MAK TECHNOLOGIES

Adding Content to MAK ONE Applications Reference Manual

VR-Vantage 3.2
VR-Forces 5.2
VR-Engage 2.2

ST Engineering
North America

Copyright © 2025 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.

RFView® is a registered trademark of Information Systems Laboratories (ISL), Inc.

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, visit the MAK ONE Product Support Knowledge Base (at https://www.mak.com/knowledge) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).

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-25.0-1-251009

Contents

PREFACE

How This Manual is Organized 18
MAK Products 19
How to Contact Us 21
Telephone 21
E-mail 21
Internet 21
Post 22
Document Conventions 22
Mouse Button Naming Conventions 22
Third-Party Licenses 23

I. INTRODUCTION

Overview of Adding Content to MAK ONE Applications 26
Screen Captures (Figures) Used in this Document 26

II. TERRAIN

Introduction 28
Terrain Databases Provided with MAK ONE Applications 28
Supported File Formats 30

1. Loading an Existing Terrain

1.1. Loading a Terrain and Saving as a MAK Terrain Format File 34
1.1.1 Load Your Terrain 34
1.1.2 Save Your Terrain 34
1.2. Configuring MetaFlight Terrains 35

1.2.1 Building Efficient MetaFlight Terrains 35
1.2.2 Processing MetaFlight Files for Use in MAK ONE Applications 36
1.2.3 Splitting Datasets into Individual MetaFlight Files 36
1.2.4 Convert Source Data into MEDF Format 36
1.2.5 Converting Virtual Texture Datasets to Tiled Textures 37
1.2.6 Create an MTF File for the MetaFlight Terrain 41
1.2.7 Disabling Bad Vertex Colors or Alpha Values 41
1.3. Configuring CDB Terrains 42
1.3.1 Using CDB Databases in MAK ONE Applications 42
1.3.2 Loading Your CDB Database 42
1.3.3 Configuring CDB earth Files 43
1.3.4 CDB Configuration Files 59
1.3.5 Disabling Intersections in Model Layers 61
1.3.6 Caching CDB Data (cdbTool) 61
1.3.7 Augmenting CDB Terrains 65

2. Composing a Terrain from Source Data

2.1. Earth Files 71
2.1.1 Syntax for earth Files 71
2.1.2 Layers and Layer Types 73
2.1.3 Including External Files in earth Files 74
2.2. Image Layers 75
2.3. Elevation Layers 77
2.3.1 Referencing an Earth Gravitational Model (EGM) 78
2.3.2 Indexing Elevation Layers 79
2.3.3 Loading Multiple DTED Files 81
2.4. Loading Features (Points, Lines, and Areas) 81
2.4.1 Feature Sources 81
2.4.2 Feature Referencing 89
2.4.3 Using Features to Support VR-Forces Simulations 90
2.4.4 Configuring Altitude for Features 90
2.5. Feature Patches 90
2.5.1 Setting up a Patch Layer 91
2.5.2 Creating Patches 92
2.5.3 Editing Feature Patches 93
2.5.4 Using Patches 93
2.6. Rendering Features 93
2.6.1 Configuring Point Features 95
2.6.2 Configuring Linear Features 96
2.6.3 Configuring Areal Features 97

2.6.4 Rendering Features as an Image ... 97
2.6.5 Configuring Point-Model Replacement ... 98
2.6.6 MAK ONE Custom Feature Processing ... 99
2.7. Procedural Buildings ... 101
2.7.1 Procedural Building Elements ... 101
2.7.2 Building Style Properties ... 101
2.7.3 Procedural Building Configuration ... 101
2.7.4 Building Catalog ... 103
2.7.5 Building Library ... 107
2.8. Extruded Buildings ... 108
2.9. Procedural Bridges ... 109
2.9.1 Procedural Bridge Properties ... 109
2.9.2 Configuring a Bridge Layer ... 110
2.10. Specular Reflections on Water ... 112
2.11. Including Azure Maps Imagery Data in earth Files ... 113
2.11.1 Specifying the Subscription Key in the Terrain File ... 113
2.11.2 Specifying the Subscription Key in An Environment Variable ... 114
2.12. Using Cut-in Sites for High Resolution Insets ... 114
2.12.1 Using the Boundary Generation Tool ... 115
2.13. Procedural Vegetation ... 116
2.13.1 The Basics ... 116
2.13.2 More Mechanisms to Control LifeMap Values and Asset Selection ... 121
2.13.3 PE2 for the VR-Forces Simulation Engine ... 125
2.13.4 Configuration Files ... 129
2.13.5 MAK Implementation Guidelines ... 130
2.13.6 Defining Assets ... 133
2.14. Legacy Procedural Vegetation ... 133
2.14.1 Configuring Procedural Trees for VR-Forces ... 135
2.14.2 Configuring Whether Procedural Vegetation Loads in the Simulation Engine ... 137
2.15. Exporting the Location of Procedural Vegetation ... 137
2.15.1 Exporting Terrains that use the Biome-Based Procedural Vegetation Engine (PE2) ... 138
2.15.2 Exporting Terrains that use the Legacy Procedural Vegetation (PE1) ... 138
2.16. Rendering Buoys and Beacons ... 139
2.16.1 Specifying Model Definitions for Buoys ... 140
2.16.2 Modeling Buoys Using Texture Atlases ... 143
2.16.3 Configuring Beacons ... 143
2.16.4 Configuring Navigation Lights ... 144
2.16.5 Configuring Seasonal Buoys and Beacons ... 146
2.16.6 Daymarks ... 146

2.16.7 Topmarks 146
2.16.8 Generating Signal Sequences 147
2.17. Loading 3D Tiles 147
2.18. Options for earth Files 148
2.19. Differential Processing of Elements and Attributes 150
2.19.1 Differential Processing of model Elements 150

3. Paged and Streaming Terrains

3.1. Connecting to Terrain Servers 154
3.1.1 Adding Terrain Server Connections 154
3.1.2 Connecting to Terrain Servers Through a Proxy 155
3.2. Preprocessing Paged Terrains 155
3.3. Manually Paging In Terrain in VR-Forces 156
3.4. 2D View Layers 157
3.4.1 2D View Layers Overview 157
3.4.2 Defining 2D View Layers in earth Files 157
3.5. Procedural Airports 158
3.5.1 Configuration for MAK ONE Applications 158
3.5.2 Data Files 159
3.5.3 Editing Procedural Airport Data 162
3.5.4 Editing Airport Lighting 163
3.5.5 Airport Terminals 166
3.5.6 Other Procedural Airport Data 166
3.5.7 Adding Custom Airfield Lighting 167
3.5.8 Airfield Lighting for Inset Airports and Runways 172
3.5.9 Runway Offset to Avoid Z-Fighting 173

4. Managing Terrains Using the MAK ONE Application GUI

4.1. Introduction to the Terrain Editor 176
4.1.1 Opening the Terrain Editor 176
4.2. Adding Data to a Terrain 177
4.3. Editing Terrain Properties in the Terrain Editor 177
4.3.1 Enabling and Disabling Layers 177
4.3.2 Editing the Properties of a Layer 178
4.3.3 Setting Properties for Terrain Patches 178
4.4. Testing Terrain Components Using the Terrain Editor 180
4.4.1 Adding Elevation Data to a Terrain 180
4.4.2 Adding Images to a Terrain 182
4.4.3 Feature Layers 183
4.4.4 Saving a Terrain 185

4.5. Adding Dynamic Oceans 185
4.5.1 Adding a Dynamic Ocean Layer 185
4.5.2 Adding a Dynamic Ocean Layer Directly 188
4.5.3 Setting the Ocean LOD Elevation 189
4.5.4 Configuring the Ocean Height Map 189
4.6. Specifying Simulation Ocean Settings 192
4.7. Adding Props to a Terrain 193
4.7.1 Extracting Props from a Terrain Patch 193
4.7.2 Adding Props from a Feature Layer 195
4.7.3 Viewing a List of Props 197
4.7.4 Selecting Props 197
4.7.5 Changing a Prop's Type 197
4.7.6 Changing a Prop's Position or Orientation 198
4.7.7 Changing a Prop's Model Definition 198
4.7.8 Exporting Props and SpeedTrees to a Shapefile 198
4.7.9 Importing Props and SpeedTrees from Shapefiles 199

5. Dynamic Terrain in MAK ONE Applications

5.1. Introduction to Dynamic Terrain 202
5.2. Creating Dynamic Terrain Areas and Lines 202
5.3. Dynamic Terrain Switch Types 203
5.4. Craters, Ditches, Berms, and Tank Scrapes 203
5.4.1 Craters 203
5.4.2 Ditches, Berms, and Tank Scrapes 204
5.5. Destructible Buildings in Procedural Terrains 204
5.6. Manually Changing the Terrain 204
5.7. Resetting Dynamic Terrain Changes 205
5.8. Removing Dynamic Terrain Damage 205
5.9. Resetting All Changes in Dynamic Terrain 205

6. Material Classification for SensorFX

6.1. Material Classification on Demand 208
6.1.1 Associating Styles and Features 212
6.2. Streaming Sensorized Imagery 213
6.3. Using SensorFX with Geocentric Terrains 214
6.4. Material System Concept 214
6.4.1 Thermal Phenomena Modeling 215
6.4.2 Common 1D Material Systems 219
6.4.3 Definitions of Assignment Descriptor 220
6.4.4 Boundary Conditions Thermal Equations 221

6.5. Material Classification of CDB Terrains 225
6.5.1 Configuring CDB Raster Material 225
6.5.2 Configuring CDB to Sensorized CDB 3D Models (Geotypical or Geospecific) 225
6.5.3 CDB Base Material Mapping to Sensor Material Classification 226

7. Configuring Feature Data in VR-Forces and VR-Engage

7.1. Introduction 230
7.2. Named Queries 230
7.3. Feature Configuration Attributes 230
7.4. Attribute Transformations 231
7.4.1 Transformations as Aliases 231
7.4.2 Transformations as Derived Attributes 231
7.4.3 Transformations as New Attributes 231
7.4.4 Writing Multiple Transforms to Account for Possible Failures 232
7.5. Source-Specific Configuration 232
7.5.1 Loading Source-Specific Configuration Files 232
7.5.2 Redefining Queries 232
7.6. Extracting Feature Data from GDB to Shapefiles 233
7.7. Converting S-57 Feature Data to Shapefiles 233
7.8. Configuring Aggregate-Level Movement Restrictions 234
7.9. Dynamic Features 236

8. Terrain Performance and Configuration

8.1. Configuring File Caching 238
8.1.1 Specifying the Location to Cache Terrain Server Data 239
8.1.2 Caching earth File Terrain Data 239
8.1.3 Enabling Texture Compression 240
8.1.4 Clearing the File Cache 241
8.2. Configuring Paged and Streaming Terrains 241
8.3. Debugging earth Files 241
8.3.1 Viewing a Terrain in Debug Mode 242
8.4. Using Shader-based Effect Maps 243
8.4.1 Composite Effects Maps 245
8.4.2 Debugging Shaders 246
8.4.3 Reloading Shaders 247
8.5. Indirect Rendering of Terrain 248
8.5.1 Scaling the LOD Distance for Indirect Rendering 248
8.5.2 Setting the LOD Transition Percentage 249
8.5.3 Disabling Indirect Rendering for Terrain 249

8.6. Specifying the Maximum Number of Paged LODs 250
8.7. Indirect Rendering for osgEarth Layers 250
8.8. Displaying Terrain Intersection Lines 250
8.9. Displaying DDS Textures Correctly 251
8.9.1 Flipping DDS Textures Globally 251
8.10. Optimizing GeoTIFF Files 252
8.11. Converting OpenFlight File Texture Data 252
8.11.1 Legacy Approach to OpenFlight File Texture Data 253
8.12. Scene Graph Nodes 254
8.12.1 Drawing Bounding Volumes for Scene Graph Nodes 254
8.12.2 Visualizing Scene Graph Node Bounds and LOD Range and Centers 255
8.13. Masking the Ocean 257
8.14. Debugging the VR-Forces Simulation Terrain 257
8.14.1 Starting the Debug Simulation Terrain Tool 258
8.14.2 Refreshing the Debug Display 258
8.14.3 Viewing Terrain Skin, Normals, and Collision Objects 258
8.14.4 Viewing Simulation Terrain Features 261
8.14.5 Displaying the NavMesh 264
8.14.6 Displaying the Query Area 265
8.14.7 Displaying Elevation Tiles 266
8.14.8 Displaying Recent Intersections 267
8.14.9 Displaying Page-in and Page-out Nodes 268
8.14.10 Enabling Auto Refresh 269
8.14.11 Simulation Terrain Debugging from the Simulation Engine Console 269

9. Terrain Tutorials

9.1. Creating an earth File to Load Terrain Data 272
9.1.1 Add a Map Element 272
9.1.2 Include Options and Elevation in the earth File 272
9.1.3 Add Imagery to the earth File 272
9.1.4 Add Feature Data to the earth File 273
9.1.5 Save the earth File 276
9.1.6 Load the earth File and Save it as an MTF File 276
9.2. Extracting Props from a Terrain 278
9.2.1 Create a New Terrain and Add a Terrain Patch 278
9.2.2 Extract the Props 279
9.2.3 Change the Model Definition for a Prop 281

III. VISUAL MODELS

10. Introduction to Visual Models

10.1. Introduction to Object Modeling 286
10.1.1 Model Sets 287
10.2. Model Formats for 3D Objects, and Effects 288
10.3. OpenFlight and FBX Models 288
10.4. Managing Model and Element Definitions 289
10.4.1 Using the Simulation Object Editor 289
10.5. Visual Model Composability 290
10.6. Indirect Rendering for Entities 294

11. Schemas

11.1. Creating and Editing Schemas 296
11.2. Creating Schemas 297
11.3. Saving Schemas 298
11.3.1 Exporting Selected Schemas 298
11.4. Deleting Schemas 299
11.5. Copying Schemas 299
11.6. Adding a Parameter to a Schema 300
11.7. Editing a Schema Parameter 301
11.8. Deleting a Parameter from a Schema 301

