrain, 31
alias
  attribute transformation, 142
alpha, 288
ambient
  occlusion map, 159
Angle of Attack, 267
animation
  DI-Guy
    adding, 427
    choosing, 426
  model, 303
AOA updater, 267
appearance, 399
  actor, 401
  categories, 401
  expressive face, 401
  multiple for character type, 411
  tutorial, 438
  viewing, 405
Appearance list
  filtering, 405
appearances.cfg
  configuration file, 401
applying
  lighting, 409
applying updated model definitions, 224
area
  dynamic terrain, 138
  terrain page-in, 61
areal
  feature, 74
areal feature
  simulation terrain, 176
army mode, 411
Arrivals Hall terrain, 31
articulated part, 294, 300
atlas
  texture, 278
attenuation_distance parameter, 87

attribute
  BOYSHP, 92
  buoy, 92
    building model definition, 92
  cockpit updater, 264
  COLOUR, 92
  COLPAT, 92
  DESIGNATOR, 93
  destroyedsymbol, 263
  earth file
    differential processing, 87
  feature, 141
  imagesymbolmap, 262-263
  light characteristic, 98
  LITCHR, 98
  MAK_EARTH_FILE, 142
  MAK_LAYER, 141
  MAK_SOURCE_FILE, 142
  marker-scale, 287
  modeldefinitionschema, 261
  OBJNAM, 93
  parent, 260
  SIGGRP, 98
  signal group, 98
  signal period, 98
  SIGPER, 98
  SIGSEQ, 98
  TOPSHP, 97-98
  transformation
    alias, 142
  transforming, 142
  type, 261
  visual definition, 235
    add, 237
    delete, 239
  WidgetTheme, 261
authentication
  proxy, 60
Auto Generate, 49
auto refresh
  Simulation Terrain Debug Tool, 183
autoloading
  configuration files, 402

B

back-end
  configuring feature data, 40
  debugging simulation terrain from console, 183
Bakersfield topo map terrain, 31
beacon, 98
  model definition, 92
  seasonal, 97
  streaming, 90, 95
  topmark, 98
bead
  DOF, 300
  switch, 300
best practice
  model design, 288
bitmap
  adding new, 263
  displaying, 39
  icon, 258
blending
  LOD transition, 165
Boundary Generation Tool, 78
  running, 78
  using, 78
boundary.txt, 79
bounding volume, 410
  drawing for terrain, 170
BOYSHP
  attribute, 92
Brooklyn terrain, 31
building
  extruded, 75
  indirect
    fading, 87
  prop type, 53
  terrain, 37
bump map, 159
buoy
  attributes, 92
  daymark, 97
  designator, 93
  model definition, 90, 92
    using texture atlas, 94
  seasonal, 97
  signal sequence, 98
  streaming, 90
  topmark, 98
  unique, 94
buoyancyLODDistance parameter, 276

C

cache
  clearing, 157
  location, 155
  osgEarth Cache tool, 155
  streaming terrain, 157
caching
  earth file data, 155
  file, 154
  terrain, 155

terrain server data, 155
California terrain, 31
category
appearance, 401
Props Palette
adding, 268
CDB
configuration files, 120
earth file, 104
elevation sub-dataset, 106
lightmap, 107
lightpoint, 112, 114
properties, 114
osgEarth, 104
CDB database, 104
CDB raster driver, 105
changing
model definition
prop, 194
prop
model definition, 54
position or orientation, 54
type, 53
viewing angle, 406
ChannelKeyword parameter, 266
character
DI-Guy
adding, 427
geometry, 416
gesture
viewing, 405
quality level, 407
skinned, 400
type, 399
viewing, 405
waypoint, 400
Character Viewer
starting, 404
character_animations_table.mtl, 427
character_type
definitions, 401
character_types_list.cfg
configuration file, 401
check box
Enable Occlusion Queries, 168
child
element definition, 230
CIGI
configuring tidal stream speed and direction, 274
class
diguyApp, 416
diguyCharacter, 416
diguyScenario, 416
DtLightSignalSequenceGenerator, 98
DtLightSignalSequenceGeneratorCreator, 98
clearing
file cache, 157
clipping plane, 285
cockpit display, 248, 264
adding new, 264
element definition, 230
installing, 265
model definition
creating, 265
updater, 264
attribute, 264
collision object
displaying, 173
color
pattern, 92
COLOUR attribute, 92
COLPAT attribute, 92
combat engineering object
as feature, 146
command
diguyEncryptGeometryFile, 398
Set Runway Lights, 152
command-line argument
disableIndirectEntities, 208
command-line option
--col, 126
--encryptImages, 126
--entityDefsDir, 245
--extendLevelZero, 126
--file, 126
--generateFiles, 126
--help, 126
--ignore_rest, 126
--ignoreNonBaseTex, 290
--level, 126
--mergeHierarchyEntityDef, 245
--mergeHierarchyTacticalGraphicsDef, 245
--mergeHierarchyUnitDef, 245
--mergeLeafEntityDef, 245
--mergeLeafTacticalGraphicsDef, 245
--mergeLeafUnitDef, 245
--mergeModelDefs, 227
--missing, 126
--noAppDataEntityDefs, 245
--noAppDataModelDefs, 227
--noAppDataTacticalGraphicsDefs, 245
--noAppDataUnitDefs, 245
--processEmats, 126
--publishMftOnly, 127
--regenerateAll, 126
--removeHighestLod, 291