12. Model Definitions

12.1. Creating and Editing Model Definitions 304
12.2. Workflow Issues for Model Definitions 305
12.3. Model Definition Files 306
12.4. Creating a Model Definition 307
12.5. Deleting a Model Definition 307
12.6. Copying a Model Definition 308
12.7. Filtering the List of Model Definitions 309
12.8. Editing a Model Definition 309
12.8.1 Adding Parameters to a Model Definition 309
12.8.2 Editing a Model Definition Parameter 310
12.8.3 Deleting Parameters from a Model Definition 311
12.9. Saving Model Definitions 311
12.9.1 Exporting Selected Model Definitions 311
12.10. Importing Model Definitions 312
12.10.1 Importing Model Definitions from the Command Line 312

13. Element Definitions

13.1. Creating and Editing Element Definitions 314
13.2. How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions Are Organized 314
13.2.1 Leaf File Descriptions 315
13.3. Workflow Issues for Creating and Editing Element Definitions 316
13.4. Creating an Element Definition 317
13.5. Filtering the List of Element Definitions 318
13.6. Editing an Element Definition 318
13.6.1 Adding a Visual Definition 319
13.6.2 Editing a Visual Definition 319
13.6.3 Adding an Attribute to a Visual Definition 321
13.6.4 Deleting an Attribute from a Visual Definition 322
13.6.5 Copying and Pasting Visual Definitions 323
13.7. Jumping to a Model Definition 323
13.8. Changing an Element Definition's File Location 324
13.9. Deleting an Element Definition 324
13.10. Saving Element Definitions 325
13.10.1 Saving Entity, Unit, Spot Report, or Tactical Graphics Element Definitions 326
13.10.2 Saving Interaction and Smoke Element Definitions to .elem Files 326
13.11. Importing Element Definitions 327
13.11.1 Importing a Hierarchy File 328
13.11.2 Importing Leaf Files 328
13.11.3 Importing Element Definitions from the Command Line 328

14. Mapping Models and Effects

14.1. Introduction to Object Type Mapping 332
14.2. Workflow Issues for Object Type Mappings 332
14.3. Adding an Object Type Mapping 333
14.4. Editing an Object Type Mapping 334
14.5. Filtering the Element Definition List 335
14.6. Changing an Object Type Mapping's File Location 336
14.7. Jumping to an Element Definition 336
14.8. Deleting an Object Type Mapping 337
14.9. Saving Object Type Mappings 337
14.9.1 Exporting Selected Object Type Mappings 337
14.10. Importing Object Type Mappings 338
14.11. Starting Without Default Object Type Mappings 338

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

15.1. Configuring 2D Icons 341
15.1.1 Editing Font-Based 2D Icons 341
15.1.2 Using Images for 2D Icons 342
15.1.3 Adding Images for Simulation Object Icons 347
15.2. Adding New Cockpit Display Models 347
15.2.1 Installing a Cockpit DLL 348
15.2.2 Creating a Model Definition for a Cockpit 349
15.3. Adding Object Models as Props 351
15.3.1 Adding a Category to the Props Palette 351
15.4. Clutter Entities 352
15.4.1 Configuring Aircraft Clutter Entities 352
15.4.2 Configuring Ship Clutter Entities 353
15.5. Configuring Light Points 355
15.5.1 Configuring Light Point Type Models 357
15.5.2 Configuring Light Point Group Models 357
15.5.3 Adding a New Light Point Type 358
15.6. Aerodrome and Vehicle Lighting 358
15.6.1 Best Practices for Lightpoints 358
15.6.2 Airfield Lighting 359
15.6.3 Vehicle Lighting Switches 360
15.7. Configuring a Light Source for a Specific Sensor Mode 361
15.8. Configuring Wakes in the Visual Model Editor 361
15.8.1 Configuring Tidal Stream Wakes 364
15.8.2 Displaying Propeller Wash 364
15.9. Configuring Persistent Tracks in Model Definitions 365
15.10. Configuring Floating Lifeforms 366
15.11. Adding Wind-based Controls to Models 367
15.12. Using Texture Atlases to Skin Models 368
15.12.1 Configuring Texture Mapping in the Model Definition 369
15.12.2 Metafiles for Texture Mapping 369
15.13. Adding Dynamically Changeable Markings to Models 371
15.13.1 Adding @dis annotations to the Model 372
15.13.2 The Metafile for Markings 373
15.13.3 The Texture Mapping File 375
15.14. Adding a Depth Water Mask to a Model 375
15.15. Flipping DDS Textures for a Model 376
15.16. Configuring SpeedTree Trees 377
15.16.1 Randomizing the Size of SpeedTrees 379
15.16.2 Displaying SpeedTrees in Planar Reflections 380

15.17. Best Practices for Creating Models 380
15.18. Disabling Indirect Rendering for an OpenFlight Model 381
15.19. Compressing Model Files (medfTool) 381

16. Material Classification for SensorFX

16.1. Material Classification Process Overview 386
16.1.1 Software Tools 386
16.2. Files for Model Material Classification 387
16.2.1 Material System File 387
16.2.2 Active Region File 389
16.2.3 Intensity Modulation File 391
16.3. Material Classification of Models 391
16.4. Debugging Material Classification 392
16.5. Model Definitions for Materially Classified Models 393
16.5.1 The msFilename Parameter 393
16.5.2 The powerPlantActiveRegions Parameter 393
16.5.3 The movingActiveRegions Parameter 393
16.5.4 The weaponFireActiveRegions Parameter 394
16.6. Configuring Particle Systems 394

17. MAK Specification Notes

17.1. Overview of MAK Specification Notes 398
17.1.1 The WRM Specification (DIS Notes) 398
17.1.2 The MAK FBX Specification 399
17.1.3 Modeling Best Practices for Efficient Mapping 401
17.2. Model Coordinate Systems 402
17.2.1 Location of the Origin for Models 403
17.3. The Simulation Attribute Lexicon (SAL) 404
17.3.1 SAL Syntax 405
17.3.2 SAL Keywords 405
17.4. Articulated Parts 406
17.4.1 Embedding Simulation Articulation Data in the Model 406
17.5. Entity Appearances 408
17.5.1 Node Hierarchy Example 409
17.6. Animation Sequences 410
17.7. DIS Keyword Values 411
17.7.1 DI-Guy Articulated Part IDs 435

18. Special Effects (Particle Systems)

18.1. Particle Systems 440

18.2. Adding a New Particle System 440
18.3. Editing Particle Systems 442
18.3.1 Editing the Properties of Keyframes 442
18.4. Creating a Model Definition for a Particle System 443
18.5. Saving and Reverting Particle Systems 443
18.6. Previewing a Particle System 445
18.6.1 Viewing a Particle System With or Without Textures 447
18.6.2 Viewing a Particle System's Wireframes 448
18.6.3 Viewing Debug Visuals 449
18.7. Particle System Properties 451
18.7.1 Preview Parameters for Particle Systems 456
18.8. Particle System Collision Objects 457

IV. DI-GUY

Introduction to DI-Guy 460

19. DI-Guy Content

19.1. Customizing VR-Vantage Content 462
19.1.1 Encrypt Your Final Customized Geometry 462
19.1.2 Compressing Geometry Files 462
19.1.3 Where to Place Customized Files 462
19.1.4 The VR-Vantage Data Directory Structure 463
19.1.5 Understanding DI-Guy Configuration Files 463
19.1.6 Autoload Configuration Files 466
19.2. Customizing Motion 466
19.3. Customizing Texture 466
19.4. Customizing Sound 467
19.5. Creating Custom Hand Items 467
19.5.1 Files Required for a Custom Hand Item 467
19.5.2 Guidelines for Creating Weapon Visuals for Hand Items 469

20. Using the Character Viewer

20.1. Introduction to the DI-Guy Character Viewer 474
20.2. Viewing Characters, Appearances, Gestures, and Actions 475
20.2.1 Filtering the Character Type and Appearance Lists 476
20.2.2 Changing the Viewing Angle for a Character 477
20.2.3 Demonstrating a Character's Aim and Locomotion Capability 477
20.2.4 Viewing Different LOD Levels 478
20.2.5 Changing How a Character Is Rendered 479
20.2.6 Viewing Multiple Appearances at One Time (Army Mode) 481

20.2.7 Advanced Character Editing 484
20.2.8 Reloading Characters that Have Been Edited 486
20.2.9 Controlling Character Movement 486
20.2.10 Viewing Information About Character Movement and State 486
20.3. Improving Performance 487
20.3.1 Displaying the Performance Profiler 488
20.4. Running a Lua Script 488
20.5. Viewing Character Geometry Details 488
20.6. The Appearance Editor 489
20.6.1 Editing Configuration File Text 490
20.6.2 Editing Appearance Effects 491
20.7. Using the PBR Material Tester 491
20.7.1 Reloading Shaders 492
20.8. The Character Viewer Log Window 493
20.8.1 Setting the Notification Level 493
20.9. Broadcasting the Character Viewer over the Network 493
20.9.1 Joining a DIS Exercise 493
20.9.2 Joining an HLA Exercise 495

21. Adding DI-Guy Models

21.1. How MAK ONE Applications Map DI-Guy Models 498
21.2. Adding New DI-Guy Models to VR-Forces 498
21.2.1 Copy DI-Guy Files to the Data Directory 498
21.2.2 Add a New DI-Guy Entity in the Simulation Object Editor 499
21.2.3 DI-Guy Configuration Files 499
21.3. Adding New DI-Guy Models to VR-Vantage 500
21.3.1 Copy DI-Guy Files to the Data Directory 500
21.3.2 Add Model Definitions and Element Definitions 501

22. DI-Guy Tutorials

22.1. Customizing Appearances Using Texture Swaps 508
22.1.1 Select an Appearance to Modify 508
22.1.2 Change the Texture Map 512
22.1.3 Create Your New Texture Swap and Declare Your New Appearance 512

MAK PRODUCTS GLOSSARY

INDEX

Preface

This manual is written for persons who are creating content for MAK ONE 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, when you are creating 3D content, 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 is used in this manual, it refers 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 25.
- "Section II. Terrain" on page 27.
  - "Chapter 1. Loading an Existing Terrain" on page 33 explains how to process and load existing MetaFlight (*.mft*) and Common Database (CDB) terrains into MAK ONE applications.
  - "Chapter 2. Composing a Terrain from Source Data" on page 69 explains how to build terrain databases from your own data.
  - "Chapter 3. Paged and Streaming Terrains" on page 153 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 4. Managing Terrains Using the MAK ONE Application GUI" on page 175 explains how to use the terrain editor to test changes to a terrain.
  - "Chapter 5. Dynamic Terrain in MAK ONE Applications" on page 201 explains how to use the dynamic terrain capability in MAK ONE applications.
  - "Chapter 6. Material Classification for SensorFX" on page 207 describes concepts related to material classification of terrains.
  - "Chapter 7. Configuring Feature Data in VR-Forces and VR-Engage" on page 229 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 8. Terrain Performance and Configuration" on page 237 covers miscellaneous issues that affect terrain performance and function.
  - "Chapter 9. Terrain Tutorials" on page 271 provides step-by-step instructions for creating and saving a terrain database.
- "Section III. Visual Models" on page 283.
  - "Chapter 10. Introduction to Visual Models" on page 285 introduces visual modeling concepts.
  - "Chapter 11. Schemas" on page 295 explains how to create and edit model schemas.
  - "Chapter 12. Model Definitions" on page 303 explains how to create and edit model definitions.
  - "Chapter 13. Element Definitions" on page 313 explains how to create and edit element definitions.
  - "Chapter 14. Mapping Models and Effects" on page 331 explains how to map entity enumerations to the models configured in MAK ONE applications.

  - "Chapter 15. Wakes, Cockpits, 2D Icons, and Other Models" on page 339 explains how to configure some specific visual models, including 2D Icons, cockpits, and SpeedTree trees.
  - "Chapter 16. Material Classification for SensorFX" on page 385 describes the process for material classification of models.
  - "Chapter 17. MAK Specification Notes" on page 397 explains the WRM and FBX specifications for 3D modeling.
  - "Chapter 18. Special Effects (Particle Systems)" on page 439 explains how to edit particle systems.
- "Section IV. DI-Guy" on page 459.
  - "Chapter 19. DI-Guy Content" on page 461 explains how to customize DI-Guy models.
  - "Chapter 20. Using the Character Viewer" on page 473 explains how to use the Character Viewer to examine DI-Guy animations and character types.
  - "Chapter 21. Adding DI-Guy Models" on page 497 explains how to add DI-Guy models to VR-Forces and VR-Vantage.
  - "Chapter 22. DI-Guy Tutorials" on page 507 is a tutorial for editing a texture file.

"MAK Products Glossary" on page 515 defines terms common to DIS, HLA, and MAK products.

MAK Products

The MAK Technologies product line includes:

- 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 NETN FOM and 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™. VR-TheWorld 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.
- RFView® for MAK ONE. Synthetic Aperture Radar (SAR) imaging is an indispensable tool for today's modern militaries. Developed in partnership with ISL, RFView for MAK ONE acts as a SAR server and delivers simulated SAR images that are both physically accurate and fully correlated with out-the-window views and EO/IR images of MAK ONE synthetic environments. It includes a toolkit that enables you to build a client application to request SAR images from the server.
- 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:	https://www.mak.com/mak-one/support/portal
Technical support:	https://www.mak.com/mak-one/support/portal
Product version and platform information:	https://www.mak.com/mak-one-specs
For the free, unlicensed MAK RTI:	https://www.mak.com/mak-one/support/help/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 these typographic conventions:

Sans Serif Bold	Indicates a key on the keyboard or menu options. Mentions of letter keys do not require you to press the **Shift** key. If Shift is necessary, that will be included, for example, **Shift+Q**. Also indicates parameters.
`Monospaced`	Indicates values you enter such as commands or arguments.
`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, `(--help | -h)`.
/	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 VR-Vantage home directory. For example, the directory *./vrlink#.#/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. For an up-to-date list of supported third-party and open source libraries and the associated license information, please visit the MAK ONE Product Support Knowledge Base (at https://www.mak.com/knowledge) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).

I. Introduction

Overview of Adding Content to MAK ONE Applications

Screen Captures (Figures) Used in this Document

Overview of Adding Content to MAK ONE Applications

The MAK ONE applications for visualization and simulation—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-TheWorld 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 for creating 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 (Figures) Used in this Document

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 use VR-Forces, the dialog boxes you see may have additional options and pages. Refer to the VR-Forces help or the *VR-Forces User Guide* to see the information particular to that product.

II. Terrain

Introduction

Terrain Databases Provided with MAK ONE Applications

Chapter 1. Loading an Existing Terrain

Chapter 2. Composing a Terrain from Source Data

Chapter 3. Paged and Streaming Terrains

Chapter 4. Managing Terrains Using the MAK ONE Application GUI

Chapter 5. Dynamic Terrain in MAK ONE Applications

Chapter 6. Material Classification for SensorFX

Chapter 7. Configuring Feature Data in VR-Forces and VR-Engage

Chapter 8. Terrain Performance and Configuration

Chapter 9. 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.

When you open a terrain in MAK ONE applications, you load a MAK Terrain Format (MTF) file, which defines various terrain properties and is a container for other terrain formats. These formats can include:

- Polygonal database formats, such as OpenFlight, MetaFlight, or TerraPage
- Terrain data models and structures such as the Open Geospatial Consortium (OGC) Common Database (CDB).
- GIS source data formats, both as local files and streamed from a terrain server, wrapped in an earth (*.earth*) file

Depending on the terrain sources you use, the general workflow is:

1. Optionally, create an earth file.
2. Add Terrain Patches, which can be a combination of an earth file and any inset datasets.
3. Save the terrain as an MTF (*.mtf*) file.
4. Open the MTF file for visualization or as the basis for a simulated scenario.

Terrain Databases Provided with MAK ONE Applications

Table 1 describes the terrain databases provided with MAK ONE applications. These terrains were created for you to use in your simulations. You may also use them to help create your own terrains. Some have variants that may be a different coordinate system, subset of the data, or other variation on the terrain. Terrains with "(online)" in the title connect to VR-TheWorld or another online server and therefore 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 not listed here. Please contact your MAK salesperson for download details.

Table 1. Terrain databases shipped with MAK ONE applications

Terrain Name	Description
Ala Moana.mtf	A high-resolution area of Honolulu, Hawaii, USA, set into a raster map of the world.
ArrivalsHall.mtf	An interior model of an international arrivals hall with passport control stations.

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

Terrain Name	Description
Bakersfield Topo Map (online).mtf	VR-TheWorld 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 Brooklyn with Vegetation.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. The buildings also include switches for the doors and windows, allowing them to be opened or closed.
Example-ComposedTerrain.mtf	An example of a terrain created from pieces. Use this terrain and the associated earth file to help learn how to create your own.
Grid.mtf	A basic, flat terrain featuring only a simple grid design. This terrain is not suitable for VR-Forces scenarios.
Ground_DB II.mtf	FBX. A simple, generic ground terrain, including a road, a runway, and some buildings, surrounded by water.
MAK Earth (online).mtf MAK Earth (online) - Simple For High Fliers.mtf	VR-TheWorld terrain with high resolution insets for California, Honolulu, Hawaii, and some other cities. Includes streaming feature data. The simple version is intended for use when the observer will be high in the air. It is a more basic version of the MAK Earth (online) terrain and has improved performance for simulations where high-resolution ground details are not needed.
MAK Earth Aggregate (online).mtf	VR-TheWorld terrain designed for use in aggregate-level scenarios. When in 2D projection (plan view), you can choose the appearance from the 2D View Layer list. The Topo View layer looks like a paper map, whereas the default view resembles satellite imagery.
MAK Earth Air and Space (online).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), with simple lines and labels indicating countries. The VFR and and LL IFR 2D view layers show different information for the U.S.A. and its territories.
MAK Earth Base (online).mtf	VR-TheWorld MAK Earth terrain with high resolution insets for Honolulu, Hawaii, and Boston, Massachusetts. Includes streaming feature data.

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

Terrain Name	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 entity models.
NTC (online).mtf	A high-resolution area of the National Training Center at Fort Irwin in California, U.S.A.
PacificGroveTrail.mtf	A procedurally generated earth merged with high-resolution imagery and elevation. It includes dense vegetation to simulate a fictional forested area in San Diego county.
Sadr City.mtf	An urban geotypical Middle Eastern terrain in Sadr City, Iraq, a suburb of Baghdad. Ideal for urban scenarios.
Test terrains	Multiple terrains used for testing models and different terrain database capabilities. They include a terrain with S-57 chart data, procedural terrains, as well as flat earth and geocentric terrains. Not all test terrains are suitable for VR-Forces scenarios, for example, Test-WorldFlatEarthNoElevation.
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.

Supported File Formats

MAK ONE applications can import elevation, terrain, images, models, and feature data. Some types of files can contain more than one type of information. For example, GeoTIFF files can contain elevation data, imagery, or land cover data. The following sections list the supported formats.

In theory, MAK ONE applications can load any format supported by osgEarth if you use the earth file format to load them. However, MAK ONE applications do not include drivers for all of the formats supported by osgEarth. Furthermore, when using osgEarth, MAK ONE applications support only geocentric terrains. If you are uncertain about support for a particular format, please contact your MAK distributor or use the Support Portal at https://www.mak.com/mak-one/support/portal.

Terrain Data

MAK ONE applications import terrain data directly as terrain patches. Supported formats include:

- MAK encrypted data format (*.medf*)
- OpenFlight (*.flt*)
- MetaFlight (*.mft*)
- TerraPage (*.txp*)
- CDB (using an earth file)
- GIS source data formats (using an earth file) including imagery, elevation, and features

MAK ONE applications provide limited support for data in these formats. If you have problems loading data in these formats, MAK will try to assist you but cannot guarantee that it will be possible to use your data.

- 3DS
- OSG
- FBX

Elevation Data

MAK ONE applications support these types of elevation data:

- DTED (*.dt0*, *.dt1*, *.dt2*, *.dted*)
- GeoTIFF
- DEM
- Arc/Info Binary Data (*.adf*)
- Other formats supported by osgEarth

Supported Imagery Formats

You can load imagery in these formats:

- GeoTIFF (*.tif*, *.tiff*)
- JPEG2000 (*.jpg2*)
- MBTiles (Raster)
- ECW (Windows only)
- Other formats supported by osgEarth using the GDAL driver. For a list of formats supported by GDAL, go to https://gdal.org. If you have problems loading a format that is not in this list, please ask on the Support Portal at https://www.mak.com/mak-one/support/portal or contact your distributor.

Although many formats and properties are supported, for optimal performance, imagery or elevation files should:

- Use the WGS-84 (EPSG:4326) Coordinate Reference System (CRS).
- Be tiled.
- Have overviews.

An ideal format is Cloud Optimized GeoTIFF (COG).

We cannot guarantee that unlisted formats are fully supported. For more information about earth files, see "3.1. Connecting to Terrain Servers" on page 154 and "Chapter 2. Composing a Terrain from Source Data" on page 69.

Supported Feature Data Formats in the GUI

VR-Vantage supports these feature data formats in the GUI:

- Shape files
- S-57
- FlatGeobuf
- OGR vector formats referenced in earth files

1. Loading an Existing Terrain

This chapter explains how to load a terrain file that you have into a MAK ONE application. It also covers how to process a MetaFlight or CDB terrain so that you can load it.

1.1. Loading a Terrain and Saving as a MAK Terrain Format File 34
1.1.1 Load Your Terrain 34
1.1.2 Save Your Terrain 34
1.2. Configuring MetaFlight Terrains 35
1.2.1 Building Efficient MetaFlight Terrains 35
1.2.2 Processing MetaFlight Files for Use in MAK ONE Applications 36
1.2.3 Splitting Datasets into Individual MetaFlight Files 36
1.2.4 Convert Source Data into MEDF Format 36
1.2.5 Converting Virtual Texture Datasets to Tiled Textures 37
1.2.6 Create an MTF File for the MetaFlight Terrain 41
1.2.7 Disabling Bad Vertex Colors or Alpha Values 41
1.3. Configuring CDB Terrains 42
1.3.1 Using CDB Databases in MAK ONE Applications 42
1.3.2 Loading Your CDB Database 42
1.3.3 Configuring CDB earth Files 43
1.3.4 CDB Configuration Files 59
1.3.5 Disabling Intersections in Model Layers 61
1.3.6 Caching CDB Data (cdbTool) 61
1.3.7 Augmenting CDB Terrains 65

1.1. Loading a Terrain and Saving as a MAK Terrain Format File

When you have a terrain that is ready to use in a MAK ONE application, you add it as a terrain patch. You then save it as an MTF (MAK Terrain Format configuration) file, which MAK ONE applications can then load.

1.1.1 Load Your Terrain

You can add MetaFlight (*.mft*), TerraPage (*.txp*), OpenFlight (*.flt*), and osgEarth (*.earth*) terrains as patches.

- To load a Common Database (CDB) terrain, you must first reference it in an earth file and then add that file as a patch. See "1.3.2 Loading Your CDB Database" on page 42.
- To load a MetaFlight file with virtual texture, you must convert virtual texture datasets into tiled textures and then load each of the *.mft* files as a terrain patch. See "1.2.5 Converting Virtual Texture Datasets to Tiled Textures" on page 37.

➤ **To load a MetaFlight, TerraPage, OpenFlight, or osgEarth terrain:**

1. In a MAK ONE application, choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.

   In VR-Vantage and VR-Engage, you can also choose **File > Add Terrain Patch** and then go directly to step 4.

2. Select the Add Data tab.
3. Click Terrain Patch. The Add Terrain Patch dialog box opens, as well as the Add Terrain Patch file browser window.
4. Navigate to your terrain file (*.mft*, *.txp*, *.flt*, or *.earth*) and then select it.
5. Click Open. The file populates the Add Terrain patch dialog box.
6. Click OK. The terrain loads as a patch on the Earth. The content of your terrain file displays in the Terrain Editor Panel.

1.1.2 Save Your Terrain

Once you successfully load the terrain in the MAK ONE application, save it as an MTF file. After that, you can simply load the MTF file into the application.

➤ **To save a terrain as an MTF file:**

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. Configuring MetaFlight Terrains

MAK ONE applications support MetaFlight 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.

When you load a MetaFlight file, the application is actually loading many individual OpenFlight files. When you enable file caching, the application caches the OpenFlight files to disk in a quick-loading, encrypted format, allowing the files to 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 "8.1. Configuring File Caching" on page 238.

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

1.2.1 Building Efficient MetaFlight Terrains

MAK ONE applications were designed to load MetaFlight Schema v.1.3. In general, all MetaFlight files must be preprocessed before you can load them in a MAK ONE application.

These recommendations apply to constructing MetaFlight terrains for use in MAK ONE applications:

- 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 such 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.)

1.2.2 Processing MetaFlight Files for Use in MAK ONE Applications

Before you can load a MetaFlight file in a MAK ONE application, you must process it as described in this chapter. 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.

1.2.3 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 must be moved into its own file. It is your responsibility to understand the syntax of the MetaFlight file and to know whether you need to split the 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).

1.2.4 Convert Source Data into MEDF Format

Many file formats, including OpenFlight files, are often slow to load, therefore you should 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 --search C:\yourTerrain\flt\textures
  --directory C:\yourTerrain\flt\objects --extensions flt
  --image_extensions rgb --image_extensions rgba
  --image_extensions jpg --image_extensions bmp --threads 0
  --optimize 1
```

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

```
medfTool --directory C:\yourTerrain\flt\textures
  --image_extensions rgb --image_extensions rgba
  --image_extensions jpg --image_extensions bmp
  --threads 0 --optimize 1
```

1.2.5 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.

The mftTool

The mftTool is a command-line tool for converting virtual texture datasets into tiled textures to enable MAK ONE applications to load them. It creates the highest resolution textures possible from the virtual texture dataset and binds them to individual tiles.

The syntax for mftTool is:

```
mftTool --file filename [--col index --row index --level index
  --missing string --searchDir path --textureSize size
  --subDir directory --vtFile filename --regenerateAll
  --generateFiles --encryptImages --publishMftOnly --threads threads
  --processEmats --extendLevelZero distance --ignore_rest
  --version --help]
```

Table 2 describes the command-line options.

Table 2. mftTool command-line options

Option	Description
(--col | -c) *index*	The column to convert, using a zero-based index.
(--encryptImages | -e)	The tiled textures are generated in the MEIF file format. If not specified, the file format will be DDS.

Table 2. mftTool command-line options (continued)

Option	Description
(`--extendLevelZero` | `-z`) *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.
(`--file` | `-f`) *filename*	The name of the MetaFlight file that contains the GGDS.
(`--generateFiles` | `-g`)	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 `--file` option with "_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 `--generateFiles` and `--publishMftOnly` arguments are mutually exclusive (you can use only one or the other, not both).
(`--level` | `-l`) *index*	The level to convert, using a zero-based index.
(`--missing` | `-m`) *string*	The missing texture placeholder.
`--processEmats`	Find materially classified (emat) versions of the virtual textures and generate new emat textures.
(`--publishMftOnly` | `-u`)	Publish a new MetaFlight file that can be loaded by MAK ONE applications. The output name will be equal to the MetaFlight file specified with the `--file` option with "_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*. The `--publishMftOnly` and `--generateFiles` arguments are mutually exclusive.
(`--regenerateAll` | `-A`)	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.
(`--row` | `-w`) *index*	The row to convert, using a zero-based index.
(`--searchDir` | `-r`) *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 2. mftTool command-line options (continued)

Option	Description
(`--subDir` | `-s`) *directory*	The path to the substitution texture specified by the VTDS. The arguments `--subDir` and `--vtFile` are mutually exclusive.
(`--textureSize` | `-x`) *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.
(`--threads` | `-p`) *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.
(`--vtFile` | `-t`) *file*	The name of the MetaFlight file that contains the VTDS. The arguments `--vtFile` and `--subDir` are mutually exclusive.
(`--ignore_rest` | `--`)	Ignores all command-line options that are listed after this argument.
(`--version` | `-v`)	Displays version information.
(`--help` | `-h`)	Displays command-line usage information.

If you do not specify either the generate (`--generateFiles`) or the publish only (`--publishMftOnly`) argument, mftTool prints out information regarding the MetaFlight terrain.

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:

```
mftTool --file C:\yourTerrain\yourTerrain.mft
  --subDir C:\yourTerrain\ds200-vt-otw-virtual-texture
  --generateFiles --regenerateAll --encryptImages
```

This 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
  --file C:\yourTerrain\ds100-flight-terrain\
      ds100-flight.mft
  --vtFile C:\yourTerrain\ds200-vt-otw-virtual-texture\
      ds200-vt.mft
  --searchDir C:\yourTerrain\textures
  --searchDir C:\yourTerrain\objects
  --missing C:\yourTerrain\MissingTexture.png
  --textureSize 4096
  --generateFiles
```

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

Converting 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. You can convert these using the medfTool like all other files. The command syntax for the medfTool is described in "15.19. Compressing Model Files (medfTool)" on page 381.