--row, 127
--scaleLod, 291
--searchDir, 126
--subDir, 127
--textureSize, 127
--threads, 126
--verbose, 291
--version, 127
--vtFile, 127
-A, 126
-b, 290
-c, 126
-e, 126
-f, 126
-g, 126
-h, 126
-l, 126
-m, 126
-p, 126
-r, 126, 291
-s, 127
-t, 127
-u, 127
-v, 127
-V, 291
-w, 127
-x, 127
noAppDataEntityTypeMap, 254
comment
@dis, 96
common_actors.cfg
configuration file, 400
common_characters.cfg
configuration file, 400
common_links.cfg
configuration file, 400
Component Visualization, 162
CompositeElevation, 67
CompositeImage, 67
compressing
geometry, 398
model and image files, 289
compression
texture, 156
configName parameter, 278, 280
configuration file
actor, 400-401
appearances.cfg, 401
autoloading, 402
character_animations_table.mtl, 427
character_types_list.cfg, 401
common_actors.cfg, 400
common_characters.cfg, 400
common_links.cfg, 400
default_appearance_effects.cfg, 401
default_particle_descriptions.cfg, 401
diguy.cfg, 399
diguy_actors.cfg, 400
diguy_character_type_maps.cfg, 401
diguy_characters.cfg, 401
exface_shapesets.cfg, 401
feature, 143
gestures.cfg, 401
including configuration files in, 399
loaders.cfg, 400
motex_arcs.cfg, 401
required_characters.cfg, 400
shader_matrix_index_tables.cfg, 400
shape_and_joint_data.cfg, 400
shapeset, 401
configuring
2D icon, 257
3D tiles, 85
DI-Guy
character and appearance, 427
earth file, 65
entity
swimming, 276
feature data, 140
LOD, 157
model definition, 218
schema, 210
ship wakes, 271
tire tracks, 274
constructing
terrain, 37
copying
model definition, 222
schema, 213
visual definition, 239
crater, 138
creating
element definition, 233, 432
MEDF and MEIF files, 289
model
guidelines for, 288
model definition, 218, 221, 429
cockpit display, 265
prop, 268
schema, 211
terrain, 36
custom
directory, 398
customizing
sound, 402
texture, 402

cut-in site, 78

D

DAE
  file, 400
data
  caching terrain, 155
  debug
    refreshing\, 173
  feature
    configuring, 140
  S-57, 48
  streamed
    processing, 69
  vertical, 75
data_version, 400
database
  CDB, 104
  terrain
    provided, 30
dataset
  CDB
    geospecific, 110
  elevation
    CDB, 104
  extended imagery
    CDB, 118
  geotypical
    CDB, 116
  imagery
    CDB, 106
  material
    CDB, 108
  virtual and tiled, 125
Datasets.xml, 120
date
  displaying later based on, 86
daymark
  buoy, 97
daymark.dtxc, 97
daymark_attr_map.csv, 97
daymark_layer_list.csv, 97
DDS texture, 157, 167
  flipping, 284
debris, 376
debug data
  refreshing, 173
debug mode
  osgEarth terrain, 158
Debug Simulation Terrain Tool, 172
debug visuals
  particle system, 385
debugDrawColors.csv, 173
debugging
  earth file, 157
  NavMesh, 178
  shaders, 162, 410
  simulation terrain, 172
  terrain, 166
decal effects, 248
default
  particles, 401
default_appearance_effects.cfg
  configuration file, 401
default_particle_descriptions.cfg
  configuration file, 401
definition
  character_type, 401
  element, 230
  model
    buoy, 90, 92
DegPerKPH parameter, 277
degrees of freedom, 300
DelayBeforeRot parameter, 277
deleting
  earth file property, 102
  element definition, 241
  model definition, 222
  model mapping, 253
  parameter
    model definition, 225
    schema, 215
  schema, 213
  terrain
    cached, 157
depth
  ocean, 48
designator
  buoy, 93
DESIGNATOR
  attribute, 93
destroyedsymbol attribute, 263
destructible
  terrain, 31
detonation
  effects, 248
developer debug overlay, 112
DI-Guy
  action table, 427
  adding new characters, 427
  animation
    adding, 427
  choosing animation, 426
  configuring
    character and appearance, 427

model definition, 221
motion file, 427
swimming, 276
DI-Guy Motion Editor, 402
dialog box
Display Settings
Render Settings page, 168
Experimental Features, 164
Terrain Settings
Edit Existing Props page, 53-54, 194
diguy.cfg
configuration file, 399
diguy_actors.cfg
configuration file, 400
diguy_character_type_maps.cfg
configuration file, 401
diguy_characters.cfg
configuration file, 401
diguyApp class, 416
diguyCharacter class, 416
diguyEncryptGeometryFile
command, 398
diguyScenario class, 416
directory
custom, 398
DirectX, 167
DIS
encoding information in OpenFlight format, 294
WRM specification, 293
disableIndirectEntities command-line argument, 208
disabling
indirect rendering, 165, 289
entity, 208
model layer, 121
terrain layer, 101
texture compression, 156
Display Settings dialog box
Render Settings page, 168
displaying
collision object, 173
elevation tile, 180
NavMesh, 178
query area, 179
raster maps, 39
terrain layer
showing
terrain layer, 101
terrain normal, 173
terrain skin, 173
DLL
cockpit display, 264
installing, 265
DOF bead, 300
driver
OGR, 72
osgEarth, 68
simple, 88
DTED
loading multiple, 69
DtLightSignalSequenceGenerator class, 98
DtLightSignalSequenceGenerator.h header file, 98
DtLightSignalSequenceGeneratorCreator class, 98
dtxc
file, 280
duration parameter, 87
dust, 376
dynamic feature, 146
dynamic ocean
layer, 41
wakes, 271
dynamic terrain
craters, 138
switch type, 138
dynamic terrain area, 138

E

earth file, 65
adding feature layer, 70
adding mask, 77
areal features, 74
buoys and beacons, 90
caching, 155
CDB, 104
debugging, 157
differential processing of attributes and elements, 87
example, 65
including files in, 67
land use layer, 98
layer
displaying based on date, 86
loading DTED, 69
template, 67
earth gravitational model, 86
Edit Existing Props page, 53-54, 194
editing
element definition, 234
icon
2D, 257
keyframe, 378
model definition, 218, 223
model mapping, 250

parameter
model definition, 225
schema, 215
particle system properties, 377
terrain
property, 102
terrain in terrain editor, 101
visual definition, 235
parameter, 236
effect
lighting, 159
mapping, 248
model, 202
special, 376
EGM2008, 86
EGM84, 86
EGM96, 86
elem file, 241
element
earth file
differential processing, 87
fading, 87
feature_fading, 87
mask, 77
model, 70, 88, 287
option, 85
parent, 235
element definition, 230
adding visual definition, 234
changing file location, 240
create, 233
creating, 432
deleting, 241
editing, 234
export, 241
exporting hierarchy and leaves, 242
filter list, 234
filter list in Simulation Object Type Mappings, 251
import, 244
importing, 243
interaction
export, 241
jump from object type mapping, 252
jumping to model definition, 240
save, 242
simulation object, 230
elevation
layer, 75
elevation data
EGM, 86
elevation dataset
CDB, 104
elevation sub-dataset
CDB, 106
elevation tile
displaying, 180
Enable Occlusion Queries check box, 168
enabling
terrain layer, 101
texture compression, 156
encrypting
geometry, 398
entity
adding to Props Palette, 219
appearance, 294
element definition, 230
icon
adding new image, 263
indirect rendering, 208
model, 202
surface
configuring wake, 271
swimming, 276
type
enumeration, 249
Entity Image Symbol
visual definition, 257-258, 263
enumeration, 249
environment variable
GDAL_CACHEMAX, 157
OSG_CURL_PROXY, 60
OSG_CURL_PROXYPORT, 60
OSG_MAX_PAGEDLOD, 157
OSGEARTH_CACHE_PATH, 155
OSGEARTH_CURL_PROXYAUTH, 60
exface_shapesets.cfg
configuration file, 401
Experimental Features dialog box, 164
explosion, 376
exporting
element definition, 241
hierarchy, 241
interaction element definition, 243
model definition, 226
object type mapping, 253
props to shapefile, 54
tactical smoke element definition, 241, 243
expressive face
appearances, 401
extended imagery dataset
CDB, 118
extended imagery sub-dataset
CDB, 119
external reference, 288

extracting
  prop
    from terrain patch, 49
  water texture, 42
extruded
  building, 75

F

fading
  indirect layers, 87
fading element, 87
feature
  areal, 74
  attribute, 141
  configuration file, 143
  data
    buoys and beacons, 90
    configuring, 140
    configuring in front-end and back-end, 40
    extracting from GDB, 144
    layer, 144
    ocean, 48
    processing, 69
  dynamic, 146
  extracting as prop, 49
  geometry layer, 69
  query, 143, 176
  querying, 141
  simulation terrain, 176
  SwitchStateMapper, 279
  UvAtlasMapper, 279
feature layer, 90
  adding, 39
  adding prop from, 51
  adding to earth file, 70
  name
    specifying, 140
Feature_Data_Dictionary.xml, 120
feature_fading element, 87
feature_geom, 65, 69, 73, 132, 187
featureconfig.txt
  mapping
    features, 40
featureDataBroker, 69
FeatureImage, 67
Feet Above Ground Level, 267
Feet Above Mean Sea Level, 267
FeetAGL updater, 267
FeetMSL updater, 267
FeetPerMinute updater, 267
file
  cache
    clearing, 157
  caching, 154
  compressed, 289
  custom
    directory, 398
  DAE, 400
  dtxc, 280
  geometry, 278
file location
  change for object type mapping, 252
  element definition
    changing, 240
Filename parameter, 266
filename_shp_map.txt configuration file, 40
filtering
  Appearance list, 405
  element definition list, 251
  list of element definitions, 234
  model definition list, 223
fire, 376
firing
  effects, 248
flag
  movement in wind, 276
flaming effects, 248
flipping
  DDS textures, 284
FLT file, 202
format
  OpenFlight, 294
frame rate
  improving, 168
fxSystem parameter, 379