To simply publish a new MetaFlight for the cultural dataset, use the mftTool:

```
mftTool
  --file C:\yourTerrain\flt\ds7_culture_db5_building_area_new-
      hier_cult\ds7_culture_db5_building.mft
  --publishMftOnly
  --threads 0
```

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

```
medfTool
  --search C:\yourTerrain\flt\textures
  --search C:\yourTerrain\flt\objects
  --directory C:\yourTerrain\flt\
      ds7_culture_db5_building_area_new-hier_cult
  --extensions flt
  --image_extensions rgb --image_extensions rgba
  --image_extensions jpg --image_extensions bmp
  --threads 0 --optimize 1
```

> ⚠ Never run the medfTool on a directory that contains GGDS tiles that use Virtual Textures.

Loading a MetaFlight File with Virtual Texture Data

When you have tiled texture files, you can add them as terrain patches and then save the terrain as an MTF file.

Load the Files

Load each of the tiled texture files.

➤ **To load the *.mft* files:**

1. In a MAK ONE application, 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 opens, as well as the Add Terrain Patch file browser window.
4. Navigate to the directory.
5. Select the first file, for example, *ds100-flight.mft*.

> Each time the Add Terrain Patch file browser window opens, it resets to the *./SharedData/19/latest* directory. When you add a patch, copy the file path at the top of the window. For each subsequent file that you add, you can simply paste the path into the box and then press **Enter** to return to the folder with your files.

6. Click Open. The file populates the Add Terrain patch dialog box.
7. Click the Add Terrain Patch button (+) to add the next file, for example, *ds101-flight.mft*.
8. Continue adding *.mft* files until you have added all of the files that comprise your virtual texture dataset.
9. When you finish adding the files, click OK. The MAK ONE application loads the files and they display in the Terrain Editor Panel.

Save the Terrain

Save the terrain as an MTF file so that a MAK ONE application can then load the file.

➤ **To save the terrain as an MTF file:**

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.

1.2.6 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 MAK ONE application.
2. Add each MetaFlight terrain as a terrain patch.
3. Save the terrain as an MFT terrain as described in "1.1.2 Save Your Terrain" on page 34.

1.2.7 Disabling Bad Vertex Colors or Alpha Values

Because most renderers do not support vertex or alpha color, you should ignore it when it is in the model file. In the MFT file, add `<myDiscardVertexColor>1</myDiscardVertexColor>` under `<myTerrainPatchDefinitionRecord>`.

1.3. Configuring CDB Terrains

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

> For information on configuring CDB terrains to support physically accurate sensors for use with SensorFX, see "6.5. Material Classification of CDB Terrains" on page 225.

1.3.1 Using CDB Databases in MAK ONE Applications

MAK ONE applications can display multiple CDB datasets (elevation, imagery, geotypical points), which they load 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. (To learn more, see "1.3.2 Loading Your CDB Database" below.)

Because MAK ONE applications use 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), 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.

> - Out of the box, osgEarth itself does not support CDB. The plug-ins used by MAK ONE applications enable support of CDB.
> - MAK ONE applications include a CDB terrain, *Test-CDB.mtf*, which is an example of loading data in CDB format. For details about the CDB API, please see https://portal.presagis.com/support/solutions/articles/19000014917-cdb-api-.

1.3.2 Loading Your CDB Database

When you have a CDB terrain, you must reference it in an earth file so that you can load it into MAK ONE applications.

➤ **To create an earth file for a CDB terrain:**

1. Copy *Test-CDB-Generic-Template.earth* (found in the *./SharedData/19/latest/TerrainData/TerrainConfiguration* directory) and rename the file, for example *myCDB.earth*.
2. Edit the earth file and change all instances of `<cdb_root>` to point to your CDB directory. For example:

   ```
   <cdb_root>c:\myData\myCDB</cdb_root>
   ```

3. In the earth file, edit the `<elevation>` block and set the `max_data_level` to match the levels in the CDB file. For CDB cells between -50 and 50 degrees latitude, the CDB level (L) corresponds to an osgEarth level = L + 9. For example, CDB L10 would be level 19.

4. In the earth file, delete any blocks for which there is no data in your CDB dataset or those that are not needed for your simulation. For example, if your CDB dataset does not contain airport lights, you can remove that block.
5. After you complete your edits, save the earth file. You can then add it as a terrain patch as described in "1.1. Loading a Terrain and Saving as a MAK Terrain Format File" on page 34.

1.3.3 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 these datasets:

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

Configuring the Elevation Dataset (001)

To load an elevation dataset, use the `elevation` element, as in this example:

```
<elevation name="cdb_elevation"
      driver="cdbraster"
      cdb_dataset="1"
      cdb_component_selector1="1
      cdb_component_selector2="1"
      cdb_cache_file="25"
  <cdb_root>../../../../../../../Presagis/CDB/
        myCDBterrain</cdb_root>
</elevation>
```

It tells osgEarth to use the CDB raster driver.

Table 3 describes the attributes for the `elevation` element.

Table 3. 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.

Table 3. Attributes for elevation and imagery datasets (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, see Table 4. 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 restrains 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 when caching is enabled.
cdb_root	Specifies the top folder of the CDB database.

Table 4 lists the `elevation` sub-dataset codes.

Table 4. elevation sub-dataset codes

Dataset Code (Dxxx)	Component Selector 1 (Sxxx)	Component Selector 2 (Txxx)	Description
1	1	1	Primary Elevation.
1	1	2	Primary Elevation Control (deprecated in CDB 3.2). (Not supported.)
1	1	3	Primary Alternate Elevation. (Not supported.)
1	2-99	1	Subordinate Elevation (deprecated in CDB 3.2). (Not supported.)
1	2-99	2	Subordinate Elevation Control (deprecated in CDB 3.2). (Not supported.)
1	100	1	Subordinate Bathymetry.
1	100	2	Subordinate Alternate Bathymetry. (Not supported.)
1	101	1	Subordinate Tide Elevation. (Not supported.)

Configuring the Elevation Sub-dataset (Bathymetry)

Adding the bathymetry allows you to add dynamic ocean. This 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"
  <cdb_root>../../../../../../../Presagis/CDB/
        myCDBterrain</cdb_root>
  <offset>true</offset>
</elevation>
```

If you experience issues with the bathymetry in your CDB database, try the following alternative. Add the `no_data_value` and set it to 0. This acts as a mask in both the elevation and bathymetry datasets. Also, try placing the bathymetry information first in the earth file because the resolution is typically lower than the elevation.

```
<elevation name="cdb_elevation_bathy"
      driver="cdbraster"
      cdb_dataset="1"
      cdb_component_selector1="100"
      cdb_component_selector2="1"
  <cdb_root>../../../../../../../Presagis/CDB/
        myCDBterrain</cdb_root>
  <no_data_value>0</no_data_value>
</elevation>

<elevation name="cdb_elevation"
      driver="cdbraster"
      cdb_dataset="1"
      cdb_component_selector1="1"
      cdb_component_selector2="1"
  <no_data_value>0</no_data_value>
  <cdb_root>../../../../../../../Presagis/CDB/
        myCDBterrain</cdb_root>
</elevation>
```

Configuring the Imagery Dataset (004)

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

```
<image name="cdb_imagery"
      driver="cdbraster"
      cdb_dataset="4"
      cdb_component_selector1="1"
      cdb_component_selector2="1"
      cdb_cache_file="25"
```

```
    <cdb_root>../../../../../../../Presagis/CDB/
          myCDBterrain</cdb_root>
  </image>
```

Table 3 on page 43 describes the attributes for the `image` element. They are the same as for the elevation attribute, except that `cdb_dataset` must be 4.

Table 5 lists the codes for the imagery sub-dataset.

Table 5. 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)

This 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"
          enable_cache=myEnableCache >
```

```
    <cdb_root>../../../../../../../Presagis/CDB/
          myCDBterrain</cdb_root>
    </image>
  </lightmap>
```

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 this 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" >
  <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"
        cache_enabled="false">
    <cdb_root>../../../../../../../Presagis/CDB
          /myCDBterrain</cdb_root>
    </image>
</LandCover>
```

Table 3 on page 43 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 VR-Vantage/VR-Forces CDB plug-in automatically gathers all the needed information for the datasets below.
- If the CDB is not square, you might have an issue querying the soiltype in VR-Forces. In that case, remove the line: `<max_data_level>17</max_data_level>`

Table 6 on the next page lists the codes for the Raster Material sub-dataset.

Table 6. 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.

How MAK ONE Applications 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, MAK ONE applications 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, MAK ONE applications 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/19/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/19/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 soiltype: asphaltorotherhardsurface.

For information on configuring CDB terrains to support physically accurate sensors for use with SensorFX, see "6.5. Material Classification of CDB Terrains" on page 225.

Configuring the Geospecific (GS) Dataset (100)

Geospecific data replicates the real world data. This 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" >
  <cdb_root>../../../../../../../Presagis/CDB
        /myCDBterrain</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 7 describes the attributes for the geospecific dataset.

Table 7. Attributes for geospecific dataset

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
driver	Specifies the driver to use for this dataset: must be `cdb`.
cdb_dataset	Specifies the CDB dataset to read. For geospecific, always use "100".
cdb_component_ selector1	(Optional) Specifies the first CDB component selector to use. Component selectors are part of the sub-dataset naming convention. For a list of selectors, see Table 8. Default: 1.
cdb_component_ selector2	(Optional) Specifies the second CDB component selector to use. Default: 1.
cacheid	(Optional) Specifies a cache file name and location. Requires that caching be 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), colors the models with the same color as the tiles.
range_factor	(Optional) Multiplier of the tile paging values found in *./appData/cdb/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 (*./appData/importConfig/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 8 lists geospecific sub-dataset codes.

Table 8. 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 Light Point Sub-dataset (100-004)

This example code shows how to configure a geospecific airport light point dataset:

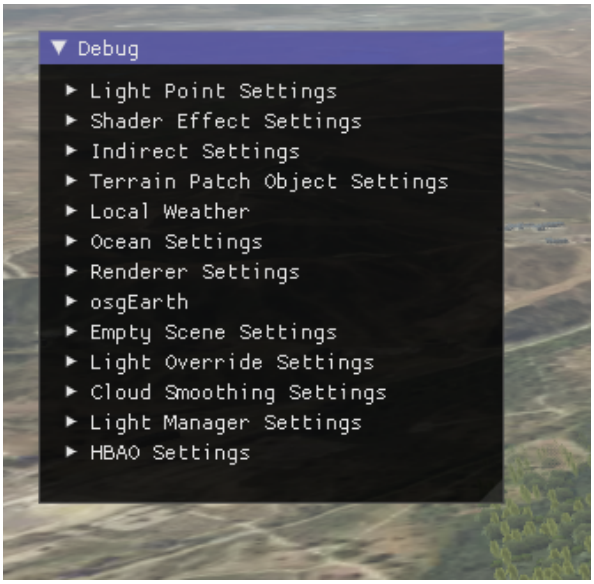
```
<model driver="cdb"
        name="GS Airport Light"
        cdb_dataset="100"
        cdb_component_selector1="4"
        cdb_component_selector2="1" >
  <cdb_root>../../../../../../../Presagis/CDB
        /myCDBterrain</cdb_root>
</model>
```

You can test the proper functioning of the light points by turning them on and off using debugging tools in MAK ONE applications:

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

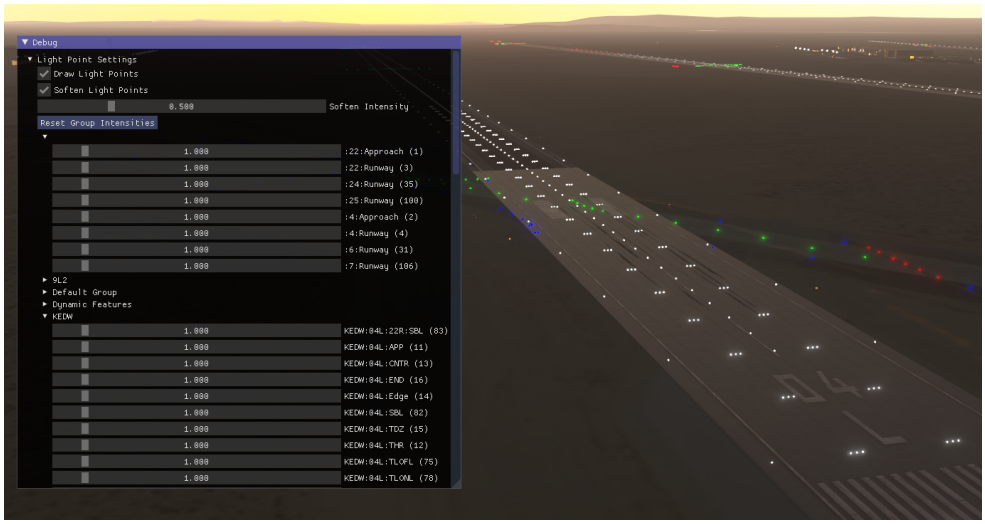
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 42 on page 247).
3. Select the Developer Debug Overlay check box. (You can also simply press **F4**.) The Debug window opens (Figure 1).

Figure 1. Debug window



4. Expand Light Point Settings.
5. Toggle the sliders of the runway you would like to change. For example, in the MAK Earth (online) terrain, if you want to close the runway KEWD: 04L, set the sliders to 0.

Figure 2. Edwards Air Force Base airport with debug window



Configuring the Geospecific Environmental Light Point Sub-dataset (100-005)

This example shows how to configure geospecific environmental light points:

```
<model driver="cdb"
          name="GS Environment Light"
          cdb_dataset="100"
          cdb_component_selector1="5"
          cdb_component_selector2="1"
```

```
    <cdb_root>../../../../../../../Presagis/CDB
            /myCDBterrain</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 these 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 to 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. It 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. Press **F4**. The Developer Debug Overlay opens (Figure 1).
2. Expand Light Point Settings.
3. Select the Debug Type Label check box.
4. Zoom in on the light point that you want to edit. It is labeled with a number (Figure 3).