G

GDAL, 144
  configuring cache, 157
GDAL_CACHEMAX
  environment variable, 157
GDALElevation, 67
GDALImage, 67
GDB
  extracting feature data, 144
GDB terrain
  shapefile, 39
geocentric
  terrain, 58
geometry, 288
  character, 416
  compressing, 398

encrypting, 398
file, 278
Geometry Grid Dataset, 125, 128
geometry processor, 398
geometry_set_priority_list, 400
geospecific airport lightpoint sub-dataset
CDB, 112
geospecific dataset, 110
geospecific environmental lightpoint sub-dataset
CDB, 114
geospecific sub-dataset
CDB, 111
geotypical dataset
CDB, 116
geotypical sub-dataset
CDB, 118
gesture, 401
character
viewing, 405
gestures.cfg
configuration file, 401
GGDS, 128
GL Studio, 264
group
map, 132
guidelines, 288
creating models, 288

H

head-up display, See cockpit display
header file
DtLightSignalSequenceGenerator.h, 98
Headlight updater, 267
height
ocean, 43-44
height map
ocean
regenerating, 47
resolution, 46
texture size, 46-47
hiding
terrain layer, 101
hierarchy
element definition, 230
exporting, 241-242
importing, 243
hierarchy file
importing, 244
HUD, See cockpit display

I

icon
2525B, 257
2D, 258
bitmap, 258
edit, 257
bitmap
adding new, 263
configuring 2D, 257
image, 258
adding new, 263
image
adding, 263
compressing, 289
displaying, 39
icon, 258
imagery dataset
CDB, 106
imagery lightmap sub-dataset
CDB, 107
imagery sub-dataset
CDB, 106
imagesymbolmap attribute, 262-263
ImGui overlay
disable indirect rendering, 165
importing
element definition, 243-244
hierarchy, 243
hierarchy file, 244
leaf file, 244
leaf files, 243
model definition, 226
props from shapefiles, 55
SpeedTrees from shapefile, 55
including XML files in earth files, 67
indirect
layers
fading, 87
indirect rendering, 118, 164, 166
disabling, 165, 289
entity, 208
LOD distance scale, 164
LOD transition blending, 165
infrared
light source, 271
inheritance
element definition, 230
interaction
element definition
export, 241

interaction element definition
  export, 243
intersection
  disable for feature_geom layer, 69
  disable for model layer, 121
  displaying, 181
  map layer, 132
  simulation object, 121
intersector line
  terrain, 166
isamphibious parameter, 276
IVE, 61

J

joint, 399

K

keyframe, 388, 390
  editing in particle system, 378
KilometersPerHour updater, 267
KnotsPerHour updater, 267

L

lamppost
  prop type, 53
land use data, 98
land use layer, 98
layer
  displaying based on date, 86
  earth file
    debugging, 157
  editing in Terrain Editor, 102
  elevation, 75
  enabling or disabling, 101
  fading, 87
  feature, 90
    adding, 39
    adding prop from, 51
    adding to earth file, 70
  feature data, 144
  land use, 98
  map, 132
  name, 72
    shapefile, 72
  ocean, 41
    adding, 44
leaf
  exporting, 242
leaf file
  import, 244
leaf files
  importing, 243
LettersNumbers.ctxc, 95
level-of-detail, 285, See also LOD
lifeform
  configuring DI-Guy appearance and character, 427
light
  navigation, 96
  sensor mode specific sources, 271
light characteristic attribute, 98
light point
  CDB
    properties, 114
Light_DefaultValues_3_0.xml, 121
Light_DefaultValues_3_2.xml, 120
light_layer_list.csv, 96
lighting
  advanced, 409
  aerodrome, 150
  airport runways, 150, 152
  applying, 409
  effects, 159
lightmap
  CDB, 107
lightpoint
  CDB, 112, 114
Lights_xxx.xml, 121
linear feature
  simulation terrain, 176
link, 399
list
  props, 53
LITCHR attribute, 98
Little Pond terrain, 31
loaders.cfg
  configuration file, 400
loading
  MetaFlight, 60
  models, 154
  multiple DTED files, 69
  terrain, 154
location
  cache, 155
LOD, 285, 288
  configuring, 157
  ocean elevation, 45
  scaling
    indirect rendering, 164
  specifying maximum for earth file, 166
  transition blending, 165

LOD level
viewing different, 408
Lua
script
running, 416

M

MAK Earth terrain, 31
MAK Encrypted Data Format, 128
MAK_EARTH_FILE attribute, 142
MAK_LAYER attribute, 141
MAK_SOURCE_FILE attribute, 142
Makland terrain, 32
map
texture, 159
specular, 77
map layer, 132
mapgroup, 132
mapgroup_isect, 132
mapping
model
to object type, 249
object type, 249
shapefiles to features, 40
soil types, 169
marker-scale attribute, 287
mask
adding to earth file, 77
element, 77
ocean, 171
Massachusetts S-57 terrain, 32
material dataset
CDB, 108
max_range parameter, 87
MBTilesElevation, 67
MEDF, 128
creating, 289
medfTool, 128
MEIF
creating, 289
merging
model definition, 226
metafile
texture skinning, 278-279
metaFileName parameter, 278-280
MetaFlight
building efficient terrains, 124
loading, 60
processing, 124
mftTool, 125, 127
MilesPerHour updater, 267
Military Symbol Icon
visual definition, 257
mode
army, 411
model
3D, 202
adding
as prop, 268
animation, 303
compressing, 289
creating, 288
definition
buoy, 92
DI-Guy
choosing, 426
editing model mapping, 250
element, 69-70, 287
file caching, 154
flipping DDS textures, 284
indirect
fading, 87
intersections, 181
mapping
add, 249
delete, 253
editing, 250
schema, 218
set, 234
texture, 278
texture atlas, 278
model definition
apply updated, 224
buoy, 90, 92
building from attributes, 92
texture atlas, 94
copying, 222
create, 221
creating, 218, 429
cockpit display, 265
deleting, 222
DI-Guy character, 221
edit, 223
export, 226
filtering list of, 223
import from command line, 227
importing, 226
jumping to from element definition, 240
merging, 226
parameter
adding, 224
deleting, 225
editing, 225
particle system, 379

prop
changing, 54, 194
save, 226
schema, 210
SpeedTree, 221
texture atlasing, 278
Model Definition Editor page, 226, 263, 276
model element, 88
model layer, disabling, 121
model mapping
jump to element definition, 252
Model Specification for DIS Interoperability, 294
Model_PagingDistance.xml, 120
modeldefinitionschema
attribute, 261
motex_arcs.cfg
configuration file, 401
motion, 402
file, 427
textures, 401
movement
unit
restriction, 145
MultiGen-Paradigm
OpenFlight format, 294
multiple
parameter values, 236

N

name
feature layer
specifying, 140
layer, 72
named query, 141
navigation
light, 96
streaming, 96
navigation data
regenerating, 136
navigation profile, 136
navigational aid light, 98
navigationProfiles.mtl, 136
NavMesh
debugging, 178
networking, 422-423
new
terrain, 36
night vision
light source, 271
noAppDataEntityTypeMap command-line option, 254
node
switch, 277, 279
normal map, 159

O

object
adding
to Props Palette, 219
model, 202
object type
mapping, 249
object type mapping
change file location, 252
export, 253
jump to element definition, 252
OBJNAM
attribute, 93
observer
LOD
ocean, 45
occlusion query, 168
ocean
depth, 48
height, 43-44
height map
regenerating, 47
resolution, 46
texture size, 46-47
layer, 41
adding, 42, 44
masking
Triton.config, 171
Ocean Height Map Regeneration Threshold, 47
ocean height map resolution, 46
Ocean Layer, 45
Ocean LOD elevation, 45
offset
UV, 280
OffsetX parameter, 266
OffsetY parameter, 266
OffsetZ parameter, 266
OGR, 144
driver, 72
ogr2ogr, 144
OGRFeatures, 71
OpenFlight, 202
converting texture data, 169
disabling indirect rendering for, 289
format, 294
OpenGL, 167
opening
Terrain Editor, 100

option
element, 85
orientation
prop
changing, 54
OSG_CURL_PROXY
environment variable, 60
OSG_CURL_PROXYPORT
environment variable, 60
OSG_MAX_PAGEDLOD
environment variable, 157
osgEarth, 58, 65, 157
adding terrain servers, 60
CDB, 104
debug mode, 158
driver, 68
indirect rendering, 166
osgEarth Boundary Generation Tool, 78
osgEarth Cache Tool, 155
osgEarth terrain, 98
OSGEARTH_CACHE_PATH
environment variable, 155
OSGEARTH_CURL_PROXYAUTH
environment variable, 60
osgearth_exportgroundcover tool, 82
overlay
developer debug, 112

P

page
Model Definition Editor, 226, 263
Raster Maps, 161
paged terrain
flipping DDS textures, 168
preprocessing, 61
paging
terrain
manually, 61
paper map, 131
parameter
adding
to model definition, 224
attenuation_distance, 87
buoyancyLODDistance, 276
ChannelKeyword, 266
configName, 278, 280
DegPerKPH, 277
DelayBeforeRot, 277
deleting
from model definition, 225
duration, 87
Filename, 266
fxSystem, 379
isamphibious, 276
max_range, 87
metaFileName, 278-280
model definition
edit, 223
editing, 225
OffsetX, 266
OffsetY, 266
OffsetZ, 266
particle system, 387
Perspective, 266
PerspectiveHeight, 266
PerspectiveWidth, 266
possiblegroundclampingmode, 276
RotDir, 277
RotOffsetDeg, 277
RSOName, 266
schema
add, 214
delete, 215
edit, 215
visual definition
editing, 236
waterlineOffset, 276
WindSpeed, 277
WindState, 277
parent
attribute, 260
element, 235
element definition, 230
particle
default, 401
Particle Collision Object visualizer, 393
particle system, 376
adding, 376
debug visuals, 385
keyframe
editing, 378
model definition, 379
parameters, 387
preview parameters, 392
previewing, 380
properties, 387
editing, 377
reverting changes, 379
rotor wash, 393
saving, 379
textures, 382
importing, 389
wireframe mode, 384
pasting
visual definition, 239