Figure 3. Light point type labels

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.

This 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" >
  <cdb_root>../../../../../../../Presagis/CDB
        /myCDBterrain</cdb_root>
  <sigsize_file>../userData/terrains/Sample_SignificantSize_
        SwitchDistance.xml</sigsize_file>
  <pagingdistance_file>../userData/terrains
        /Sample_Model_PagingDistance.xml</pagingdistance_file>
  <speedtree_catalog>myterrain_SpeedTrees.xml</speedtree_catalog>
  <ogrfeatures name="Vegetation"/>
  <styles>
    <style type="text/css">
    Vegetation {
      marker-scale:        1;
      marker-placement:    vertex;
```

```
      altitude-clamping:   terrain;
      altitude-resolution: 0.000045;
      altitude-offset:     0;
    }</style>
  </styles>
</model>
```

Table 9 describes the attributes for a geotypical dataset.

Table 9. 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, see Table 10. 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 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 *./appData/cdb/Model_PagingDistance.xml*.
sigsize_file	(Optional) Replace the default significant size file (*./appData/cdb/SignificantSize_SwitchDistance.xml*) with another one.

Table 9. Attributes for geotypical dataset (continued)

Attribute Name	Description
pagingdistance_ file	(Optional) Replace the default paging distance file (*./appData/cdb/Model_PagingDistance.xml*) with another one.
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, see"1.3.4 CDB Configuration Files" on page 59.) The `ogrfeatures` and `styles` elements must be present as in the example for SpeedTrees to work with CDB.
model_scaling	(Optional) Specifies a value that multiplies the CDB - SCALz instances attribute. For example, you might have a CDB GT dataset that was published to have the tree size in feet because of the IG you are using (note that all CDB should be in metric system). You can rectify this for VR-Forces by specifying a scaling value of 0.3.

Table 10 lists the codes for the geotypical sub-dataset.

Table 10. 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).

Configuring the Tiled 2D Model Dataset (310)

The Tiled 2D Model (T2DModel) dataset is typically used for features that are essentially two dimensional in relation to the terrain and have no significant height. They are ideal for features such as roads or runways.

A feature has a 2D modeled representation when it is associated with a modeled representation that has no significant height with respect to the underlying terrain and generally conforms to the terrain profile. The 2D models are a complement an extension to the Primary Elevation and VSTI Imagery datasets. 2D models provide the means to represent 2D surface features that conform to the underlying terrain. These are the modeled representation of geotypical and geospecific features:

- 2D lineal features such as roads, runways and taxiways, stripes.
- 2D areal features such as aprons, surface markings, contaminants, land usage (campgrounds, farms, and so on).

2D models can also be used to model geotypical terrain textures as a mesh of 2D textured polygons overlaying the terrain. This modeling technique replicates approaches used in early IGs, which had limited ability to page in geospecific terrain textures.

To use the T2DModelGeometry dataset from your CDB database, add the following XML to your earth file:

```
<model
  driver="cdb"
  name="T2D"
  cdb_dataset="310"
  cdb_component_selector1="1"
  cdb_component_selector2="1"
  vrfsim:enabled="false" >
    <cdb_root>../../../../../../../Presagis/CDB
      /myCDBterrain</cdb_root>
    <range_factor>1.0</range_factor>
</model>
```

Table 11 describes the attributes for a road dataset. Note that **cdb_dataset**, **cdb_component_selector1**, and **cdb_component_selector2** must use the specified values in order to load the T2DModelGeometry.

Table 11. Attributes for road dataset

Attribute Name	Description
name	Specifies a name that identifies the dataset you are loading in osgEarth.
cdb_dataset	Specifies the CDB dataset to read. For road data, always use "310".
cdb_component_selector1	Specifies the first CDB component selector to use. For road data, always use "1".
cdb_component_selector2	Specifies the second CDB component selector to use. For road data, always use "1".

Table 11. Attributes for road dataset (continued)

Attribute Name	Description
cacheid	Specifies where to save the cache files if caching is enabled. If Accelerated File Loading is enabled, the CDB geotypical models are converted to medf and the textures are converted to meif on the first load. They will be used the next time the CDB database is loaded.
cdb_root	Specifies the top folder of the CDB database.
range_factor	(Optional) Multiplier of the tile paging values found in *Model_PagingDistance.xml*.

The models used in the T2DModelGeometry dataset are not modeled to the elevation of the terrain they are placed upon. Because of this, they must be made to conform at runtime. Currently, the VR-Vantage CDB plug-in implements only Vertex Conformance mode. This is when each vertex of the model is clamped to the elevation of the terrain below it. As a result, models can penetrate the terrain in places. For example, if the model is a long segment of road made from a single quad with only 4 vertices, the terrain may be irregular along the length of the quad and penetrate it in places.

Future releases will aim to support Surface Conformance mode and prevent this from happening.

Configuring the Extended Imagery Dataset (900)

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

This dataset requires 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" >
  <cdb_root>../../../../../../../Presagis/CDB
        /myCDBterrain</cdb_root>
</image>
```

Table 12 describes the attributes for configuring extended imagery.

Table 12. 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 the list of selectors, see Table 13. 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 then you can set that value to 0. This causes 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 13 shows the sub-dataset code for extended imagery.

Table 13. Extended imagery sub-dataset codes

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

1.3.4 CDB Configuration Files

MAK ONE applications use these 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 this code to your earth file:

  ```
  <sigsize_file>./yourSigSizeFile.xml</sigsize_file>
  ```

- *Model_PagingDistance.xml*. Configures the distance for the view point while each LOD tile loads. This can be configured by CDB datasets.

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

  ```
  <pagingdistance_file>./Another_Model_Paging_
      Distance_File.xml</pagingdistance_file>
  ```

- *Light_DefaultValues_3_2.xml*. This file contains all of 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>./Another_Light_Default_
      Values_File.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 these 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 a 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 *myCDB_SpeedTrees.xml* that will be referenced in the Geotypical tree section of your earth file. (For details, see "1.3.3 Configuring CDB earth Files" on page 43.) The file will usually be in the same folder as the earth file.

1.3.5 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.

1.3.6 Caching CDB Data (cdbTool)

When caching is enabled, MAK ONE applications automatically cache terrain data as you load it and store 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 `myTerrain-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 improve 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.)

Caching your files does not make changes to the source files. It creates files in a cache directory that VR-Vantage applications then load. See also "8.1. Configuring File Caching" on page 238.

Depending on the amount of data in the terrain, caching it can take a long time. When there are many models, it can take hours.

- The cached CDB terrain created by the cdbTool in one release of a MAK ONE application may not work in a different release. If the cached terrain from a different release does not work correctly, create a fresh cache of your terrain using the tool in the release you're currently working with. For more information, refer to the MAK ONE Product Support Knowledge Base.
- Updating GPU drivers deletes any cached data on a machine. When you update the GPU drivers on a machine where you have MAK ONE cached data, you must regenerate the cache.

Using the cdbTool Caching Utility

The cdbTool takes the information in the earth file and uses it to generate files, which it saves in a separate directory. These cached files are then used on the next run to improve load times. There are two ways to use the tool:

- Generate files for models and textures.
- Combine and optimize geospecific models.

You specify how you want the tool to process your earth file using the options described in Table 14 on page 64, as shown below. You must have the CDB database and the earth file referencing that database on disk. The tool does not modify the original CDB database; it generates cached files for the MAK ONE application to load.

Note that, if you choose to have the tool save the cached files that it generates in a location other than the default directory (*./appData/cache*), you must point the MAK ONE application to the directory holding the files. Use the `--appCacheDir` command-line argument when running the application, for example:

```
vrvStealth.exe --appCacheDir "C:\myCdbCache"
```

The CDB tool saves console output to a log file, *cdbToolLog.txt*.

Generate Files for Models and Textures

You can simply generate files—MEDFs (MAK Encrypted Data File) for models and MEIFs (MAK Encrypted Image File) for textures—for both geospecific (GS) and geotypical (GT) models. These are the same files that the MAK ONE application would generate when moving around the database in real time and use on the next run to improve load times. For example, if we have a CDB terrain with 10 GS buildings and load it without using the

cdbTool, the MAK ONE application loads all 10 OpenFlight files from disk and generates 10 MEDF models in the cache directory. On subsequent runs, the application intercepts each attempt load of the 10 OpenFlight files and loads the 10 optimized MEDF models instead.

This example simply caches *myCDB.earth* in the default directory using four threads:

```
cdbtool --EarthFile "..\userData\terrains\myCDB.earth"
  --threads 4
```

All file paths are relative, allowing you to run the tool on one machine and then copy the files to other machines that need to use them.

Combine and Optimize Geospecific Models

You can combine multiple GS models that are in close proximity to each other into a single MEDF model based on the specified command-line options. This can greatly improve performance of CDB terrains.

In the example above, we had a CDB terrain with 10 GS buildings. Now, consider a terrain with 100 GS buildings. This multiplies the initial load and generation of files, and the subsequent interception of each file load, by a factor of 10. By using the options to process and optimize the GS models, you reduce the number of MEDF models that the application loads. In a terrain with 100 GS buildings that are close to each other, for example, cdbTool combines them into one MEDF file. When the MAK ONE application intercepts each OpenFlight load request to load from the cache, it loads the one combined file at the first building and then ignores the subsequent 99 calls because it knows they are part of that file.

The following example also uses the *myCDB.earth* file, saving it in a different location than the default. It processes the GS models, combines and optimizes them into groups, and overwrites any previously generated groups and mappings with the ones it has processed. It uses the *C:\MAK\vrvantage#.#\appData* for the CDB XML mappings. The output is saved in *C:\myCdbCache*.

```
cdbTool.exe --processGs --optimiseGsModels --overwriteGsModels
  --EarthFile "C:\MAK\makData\sharedData\19\TerrainData
  \TerrainConfiguration\myCDB.earth" --CachePath "C:\myCdbCache"
  --AppDataPath "C:\MAK\vrvantage#.#\appData"
```

cdbTool Command-Line Options

The syntax for the cdbTool is:

```
./bin64/cdbTool.exe --EarthFile earth_file
  [--minCellSizePerGsGrouping cell_size --maxModelsPerGsGrouping
  max_models --processGt --processGs --lodToProcess lod
  --overwriteGsModels --optimiseGsModels --threads threads
  --AppDataPath file_path --CachePath cache_path --ignore_rest
  --version --help]
```

Table 14 describes the command-line options:

Table 14. cdbTool command-line options

Option	Description
(--AppDataPath | -a) *file_path*	The directory to use for CDB XML mappings. The default is *./appData*. If you are not running the cdbTool from a VR-Vantage application's *./bin64* directory, this is required.
(--CachePath | -c) *cache_path*	The directory where the tool will save the information. Default: *./appData/cache*. Note that this is separate from the location for the cache at runtime. You either need to move the cached information to the runtime cache folder, whether that is the default folder or one you specify.
--disable301TextureOptimization	Use this option for older CDB databases that do not use shared GT textures for GS models. By default, during caching, the tool writes unique 301 textures to a folder in the dataset cache. These textures are shared between GS models. The new GS MEDF model points to the shared cached texture, rather than the individual 301 textures. This option turns off that functionality. Note most newer CDB databases do not have a 301_GSModelTexture folder per tile.
(--EarthFile | -f) *earth_file*	The input earth file name and path containing the CDB datasets information.
(--lodToProcess | -l) *lod*	Specifies one LOD to process. For use in conjunction with `--optimiseGsModels` and can only be used with GS. If you do not specify a value, the tool processes all LODs.
(--maxModelsPerGsGrouping | -m) *max_models*	Maximum number of models that the cache tool will combine into a *.medf* file before the optimizer subdivides the quad tree. This grouped *.medf* is also optimized with baked-in transforms. For use in conjunction with `--optimiseGsModels`. The default is 50.

Table 14. cdbTool command-line options (continued)

Option	Description
(`--minCellSizePerGsGrouping` | `-d`) *`cell_size`*	Minimum size to which the optimizer will subdivide the quad tree, in meters. If a cell is below this value, the optimizer does not subdivide it and the number of models may exceed the maximum number of models specified by `--maxModelsPerGsGrouping` (see above). For use in conjunction with `--optimiseGsModels`. The default minimum cell size is 200 m by 200 m. Use this when there are a large number of models in a small area.
(`--optimiseGsModels` | `-b`)	The tool iterates the CDB database structure, looking for the DBF and SHP files, to combine and optimize GS Models into groups. This reduces the number of nodes loaded from disk for the application. The tool also strips external references.
(`--overwriteGsModels` | `-r`)	Overwrite any previously generated grouped models and mappings. For use in conjunction with `--optimiseGsModels`.
(`--processGs` | `-s`)	Process GS models.
(`--processGt` | `-t`)	Process GT models.
(`--threads` | `-p`) *`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. Too many threads will be slow and run out of memory. This does not work with `--optimiseGsModels`.
(`--ignore_rest` | `--`)	Ignores all command-line options that are listed after this argument.
(`--version` | `-v`)	Displays version information.
(`--help` | `-h`)	Displays command-line usage information.

1.3.7 Augmenting CDB Terrains

CDB terrains contain geometry that can be visualized on the terrain, for example, they can contain buildings, airports, and vegetation models. However, you can use other MAK ONE capabilities to augment and improve the visual quality of the CDB dataset.

Some options include:

- Replacing modeled trees with SpeedTrees.
- Adding procedural vegetation.

- Adding specular effects.