patch
  terrain, adding, 191
pattern
  color, 92
PBR Material Tester, 420
performance, 46
  compressing texture, 156
  frame rate, 168
  indirect rendering, 164
  intersections, 69, 121
  model
    best practices, 288
    intersections, 121
Perspective parameter, 266
PerspectiveHeight parameter, 266
PerspectiveWidth parameter, 266
Pitch updater, 267
pixel, 288
plan
  set runway lights, 152
playback control, 415
point feature
  simulation terrain, 176
polygon, 288
  underwater, 48
port
  proxy, 60
position
  prop
    changing, 54
possiblegroundclampingmode parameter, 276
preprocessing
  terrain
    paged, 61
procedural earth terrain, 32
procedural terrain, 79
  trees, 82
  vegetation
    VR-Forces, 81
processing
  MetaFlight, 124
  streamed data, 69
prop
  adding
    from feature layer, 51
    to Props Palette, 219, 268
    to terrain, 49
  creating, 268
  exporting to shapefile, 54
  extracting, 49
  importing from shapefile, 55
  model definition
    changing, 54, 194
  saving, 268
  selecting, 53
  type, 53
  viewing list of, 53
propeller wash
  configuring, 274
properties
  particle system
    editing, 377
property
  editing
    Terrain Editor in, 102
  particle system, 387
  terrain
    deleting, 102
Props Palette
  adding objects to, 219, 268
  category
    adding, 268
proxy
  connecting through, 60
  port, 60

Q

quality level
  character, 407
query
  feature, 143, 176
  named, 141
query area
  displaying, 179

R

raster driver
  CDB, 105
raster map
  displaying, 39
Raster Maps page, 161
reflection
  map, 159
reflections
  on water, 77
refreshing
  simulation terrain debug data, 173
  simulation terrain debug display, 183
regenerating
  navigation data, 136
  ocean height map, 47
registrant_feature_list.csv, 69
Render Settings page, 168

rendering
  indirect, 164
required_characters.cfg
  configuration file, 400
required_sdk_version, 400
resolution
  ocean height map, 46
  tree, 285
restriction
  unit movement, 145
reusable software object, 264
Rhone Valley terrain, 32
Roll updater, 267
RotDir parameter, 277
RotOffsetDeg parameter, 277
rotor wash particle system, 393
roughness
  soil, 173
RSO, 264
RSOName parameter, 266
running
  Boundary Generation Tool, 78
  Lua script, 416
runway lights
  setting from panel, 150
  setting in plan, 152

S

S-57
  buoys and beacons, 90
  converting to shapefile, 144
  data, 48
  signal sequence, 98
s57_attr_map.csv, 93, 95
s57_defaults.csv, 95
s57_layer_list.csv, 90, 95
s57_modeldef_by_name_map.csv, 94
Sadr City terrain, 32
San Luis Obispo terrain, 32
saving
  element definition, 242
  hierarchy and leaves, 242
  model definition, 226
  particle system, 379
  prop, 268
  schemas, 210
  terrain, 36
scale
  SpeedTree
    random, 286
scaling
  LOD distance
    indirect rendering, 164
schema, 210
  copy, 213
  creating, 211
  delete, 213
  export, 212
  model definition, 210
  parameter
    add, 214
    delete, 215
    edit, 215
  save, 212
  saving, 210
  visual definition, 210
screen capture, 28
script
  Lua
    running, 416
seasonal buoys and beacons, 97
selecting
  props, 53
sensor
  light source, 271
server
  terrain
    caching, 155
    cutting in site, 77
shader, 159
  debugging, 162, 410
  PBR, 420
shader_matrix_index_table, 401
shader_matrix_index_tables.cfg
  configuration file, 400
shadow
  viewing, 409
shape, 399-400
shape_and_joint_data.cfg
  configuration file, 400
shapefile
  buoys and beacons, 90
  converting S-57 to, 144
  exporting props to, 54
  importing props from, 55
  importing SpeedTrees from, 55
  layer name, 72
  mapping to feature, 40
shapeset, 399
  configuration file, 401
ship
  wake
    configuring, 271

SIGGRP
  attribute, 98
signal
  sequence, 98
signal group attribute, 98
signal period attribute, 98
SignificantSize_SwitchDistance.xml, 120
SIGPER
  attribute, 98
SIGSEQ attribute, 98
simple driver, 88
simulation object
  element definition, 230
  icon
    image, 263
Simulation Object Type Mappings
  filter element definition list, 251
simulation terrain
  debugging
    from console, 183
  features
    debuggin, 176
Simulation Terrain Debug Tool
  auto refresh, 183
site
  cutting in, 78
size
  SpeedTree
    randomizing, 286
skeletal hierarchy
  actor, 401
skeleton, 410
skinned
  character, 400
smoke, 376
soil
  roughness, 173
soil type, 169, 173
sound
  customizing, 402
specify effect, 376
specular
  map, 159
    texture, 77
SpeedTree, 69
  configuring, 285
  displaying in planar reflection, 288
  exporting to shapefile, 54
  importing from shapefile, 55
  model definition, 221
  scale
    random, 286
speedTreeManager, 69
standing_hip_height, 401
starting
  Character Viewer, 404
streamed data
  processing, 69
streaming
  beacon, 95
  buoys and beacons, 90
  navigation lights, 96
  terrain
    configuring cache, 157
sub-dataset
  elevation
    CDB, 106
  extended imagery
    CDB, 119
  geospecific
    CDB, 111
  geospecific airport lightpoint
    CDB, 112
  geospecific environmental lightpoint
    CDB, 114
  geotypical
    CDB, 118
  imagery
    CDB, 106
  imagery lightmap
    CDB, 107
surface
  characteristics, 169
surface entity
  wake
    configuring, 271
surfChar.map, 169
swimming entity
  configuring, 276
switch bead, 300
switch node, 277, 279
switch type
  dynamic terrain, 138
SwitchStateMapper
  feature, 279