Replacing Modeled Trees with SpeedTrees

CDB terrains often have point features for various tree models. Usually these models are OpenFLT X-trees or billboards, which do not look realistic when the observer is close to or above them. As part of the configuration for that CDB layer, you can optionally replace them with corresponding SpeedTree models. An example of this is in *Test-CDB.earth*, where the layer definition uses a catalog to map the CDB model to corresponding SpeedTree model definitions (modeldefs).

```
<model driver="cdb"
       name="GT Trees"
       cdb_dataset="101"
       cdb_component_selector1="2"
       cdb_component_selector2="1"
       vrfsim:enabled="true">
  <debug>false</debug>
  <debug_modelcolor>false</debug_modelcolor>
  <cdb_root>../../ModelData/Other/CDBTest/CDB</cdb_root>
  <range_factor>1.0</range_factor>
  <speedtree_catalog>speedtree.CDB.xml</speedtree_catalog>
  <!-- using ogrfeatures name="Vegetation" is a "workaround"
       technique to DtStreamingSpeedTreeForestManager load the
       SpeedTree replacement for CDB -->
  <ogrfeatures name="Vegetation"/>
  <styles>
    <style type="text/css">
        Vegetation {
           model-scale:         1;
           model-placement:     vertex;
           altitude-clamping:   terrain;
           altitude-offset:     0;
           altitude-resolution: 5m;
        }
    </style>
  </styles>
</model>
```

Adding Procedural Vegetation

Another approach for adding vegetation is to make use of the Procedural Vegetation capability, which is described in "2.13. Procedural Vegetation" on page 116. Adding this to CDB augments the terrain, adding trees, shrubs, and grass based on the Raster Material (Land Cover) and existing Procedural Vegetation configuration files.

There is an example in *Test-CDB.earth* and the included *Test-CDB.biome.config.xml*. This example uses the CDB Raster Material dataset, loaded as an image layer, and maps the Base Material values to corresponding Procedural Vegetation values. The procedural

engine (PE2) uses the base materials, Procedural Vegetation biomes, and asset tables to generate procedural ground cover and procedural imagery. The procedural imagery is blended with the CDB imagery as the observer moves closer to the terrain.

Adding Specular Reflections on Water

For inland bodies of water, you can add specular reflections to inland water for a more realistic appearance. Specular reflections are described in "2.10. Specular Reflections on Water" on page 112. However, because the current CDB implementation cannot directly use the CDB features, you must have the original source for the water areas, have a way to extract them from the CBD, or recreate them or use an alternate source like the OpenStreetMap data.

An example for the syntax to add the specular effects is available in *example-ComposedTerrain.waterTextures.xml*.

2. Composing a Terrain from Source Data

Composing a terrain allows you to load elevation, imagery, and feature data using an earth file. This chapter explains how to use various types of data files to create a new terrain or make changes to an existing one.

> For basic conceptual information about terrain agility, see "Introduction" on page 28. For information about paged and streaming terrains, see "Chapter 3. Paged and Streaming Terrains" on page 153.

2.1. Earth Files 71
- 2.1.1 Syntax for earth Files 71
- 2.1.2 Layers and Layer Types 73
- 2.1.3 Including External Files in earth Files 74

2.2. Image Layers 75

2.3. Elevation Layers 77
- 2.3.1 Referencing an Earth Gravitational Model (EGM) 78
- 2.3.2 Indexing Elevation Layers 79
- 2.3.3 Loading Multiple DTED Files 81

2.4. Loading Features (Points, Lines, and Areas) 81
- 2.4.1 Feature Sources 81
- 2.4.2 Feature Referencing 89
- 2.4.3 Using Features to Support VR-Forces Simulations 90
- 2.4.4 Configuring Altitude for Features 90

2.5. Feature Patches 90
- 2.5.1 Setting up a Patch Layer 91
- 2.5.2 Creating Patches 92
- 2.5.3 Editing Feature Patches 93
- 2.5.4 Using Patches 93

2.6. Rendering Features 93
- 2.6.1 Configuring Point Features 95
- 2.6.2 Configuring Linear Features 96

2.6.3 Configuring Areal Features ... 97
2.6.4 Rendering Features as an Image ... 97
2.6.5 Configuring Point-Model Replacement ... 98
2.6.6 MAK ONE Custom Feature Processing ... 99
2.7. Procedural Buildings ... 101
2.7.1 Procedural Building Elements ... 101
2.7.2 Building Style Properties ... 101
2.7.3 Procedural Building Configuration ... 101
2.7.4 Building Catalog ... 103
2.7.5 Building Library ... 107
2.8. Extruded Buildings ... 108
2.9. Procedural Bridges ... 109
2.9.1 Procedural Bridge Properties ... 109
2.9.2 Configuring a Bridge Layer ... 110
2.10. Specular Reflections on Water ... 112
2.11. Including Azure Maps Imagery Data in earth Files ... 113
2.11.1 Specifying the Subscription Key in the Terrain File ... 113
2.11.2 Specifying the Subscription Key in An Environment Variable ... 114
2.12. Using Cut-in Sites for High Resolution Insets ... 114
2.12.1 Using the Boundary Generation Tool ... 115
2.13. Procedural Vegetation ... 116
2.13.1 The Basics ... 116
2.13.2 More Mechanisms to Control LifeMap Values and Asset Selection ... 121
2.13.3 PE2 for the VR-Forces Simulation Engine ... 125
2.13.4 Configuration Files ... 129
2.13.5 MAK Implementation Guidelines ... 130
2.13.6 Defining Assets ... 133
2.14. Legacy Procedural Vegetation ... 133
2.14.1 Configuring Procedural Trees for VR-Forces ... 135
2.14.2 Configuring Whether Procedural Vegetation Loads in the Simulation Engine ... 137
2.15. Exporting the Location of Procedural Vegetation ... 137
2.15.1 Exporting Terrains that use the Biome-Based Procedural Vegetation Engine (PE2) ... 138
2.15.2 Exporting Terrains that use the Legacy Procedural Vegetation (PE1) ... 138
2.16. Rendering Buoys and Beacons ... 139
2.16.1 Specifying Model Definitions for Buoys ... 140
2.16.2 Modeling Buoys Using Texture Atlases ... 143
2.16.3 Configuring Beacons ... 143
2.16.4 Configuring Navigation Lights ... 144
2.16.5 Configuring Seasonal Buoys and Beacons ... 146

2.16.6 Daymarks 146
2.16.7 Topmarks 146
2.16.8 Generating Signal Sequences 147
2.17. Loading 3D Tiles 147
2.18. Options for earth Files 148
2.19. Differential Processing of Elements and Attributes 150
2.19.1 Differential Processing of model Elements 150

2.1. Earth Files

An earth (*.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 website.

- MAK ONE applications only support geocentric terrains in earth files.
- Pelican Mapping continues to improve osgEarth. Refer to their documentation as well as this manual to ensure that your terrains adhere to current standards.

2.1.1 Syntax for earth Files

An earth file is comprised of blocks of different information, with each block enclosed by tags. The `<map> </map>` tags are the start and end of the main block.

A typical earth file is constructed using these tags:

```
<!-- A Basic Earth File-->
<map>
  <image>   </image>
  <elevation>   </elevation>
  <featuremodel>   </featuremodel>
  <mask>   </mask>
  <options>   </options>
</map>
```

Each block can have properties. For example, the map can have a name, type, and version:

```
<map name="My Map" type="geocentric" version="2">
```

Table 15 briefly describes the XML elements in the file and their attributes.

Imagery and elevation are the most common types of data added to an earth file. These are defined as the `<image>` or `<elevation>` blocks of the XML file. For specific details on adding those, see:

- "2.2. Image Layers" on page 75
- "2.3. Elevation Layers" on page 77

Table 15. 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 support only geocentric terrains, which can be specified as `geocentric` or `globe`.
	version	(Optional) The earth version.
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 "2.6. Rendering Features" on page 93.
	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 "2.18. Options for earth Files" on page 148.

2.1.2 Layers and Layer Types

Each block defines a set of layers. Layers are generally order-dependent: the first layer renders first and each subsequent layer renders after the previous ones, possibly covering all or part of those layers.

Layer categories include:

- Raster. Spatial data made up of pixels (cells), typically used for imagery, elevation, land cover, or scanned maps.
- Feature. Point, line, or area features can be loaded into the system. This is a staging area where that data could be rendered or used by a simulation.
- Feature Renderer. Layers that render Feature Resources. Sources can be defined within the layer or reference externally-defined Features.
- Miscellaneous layers. A variety of other specialized layers.

The layer name also defines the layer type. The type specifies details about the data, format, source, and how the data will render.

Raster Data Types

Table 16. Raster data sources

Data Source	Description
GDAL	Loads any imagery format supported by the GDAL library, including GeoTIFF
MBTiles	Reads imagery tiles from an MBTiles (MapBox Tiles) database file
TMS	Connects to a TMS (TileMapService) repository
WMS	OGC Web Map Service server.
XYZ	Reads data in standard XYZ format (no metadata).
Composite	Combines multiple image layers into a single map layer.
ContourMap	Renders a colored representation of the elevation data in the map.
Microsoft Azure	($) Connects to Microsoft Azure service. License key required.
Microsoft Bing	Deprecated. ($) Connects to Microsoft Bing service. License key required.
Cesium Ion	($) Connects to a Cesium Ion server instance. License key required.
ESRI ArcGIS Server	($) Connects to an ESRI ArcGIS Server instance.
MapboxGL	Rasterizes a MapboxGL map to an image layer.

2.1.3 Including External Files in earth Files

Each earth file can include external files. This eases management of complex terrains and enables reuse of XML code blocks. The included files must be XML files. There are two ways to include external files:

- `xi:include` element
- earth file template, `{% include filename %}`

This example shows how to use `xi:include`:

```
<map name="Example Terrain" type="geocentric">
  <xi:include href="my_imagery.xml"/>
  <xi:include href="my_elevation.xml"/>
</map>
```

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

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

In addition to being able to reuse XML fragments, the advantage of using `xi:include` is that you can also override properties in the included XML block. For example, you can load an XML file containing an `<elevation>` definition and override its `max_data_level`:

```
  <xi:include max_data_level="10"
href="./elevation.worldwide.online.xml"/>
```

Remember that, when you use an `xi:include`, if the included file is in a different directory, any references within that XML must also be relative to that directory.

Using `xi:include` limits you to a single node, but there may be cases where you want to include multiple elements. You can do this by using earth file templates, an alternative include format. The syntax is:

```
{% include filename %}
```

This syntax is used in many of the earth files in the MAK ONE data package. These lines are from *MAK Earth (online).earth*, where the included XML files are located in the same directory:

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

The `{% include filename %}` is a pre-processor directive, meaning the included XML is inserted inline. When the included XML file has any directory references, for example, other includes, they must be relative to the earth file including the XML file, not the included file. The XML files included in the *Example-ComposedTerrain.earth* file are in the same folder, therefore the directory references are the same.

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 but do not want to delete it, instead of commenting it out, change `include` to `exclude`.

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

2.2. Image Layers

An image layer is any layer that loads and displays tiled imagery on the globe. This could be satellite or aerial imagery, rendered maps, or land cover data.

Image layers are stacked, rendered in order of appearance in the map. Therefore, the first image layer listed in the earth file renders first, followed by subsequent layers, potentially overwriting data in previous layers. In the earth file, an image layer usually has an `Image` suffix, although there are some specialized layers that do not have `Image` in the name.

Image data is generally raster data, although you can rasterize feature data. For local data, the most common type is GDALImage, which you use to load datasets in formats such as Geotiff, JPEG2000, MrSID, and so on. The syntax is:

```
<GDALImage name="Imagery 1">
  <url>../../TerrainData/Terrain/My/Imagery/file1.tif</url>
</GDALImage>
```

You can load multiple datasets as multiple layers:

```
<GDALImage name="Imagery 1">
  <url>../../TerrainData/Terrain/My/Imagery/file1.tif</url>
</GDALImage>
<GDALImage name="World Imagery 2">
  <url>../../TerrainData/Terrain/My/Imagery/file2.tif</url>
</GDALImage>
<GDALImage name="World Imagery 3">
  <url>../../TerrainData/Terrain/My/Imagery/file3.tif</url>
</GDALImage>
```

However, using many layers to load many image layers is inefficient, consumes overhead, and can lead to poor performance. Each layer has its own cache, meaning that the raw data is converted and cached separately.

If you need to load multiple files of the same resolution, it is best to combine the individual files into a Virtual Raster (VRT) and then load the single *.vrt* file:

```
<GDALImage name="World Imagery">
  <url>../../TerrainData/Terrain/My/Imagery/All-Imagery.vrt</url>
</GDALImage>
```

You can also combine imagery of different resolutions together to create a stack of images. Each image has a native resolution. The MAK ONE applications can zoom and view data at that level of detail (LOD) but, as you zoom out, those higher-resolution datasets are resampled and combined for the different LODs. As you zoom further out from the native resolution, the resampling can result in an image that is hard understand. As you zoom further in from the native resolution, the resampling can result in an image blocky. You can combine datasets at different resolutions and specify when they should be rendered as you zoom in and out.

For example, let's say we have three datasets:

- A global low-resolution 100 m dataset, made up of multiple files combined as a *.vrt* file.
- A local area higher-resolution 10 m dataset as a Geotiff file.
- A high-resolution 1 m dataset as a JPEG2000 file.

If we were zoomed to the globe level (LOD 0), the high-resolution 1 m dataset might be combined into a single pixel. However, it still requires processing and overhead to resample and render despite having little value. We can manage this by combining the data into a `CompositeImage` and manage when each layer displays. For example:

```
<CompositeImage name="Combined Imagery (Local)">
  <layers>
    <GDALImage name="World Imagery">
      <url>../../TerrainData/Terrain/My/Imagery/Global-100m.vrt</url>
    </GDALImage>
    <GDALImage name="10m Imagery" min_level="10">
      <url>../../TerrainData/Terrain/My/Imagery/Imagery10m.tif</url>
    </GDALImage>
    <GDALImage name="1m Imagery" min_level="14">
      <url>../../TerrainData/Terrain/My/Imagery/Imagery1m.jp2</url>
    </GDALImage>
  </layers>
</CompositeImage>
```

In this example, we would not display the 10 m imagery until we reach LOD 10 and not display the 1 m imagery until we reached LOD 14. By combining the layers into a `CompositeImage`, it creates a single cache. Each layer will need to be processed until data is cached but once cached, and it only needs to look at the single cache for the data rather than going to separate caches and dynamically compositing the scene each time. Above LOD 14, it will no longer need to look at the 1 m layer and above LOD 10, it will only need to process the single global layer.

Image layers support a wide range of formats, but the most efficient formats for local imagery are Cloud Optimized GeoTIFF (COG) or MBTiles Raster datasets. It also supports a wide range of coordinate systems, however, if you can preprocess the data into WGS84 (EPSG:4326) or Spherical Mercator (EPSG:3857), that data loads faster with less need for dynamic conversion.

`CompositeImage` layers can contain more than just local data. You can combine data streamed from a terrain server, either as separate streamed layers or combined with local data.

Using a `CompositeImage` is a powerful tool to display exactly what you want, when you want it, in the most efficient way. If you have many data files, in many resolutions, you can also purchase your own copy of the MAK VR-TheWorld terrain server, where you have a single location to manage all your terrain data and you can composite the data into a single VR-TheWorld layer and stream on your LAN. This enables you to reduce the overhead to possibly only a single image layer for the MAK ONE client application.

2.3. Elevation Layers

An elevation layer is similar to an image layer in that it is raster data but, rather than each cell representing a color or land cover value, the layer loads and displays tiled heightfield data. In heightfield data, each cell represents an altitude of the terrain surface. Elevation layers are stacked, rendered in order of appearance in the map. In the earth file, an elevation layer usually has an `elevation` suffix.

Elevation layers have several important properties:

- `min_level`. Similar to image layers, the first LOD where the data will be displayed.
- `max_data_level`. The maximum LOD to display. This is useful when you want to limit the data to less than the actual level.
- `offset`. When set to true, adds the height values to those from the previous layer.
- `vdatum`. See "2.3.1 Referencing an Earth Gravitational Model (EGM)" on the next page for complete details.

The `max_data_level` is a useful option when you have high-resolution data but your application does not need that level of detail. For example, you have 1 m (LOD 18), but you are conducting a global analysis simulation. You could use the data as is, however, when you zoom in, the application must load many more tiles than you need. You could use `max_data_level="13"`, to limit the LOD 13 (approximately 90 m). As a result, the application would never load tiles beyond this LOD and there would not be a performance impact loading and rendering unnecessary data.

As with image layers, the prefix specifies the source. You can combine elevation layers in a `CompositeElevation`.

2.3.1 Referencing an Earth Gravitational Model (EGM)

Elevation data is an altitude above or depth below some frame of reference. This frame of reference is called the vertical datum (vdatum). These references are usually one of:

- Geoid. coincides with mean sea level (MSL) which extends under (or over) land areas. The geoid is an equipotential surface at which gravity is normal, closely approximating MSL due to the varying densities that are present on the Earth at different places. There are gravity anomalies with undulations differing from place to place. Geoids are bumpy and irregular.
- Reference Ellipsoid. a mathematical model of the shape of the Earth with the major axis along the equatorial radius. It approximates the geoid but mostly coincides with geodetic network computations to which point coordinates (latitude and longitude) are referred.

The most common reference ellipsoid is the World Geodetic System (WGS84). For DIS and HLA RPR, the specifications require the use of WGS84. This was primarily to determine geodetic coordinates, using a reference system that defines latitude and longitude for points on Earth. However, there was no specific direction for altitude, whether based on a reference ellipsoid or a geoid.

From the DIS specification, for coordinates in DIS PDUs, it is assumed the simulation model of mean sea level is coincident with the WGS84 ellipsoid. Altitude MSL shall use the ellipsoid as their zero-reference surface in this simulation model. This is an approximation of the shape of the real world, but it is an insignificant difference for most simulations.

Part of the DIS rationale was based on the fact that people naturally assume 0 MSL is ocean level. Although true when based on a geoid, this is not the case when based on the WGS84 reference ellipsoid. The other factor was that most early distributed simulations were based on DTED data, which used the Earth Gravitational Model 1996 (EGM96), where 0 was MSL.

In general, MAK ONE applications assume 0 MSL and make use of geoid-based values as-is. However, if needed by your simulation, earth files can support loading elevation data that references a specific earth gravitational model (EGM), which essentially converts values to be relative to the WGS84 ellipsoid. To do so, add the following line to the `elevation` layer:

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

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

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

In this example, a DTED level 0 (*.dt0*) file, which is based on EGM96, will be transformed to values that are now be based on the WGS84 ellipsoid.

Note that several features in VR-Forces assume 0 MSL, for example, the "clutter ships" capability and dynamic oceans. If the altitude of your watercraft entity appears incorrect—which is probably due to using a dataset based on the reference ellipsoid—you

can edit the terrain to tells MAK ONE applications to process geodetic (ellipsoidal or HAE) elevation data and interpret those values as inverted MSL values. This essentially sets the ocean to 0 altitude. To do so, use the inverse of the `egm_model`, for example:

```
<vdatum>-egm96</vdtum>
```

2.3.2 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 that the observer may not see closely, 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 sim engine to know the minimum and maximum altitudes of the elevation in each tile. 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.

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. This example shows an elevation layer configured for indexing:

```
<TMSElevation name="Worldwide 90m Elevation
      with Bathymetry Collection">
  <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>
</TMSElevation>
```

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

You need only a single index. When you use multiple elevation blocks, you add the index to only one of the blocks. If you have a CompositeElevation layer, then you must configure an index for one of the individual elevation layers that makes up the composite, not for the composite.

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 ONE 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.

Table 17. mak terrain index attributes

Attribute	Description
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 allows you to 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 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 because 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. Typically, you should set `max_lod` 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 loading when it should.

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:

```
terrainIndexTool --terrain terrain.mtf [--name elevation_layer]
```

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, the tool processes all layers in the terrain file.

2.3.3 Loading Multiple DTED Files

DTED data is a specific format for elevation data, usually based on EGM96 and frequently provided as multiple individual files. You can load multiple DTED files from an earth file, but some options to improve performance include:

- Create a `CompositeElevation` layer (desirable when there are only a few files).
- Create a Virtual Raster (VRT) file, desirable when there are more than a few files. To create a VRT, you need the gdalbuildvrt program. For more information, go to https://gdal.org/programs/gdalbuildvrt.html.

You could also create overviews (pyramids) for better performance. To create an overview image, you need the gdaladdo program. The DTED format itself does not natively support overviews, so internal overviews (stored within the file) are not possible. However, external overviews (stored in *.ovr* files) are possible and are the typical approach. For more information, go to https://gdal.org/programs/gdaladdo.html.

Note that DTED is not a particularly efficient format. When possible, it is better to use a tool to convert to an EPSG:4326 Cloud Optimized Geotiff. You can then use the `CompositeElevation` or VRT approach for loading multiple files.

2.4. Loading Features (Points, Lines, and Areas)

You can add feature data (point, line, or area geometry) with associated attributes, which can be visualized and used to support the simulation. Features can represent almost anything—point trees, linear roads, building footprints, lake areas, utility lines—anything that could be represented as a point, line, or area.

In an earth file, the first step is to load the feature to make it accessible in memory. Once it loads, you can then use it for a VR-Forces simulation or otherwise render it.

The syntax for loading features is:

```
<features>
  <urls></url>
</features>
```

where the URL is the source of the feature dataset.

2.4.1 Feature Sources

Similar to other layers, feature layers combine the feature source with "Features" to define the layer type. Table 18 lists the available feature sources.

Table 18. Feature sources

Earth File	Description
`MVTFeatures`	Mapbox Vector Tiles specification, used to load local MBTiles vector dataset.
`OGRFeatures`	Uses a GDAL/OGR vector driver to read feature data. This is the most common feature source for reading local data (for example, ESRI Shapefile).
`TFSFeatures`	Reads vector features from a server according to the Tiled Feature Service specification (osgEarth proprietary).
`WFSFeatures`	OGC Web Feature Service specification (limited implementation).
`XYZFeatures`	Generic specification for reading tiled vector data from a server
`FilteredFeatures`	Apply filters to other Feature sources.

OGRFeatures

OGRFeatures is designed to load features using the GDAL (Geospatial Data Abstraction Library) OGR drivers. This is typically used to load local data files in formats including:

- Shapefile
- Geopackage
- Flatgeobuf

There are many uses of OGRFeatures in the MAK Data Terrain Configuration files. This example is from *example-ComposedTerrain.roads.xml*:

```
<OGRFeatures name="RoadsComposedTerrainExample"
             build_spatial_index="true" >
  <url>../../TerrainData/Terrain/ComposedTerrainExample/Features
      /RoadsComposedTerrainExampleL.shp</url>
</OGRFeatures>
```

Tags used in this example include:

- `url`. The feature source, in this case, it is a shapefile (.shp) that contains linear road network data.
- `name`. The OGRFeatures can also have a name. Depending on the scope within the XML file and usage, the name should be unique, allowing the feature to be referenced by other components.
- `build_spatial_index`. The example also uses `build_spatial_index="true"`. When loading the feature data, the OGRFeatures uses the feature spatial index (in the case of the shapefile, the associated *.idx* file) if it exists, or creates one if it does not.

  Unless the data has a small number of features and covers a very small geographic area, you should always specify `build_spatial_index="true"`.
- `profile`. One option not explicitly used in the example is `<profile>`. This is the geospatial profile of the input data. Values for this include:

- global-geodetic (default)
- spherical-mercator

and you can specify them as `<profile>spherical-mercator</profile>`.

For OGRFeatures, the entire referenced dataset loadeds at startup. Therefore, large datasets can take a long time and consume large amounts of memory. To improve performance, limit features to only those that you need and be sure that the attributes are those that are needed and are sized appropriately. For large datasets, you consider converting them to a tiled data source where you can load features as needed.

MVTFeatures

MVTFeatures is designed to load features formatted as using the Mapbox Vector Tiles specification, which is a tiled vector format. This is often used to load local MBTiles feature datasets.

This example is from *biome.density.mask.local.xml*:

```
<MVTFeatures name="osm-ocean">
  <url>../../TerrainData/Terrain/World/Features/
      osm-oceans.mbtiles</url>
  <min_level>11</min_level>
  <max_level>11</max_level>
  <profile>spherical-mercator</profile>
  <format>pbf</format>
</MVTFeatures>
```

This uses the URL to reference the local *.mbtiles* file and sets a name for the feature block. It also sets the profile=spherical-mercator, which is required when using the OSM MBTiles datasets.

Other tags used in this example are:

- `format`. Defines the format of how the tiles are encoded in the .mbtiles file, which in this case is Google Protobufs (PBF).
- `min_level` and `max_level`. MBTiles files can contain data as multiple levels of detail. Sometimes this is useful, but other times that data cannot be used across multiple levels. VRFSIM can only use data as a single level, so you must define it as a single level although the GUI can possibly use multiple levels, depending on how it is being rendered. If the values are not specified, the application uses the minimum and maximum levels of the file.

Because MVTFeatures are tiled, the entire referenced dataset is not loaded at startup. Tiles are loaded only as needed, meaning you can use them to contain very large numbers of geographically-dispersed features.

WFSFeatures

WFSFeatures is designed to load a remote data source using the OGC Web Feature Service (WFS) specification. Note that this is not a fully-compliant WFS implementation: capabilities are limited. It is primarily designed to work with the MAK ONE terrain server, VR-TheWorld, but may also support other WFS servers.

There are many uses of WFSFeatures in the MAK Data Terrain Configuration files. This example is from *example-ComposedTerrain.roads.xml*.

```
<WFSFeatures name="Lake-Bakersfield">
  <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
  <typename>82</typename>
</WFSFeatures>
```

Tags used in this example include:

- `url`. In this case, the url is the actual web address of the WFS endpoint on the VR-TheWorld server.
- `typename`. The layer ID (or name if the server is configured to use layer names).

As with OGRFeatures, the entire referenced dataset is downloaded from the server when you start a MAK ONE application. Therefore, large datasets can take a long time and consume large amounts of memory. To improve performance, limit features to only those that you needed and be sure that the attributes are those that are needed and are sized appropriately.

WFSFeatures will recognize a tiled data source and automatically load it as a TFSFeature instead. As a best practice, always use WFSFeatures for VR-TheWorld feature layers since they can be tiled or not. This ensures that it will work efficiently in either case.

TFSFeatures

TFSFeatures is used to load tiled vector features from a server according to the osgEarth proprietary Tiled Feature Service specification. This can be used for VR-TheWorld feature layers that have been tiled. However, it is more commonly used to access MBTiles datasets that are hosted on VR-TheWorld. For example, this includes all the available OpenStreetMap datasets.

This example is from osm.features.xml:

```
<TFSFeatures name="data:osm-all-features" >
  <url>http://vr-theworld.com/vr-theworld/mbtiles/osm/</url>
  <min_level>14</min_level>
  <max_level>14</max_level>
  <profile>spherical-mercator</profile>
  <fid_attribute>@id</fid_attribute>
  <format>pbf</format>
  <patch>data:osm-all-features-PATCH</patch>
</TFSFeatures>
```

Tags used in this example include:

- `url`. The feature source, in this example, the web address of the OSM MBTiles file.
- `name`. Feature name.
- `min_level` and `max_level`. The minimum and maximum levels of the file to use.
- `profile`. The geospatial profile of the input data, either global-geodetic (default) or spherical-mercator.

- `fid_attribute`. A unique ID used for some internal processing. In this example, it sets the file ID.
- `format`. The format of how the tiles are encoded in the .mbtiles file, which is typically PBF.
- `patch`. A patch is a feature patch used to override all features in a particular tile. In this example, the patch enables local modifications to the original OSM dataset. For more details, see "2.5. Feature Patches" on page 90.

XYZFeatures

XYZFeatures provides a generic specification for reading tiled vector data from a server.

This example is from *osm.features.xml*. It is the definition of the patch, which was referenced in the previous example for the OSM TFSFeatures:

```
<XYZFeatures name="data:osm-all-features-PATCH">
  <url>../../TerrainData/Terrain/World/Features/osm-all-features-
      patch/{z}/{x}/{y}.json</url>
  <min_level>14</min_level>
  <max_level>14</max_level>
  <profile>spherical-mercator</profile>
  <format>json</format>
</XYZFeatures>
```

In this example, the format is GeoJSON (*.json*). XYZFeatures requires a URL template containing `{z}`, `{x}`, and `{y}` placeholders, which represent the zoom level and tile coordinates. Additionally, `{y}` can support a unary negative (`{-y}`) for datasets flipped vertically. The template also supports round-robin characters by way of square brackets (`[]`).