T

tactical graphic, 248
  element definition, 230
tactical smoke element definition
  export, 241, 243
template
  earth file, 67
terrain
  2D maps, 131

2D View Layers example, 131
Aeronautical Charts, 31
Ala Moana, 31
Arrivals Hall, 31
Bakersfield topo map, 31
bounding volume
  drawing, 170
Brooklyn, 31
building, 37
caching, 154-155
California, 31
creating, 36
databases provided, 30
debug mode, 158
debugging, 166
deleting cached, 157
destructible, 31
displaying raster image on, 39
dynamic, 138
earth file
  debugging, 157
editing in terrain editor, 101
feature data
  configuring, 140
indirect rendering, 164
  disabling, 165
intersector line, 166
Little Pond, 31
MAK Earth, 31
Makland, 32
Massachusetts S-57, 32
MetaFlight
  designing, 124
  loading, 60
  processing, 124
osgEarth
  land use data, 98
  maximum paged LOD, 166
page-in area, 61
paged
  configuring LODs, 157
  flipping DDS textures, 168
  preprocessing, 61
paging
  manually, 61
  MetaFlight, 60
patch
  adding to terrain, 37
  extracting props from, 49
procedural, 79
  trees, 82
  VR-Forces, 81
procedural earth, 32
prop
  adding, 49
  list, 53
querying, 141
Rhone Valley, 32
Sadr City, 32
San Luis Obispo, 32
saving, 36
server, See terrain server
simulation
  debugging, 172
  debugging from console, 183
streaming
  cache, 157
Truman carrier, 32
underwater, 48
VR-Village, 32
world
  flat, 33
  geocentric, 33
terrain database
  provided, 30
Terrain Editor
  deleting property, 102
  editing terrain, 101
  introduction, 100
  layer
    editing, 102
    editing property, 102
  opening, 100
  types of layers, 101
terrain normal
  displaying, 173
terrain patch
  adding, 191
terrain server, 58
  adding, 60
  caching data, 155
  connecting through proxy, 60
  cutting in site, 77
  mask
    adding, 77
Terrain Settings dialog box
  Edit Existing Props page, 53-54, 194
terrain skin
  displaying, 173
terrain tile
  displaying, 158
TerraPage file, 61
texture, 288
  atlas
    buoy and beacon, 94

compression
enabling and disabling, 156
customizing, 402
data
converting, 169
DDS, 167
flipping
paged terrain, 168
map, 159
specular, 77
mapping to model, 278
model, 278
motion, 401
particles, 382
importing, 389
size
ocean height map, 46-47
skinning
metafile, 279
model, 278
tiled and virtual, 125
water, 41
texture atlas, 278
texture map, specifying, 389
texture swap
tutorial, 438
TFSFeatures, 71
tidal stream
speed and direction
configuring in CIGI, 274
wake, 274
tiled texture, 125
tire
tracks, 274
TMSElevation, 67
TMSImage, 67
topmark
buoy and beacon, 98
topmark_attr_map.csv, 98
TOPSHP attribute, 97-98
track
tire, 274
trailing effects, 248
transform
multiple, 143
transforming
attribute, 142
transition blending
LODs, 165
transparency, 288
tree
procedural, in VR-Forces, 82
prop type, 53
SpeedTree, 285
windspeed, 285
Triton.config
ocean masking, 171
propeller wash
configuring, 274
wakes from propeller wash
configuring, 274
TrueType font
2D icon, 258
Truman carrier terrain, 32
tutorial
appearance, 438
texture swap, 438
type
character, 399
prop, 53
type attribute, 261

U

unique
buoy, 94
unit
restricting movement, 145
updater, 264
AOA, 267
attribute, 264
FeetAGL, 267
FeetMSL, 267
FeetPerMinute, 267
Headlight, 267
KilometersPerHour, 267
KnotsPerHour, 267
MilesPerHour, 267
Pitch, 267
Roll, 267
Value, 267
Yaw, 267
Use Extracted Geometry, 49
using
Boundary Generation Tool, 78
UV offset, 280
UvAtlasMapper
feature, 279

V

Value updater, 267
vdatum, 75

vegetation
  procedural
    VR-Forces, 81
vehicle
  persistent tracks, 274
vertical datum, 75
viewing
  action, 405
  appearance, 405
  character, 405
  list of props, 53
  shadows, 409
  wireframe, skeleton, bounding volume, 410
viewing angle
  changing, 406
virtual texture, 125
Virtual Texture Datasets, 125
visual definition, 210, 230, 234
  adding, 234
  attribute, 235
    add, 237
    delete, 239
  clearing from memory, 240
  copying and pasting, 239
  edit, 235
  editing, 236
  Entity Image Symbol, 257-258, 263
  Military Symbol Icon, 257
  schema, 210
visual quality, 46
visualizer, 230
  Particle Collision Object, 393
VR-Village terrain, 32
vrfsim
  enabled, 88
  layer, 140
vrfsim:, 87
VTDS, 125

W

wake
  propeller wash
    configuring, 274
  ship
    configuring, 271
  tidal stream, 274
water
  reflections on, 77
  texture, 41
    extracting, 42
waterlineOffset parameter, 276
waypoint
  character, 400
WFSFeatures, 71
WidgetTheme
  attribute, 261
wind
  moving flags and windsocks, 276
  SpeedTree trees, 285
WindDirPart, 277
WindRotPart, 277
windsock
  movement in wind, 276
WindSpeed parameter, 277
WindState parameter, 277
WindSwitchPart, 277
wireframe, 410
wireframe mode
  enabling for particles, 384
WMSImage, 67
world terrains, 33
WRM Entity Model Specification, 293, 301

X

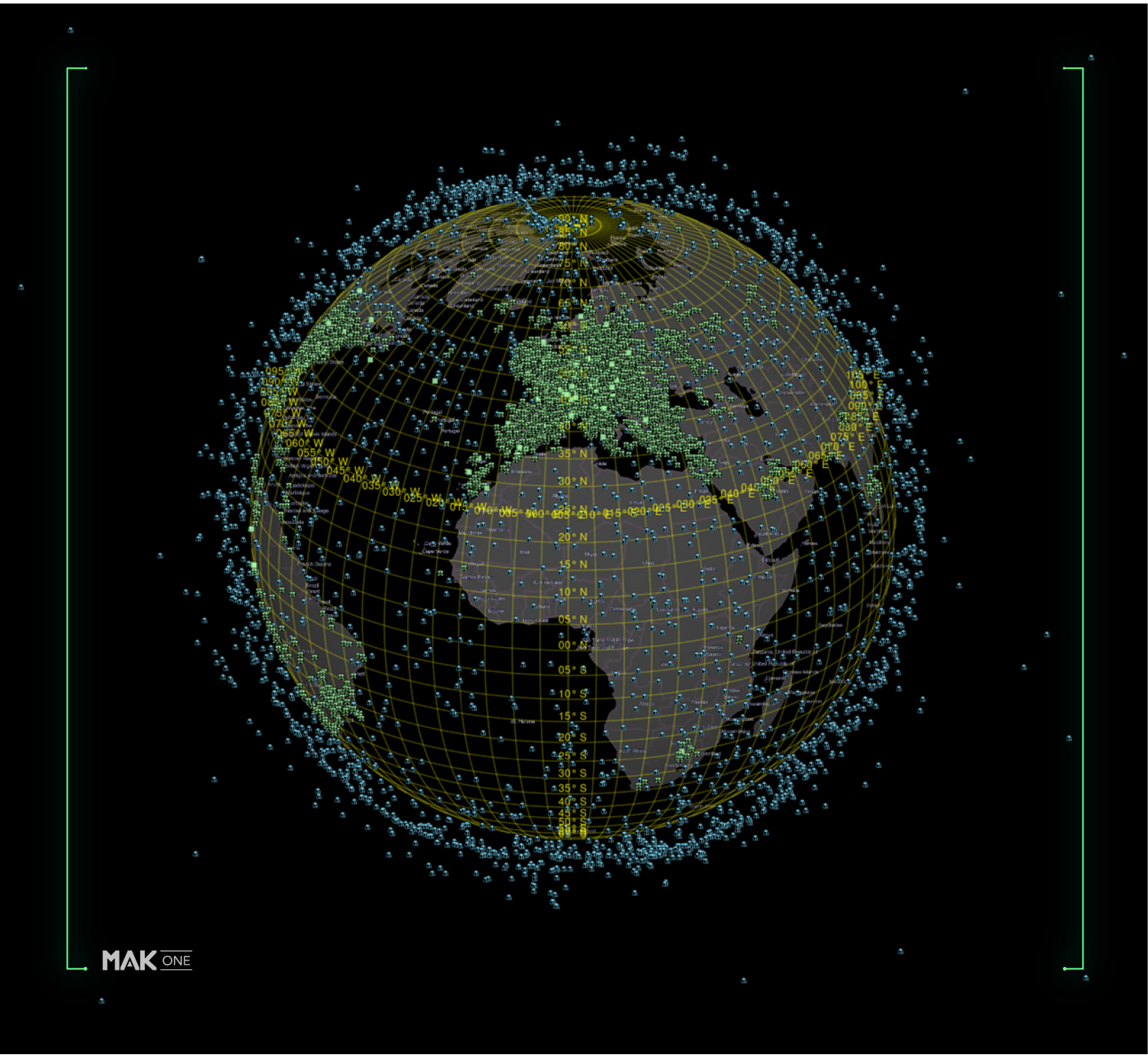
XML, 65
  include files, 67
XXX_SpeedTrees.xml, 121
XYZFeatures, 71
XYZImage, 67

Y

Yaw updater, 267

Z

Z-fighting, 41, 76



MAK-ART-1-230418