Filtering and FilteredFeatures

Filtering is done to reduce the number of features that MAK ONE applications need to render or use in the simulation. FilteredFeatures is a specialized layer that has built-in capabilities to filter geospatial data based on specific criteria. This is useful when working with large datasets where you only need to display or process a subset of features that meet certain conditions.

FilteredFeatures work on other features, creating a new feature object. The FilteredFeatures must identify the other feature. You can do this either by embedding the feature block or by referencing a previously defined feature. See "2.4.2 Feature Referencing" on page 89 for details about referencing features.

FilteredFeatures provide:

- Custom filtering logic. You can define custom filters to include or exclude features based on attributes, geometry, or other properties.
- Efficient data handling. By filtering features, you reduce memory usage and improve rendering performance.
- Integration with feature layers. Works seamlessly with other feature layers, allowing you to apply filters dynamically.

You can apply a `<filters>` block to any of the features layers. Within a `<filters>` block, you can add blocks for:

- `<script>`. Embed JavaScript to provide logic to filter specific features.
- `<intersect>`. Use another feature to filter specific features that are either inside or outside of the intersected feature.
- `<join>`. Join another feature to combine attributes.

FilteredFeatures adds the following:

- `<attributes>`. Limit the features to only the specified attributes.

You could use JavaScript logic to filter based on attributes, but the advantage of FilteredFeatures is to process filtering in the compiled code. This is significantly faster than using the interpreted JavaScript.

Filters with Scripts

One example showing filtering using a script is in *example-ComposedTerrain.roads.xml*.

```
<FilteredFeatures name="vehicle-roads-data-texture">
  <features>RoadsComposedTerrainExample</features>
  <filters>
    <script language="javascript">
      <![CDATA[
        function filter() {
          // Ignore everything except primary and secondary highways
          switch (feature.properties.highway) {
          case "secondary":
          case "primary":
            return true;
          default:
            return false;
          }
        }
        filter();
      ]]>
    </script>
  </filters>
</FilteredFeatures>
```

In this example, the referenced feature, `RoadsComposedTerrainExample`, contains all road features. The FilteredFeatures is used to reduce the features to only those marked as `"primary"` and `"secondary"`. The example does this using JavaScript to implement and execute the `filter()` function. The script looks at the feature attribute `"highway"`, accessing it as `feature.properties.highway`. If the value matches either `"primary"` or `"secondary"`, the script returns true and that feature is used. Otherwise, the script returns false and the feature is ignored.

As a best practice, you should access a feature attribute in a script by using `feature.properties.{attribute}`, where *{attribute}* is the name of the attribute in lowercase.

There are still many examples of older syntax style, for example, [*attribute*] or `feature.properties["attribute"]`. In some instances, you must use the `[]` notation. Examples are when you are accessing attributes with punctuation (such as `feature.properties["tower:type"]`) or internal attributes (such as `feature.properties["@id"]`).

Filters with Intersect

One example that demonstrates filtering using an intersect is in *osm.roads.model.xml*.

```
<filters>
  ...
  <intersect>
    <OGRFeatures name="AlaMoanaMask"
                 build_spatial_index="true">
      <url>../../TerrainData/Terrain/Hawaii/Features
                 /AlaMoanaInsetMask.shp</url>
    </OGRFeatures>
    <contains>false</contains>
  </intersect>
</filters>
```

This portion of the filter intersects the new OGRFeatures, usually an area feature, with the feature being processed. The `<contains>` tag determines whether the filter is interested in features that are inside (true) or outside (false) the intersected feature. In this example, contains=false, so only features outside the input area features are used, effectively masking (clearing) any features that exist in the masked areas.

Filters with Join

One example that demonstrates filtering using a join is in *buildings.worldwide.osm.online.xml*.

```
<filters>
  ...
  <join>
    <rough>true</rough>
    <OGRFeatures name="World" build_spatial_index="true">
      <url>../../TerrainData/Terrain/World/Features/world.fgb</url>
    </OGRFeatures>
  </join>
</filters>
```

This portion of the filter joins the new OGRFeatures with the feature being processed, in this example, a world area feature. Every feature being processed is combined with the world features, essentially merging their attributes. This results in every feature getting the world attribute, like country name, which can later be used to tell the corresponding country for each feature. The `<rough>` tag is used as an optimization to tell the join filter to use only the first feature that intersects the tilekey bounds for the feature instead of doing an actual intersection test for each one.

Filters with Attributes

One example that demonstrates filtering using an intersect is in *buildings.worldwide.osm.online.xml*.

```
<filters>
  ...
  <attributes>building, aeroway, man_made</attributes>
</filters>
```

This portion of the filter uses the attribute filter to limit the features to only those that have a `highway`, `aeroway`, or `man_made` attributes. This also could have been done using JavaScript but, because this is a built-in function, it is done within the compiled code. That is significantly faster than using the interpreted JavaScript.

Combining Filters

The previous examples focused on individual filters, but you can combine them in a single feature. The full example showing the combined filtering is in *buildings.worldwide.osm.online.xml*. The example creates a filtered feature, reducing the full set of OSM features to only the subset of buildings that will later be rendered.

```
<FilteredFeatures name="data:buildings-OSM">
   <features>data:osm-all-features</features>
   <filters>
      <attributes>building, aeroway, man_made</attributes>
      <script language="javascript">
         {% include ./buildings.worldwide.osm.base.online.filter.js %}
      </script>
      <join>
         <rough>true</rough>
         <OGRFeatures name="World" build_spatial_index="true">
            <url>../datapath/world.fgb</url>
         </OGRFeatures>
      </join>
      <intersect>
         <OGRFeatures name="BuildingsMask" build_spatial_index="true">
            <url>../datapath/MASK_MAK_Earth.shp</url>
            <typename>BuildingsMask</typename>
            <outputformat>json</outputformat>
         </OGRFeatures>
         <contains>false</contains>
      </intersect>
   </filters>

   <mak_vrf_layer>Building</mak_vrf_layer>

</FilteredFeatures>
```

The filters are applied sequentially:

1. The attribute filter reduces the features to only those with `building`, `aeroway`, or `man_made` attributes.
2. The script is run to further reduce the features. In this case, the JavaScript is an included *.js* file. The reason for doing this is to allow you to edit the *.js* file and make use of the language-specific color coding, auto-completion, and so on.
3. The features are joined with the world feature to add the country attributes.
4. The features are intersected with the mask areas and only those outside the mask areas are used.

This example uses feature referencing, `<features>`, (see "2.4.2 Feature Referencing" below) to define the features to process. It also defines a mak_vrf_layer, which tells VRFSIM to use the layer and identify the kind of layer. See "2.4.3 Using Features to Support VR-Forces Simulations" on the next page for more details on how to configure features for use with VRFSIM.

2.4.2 Feature Referencing

Other layers can use feature layers as input. To access the input feature layer, the features can be declared:

- Embedded. Defined within the scope of the layer that uses the feature.
- Referenced. Defined outside the scope of the layer but referenced within the layer.

For an embedded feature, the feature is declared within the XML for the layer.

```
<FilteredFeatures name="MyFilteredFeature">
  <OGRFeatures name="ExternalFeature" build_spatial_index="true" >
    <url>ExampleL.shp</url>
  </OGRFeatures>
  <filters> </filters>
</FilteredFeatures>
```

For a referenced feature, the feature is declared outside the layer XML and then referenced with the new layer XML using a `<features> </features>` block. As an example, a FilteredFeatures can reference an externally declared OGRFeatures:

```
<OGRFeatures name="ExternalFeature" build_spatial_index="true" >
  <url>ExampleL.shp</url>
</OGRFeatures>
<FilteredFeatures name="MyFilteredFeature">
  <features> ExternalFeature </features>
  <filters>  </filters>
</FilteredFeatures>
```

Alternatively, the FilteredFeatures could be defined as:

```
<FilteredFeatures name="MyFilteredFeature"
      features="ExternalFeature">
  <filters> </filters>
</FilteredFeatures>
```

Because the OGRFeatures is defined separately, it could be referenced by multiple other layers. However, because features are referenced by name, it must have a unique name

2.4.3 Using Features to Support VR-Forces Simulations

In order for the VR-Forces sim engine (VRFSIM) to understand the terrain, you need to configure layers to support the simulation. The layers may or may not be rendered. This is frequently done by defining a soiltype, and there are several ways to define VR-Forces soiltype in terrains. One is to use the landcover data, as described in "2.13.3 PE2 for the VR-Forces Simulation Engine" on page 125, and the OpenFlight terrain inset, as described in "8.11.1 Legacy Approach to OpenFlight File Texture Data" on page 253.

You can also set the soiltype using feature data. For example, if your earth file includes road vector data, you can use the `<mak_vrf_layer>` tag to mark the data as a `"Road"`. You must add the `<mak_vrf_layer>` tag to the feature (OGRFeatures, WFSFeatures, FilteredFeatures, and so on). VRFSIM ignores any feature without the `<mak_vrf_layer>` tag.

This code is used in *example-ComposedTerrain.waterTextures.xml*.

```
<OGRFeatures name="Water Composed Terrain Example"
             build_spatial_index="true">
  <url>../../TerrainData/Terrain/ComposedTerrainExample/Features/
             WaterComposedTerrainExampleA.shp</url>
  <mak_vrf_layer>Lake</mak_vrf_layer>
</OGRFeatures>
```

The feature, comprised of area features representing water, is defined as the mak_vrf_layer `Lake`, which means that VRFSIM now recognizes it as an inland water body. Alternatively, the mak_vrf_layer could be `River`, `Ocean`, or `Waterway`, all of which are mapped to the same internal soiltype.

The tags you can use to map your features are found in *./appData/settings/featureconfig.txt*. For additional details on how to use the information in the *featureconfig.txt*, see "Chapter 7. Configuring Feature Data in VR-Forces and VR-Engage" on page 229. This file also contains the mappings between the feature types and soiltypes. In the file, look at the section that is indicated as general feature types (for example, `URBAN: mak_layer="Municipal"`) and the following section of transforms (for example, `Transform URBAN: string mak_soil_type="Urban"`).

2.4.4 Configuring Altitude for 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"/>
```

For more information, see "2.3.1 Referencing an Earth Gravitational Model (EGM)" on page 78.

2.5. Feature Patches

Sometimes, when using a feature from a remote source, such as a VR-TheWorld Feature Layer or the OSM MBTiles data, the feature is not exactly what you need. For example, when using roads and bridges from the OSM dataset, the data producers do not

necessarily take elevation into account when creating vector data, so there are often correlation errors (for example, a bridge feature may start and end on the side of a slope instead of on level ground, based on where the creator placed the vertices). In other cases, the data could just be incorrect.

Feature Patches enable you to locally override features. This allows you to edit vertices, delete features, or change attributes. Note that these are local versions of the data and will not be updated for any changes to the data on the server for any edited tiles.

2.5.1 Setting up a Patch Layer

To set up a feature patch layer, you first define a patch layer that defines the feature tile patches. You should store your feature tile patches in a local data folder to which you have read/write access and must match the configuration of the data source you are patching (the same min_level, profile, and so on). The format must be GeoJSON.

The following example uses details in *osm.features.xml*, located in *./SharedData/19/latest/TerrainData/TerrainConfiguration.* In this example, we patch the primary OSM data source. That Feature Patch is an XYZFeatures block:

```
<XYZFeatures name="data:osm-all-features-PATCH">
  <url> ../../TerrainData/Terrain/World/Features/osm-all-features-
      patch/{z}/{x}/{y}.json </url>
  <min_level>14</min_level>
  <max_level>14</max_level>
  <profile>spherical-mercator</profile>
  <format>json</format>
</XYZFeatures>
```

Then, in our actual OSM feature layer, we specify the patch to use by specifying the <patch> property with a reference to our Feature Patch:

```
<TFSFeatures name="data:osm-all-features" >
  <url>http://vr-theworld.com/vr-theworld/mbtiles/osm/</url>
  <min_level>14</min_level>
  <max_level>14</max_level>
  <profile>spherical-mercator</profile>
  <!-- need to be able to destroy OSM buildings -->
  <fid_attribute>@id</fid_attribute>
  <format>pbf</format>

  <!-- Geojson file that contains local patches to the OSM data -->
  <patch>data:osm-all-features-PATCH</patch>

</TFSFeatures>
```

When a MAK ONE application requests a feature tile, it now checks for a patch before moving on to the server.

> You can only patch a tiled feature layer (for example, XYZFeatures or TFSFeatures). You cannot add the property to a non-tiled layer (for example, OGRFeatures or FilteredFeatures).

2.5.2 Creating Patches

➤ **To create a patch in a MAK ONE application:**

1. Load a terrain that has a Feature with a patch defined. (Note that it initially does not contain patch tiles.) For example, MAK Earth (online).
2. Press **F4** to open the Developer Debug Overlay (Figure 1 on page 52).
3. Expand osgEarth.
4. Select Feature Patching. The Feature Patch Exporter opens.
5. From the list, select your patched layer. For example, in MAK Earth (online), select data:osm-all-features.

   ⓘ If there is nothing in the list, you may have a configuration error. Check the GUI console for a warning message. To access the GUI console, use the `--showConsole | -c` command-line option when launching the VR-Vantageapplication. For more information, refer to the product's user documentation.

6. Optionally, select Overwrite Existing Patches.

   ⚠ If you select Overwrite Existing Patches, the application deletes any edits previously made to the patched files.

7. Click anywhere on the terrain. The panel lists Tile Keys that overlap the area where you clicked. You may need to zoom in on the area you want to export.

   ⓘ You might find it helpful to display the debug tiles. In the Developer Debug Overlay, under osgEarth, select Map Layers and then, in the Map menu, choose **Add > Debug Tiles (Mercator)**.

8. In the Feature Patch Exporter, click Export Tiles to Patch. This writes the original tile data to your patch layer. Make a note of the Tile Keys, as patch filenames correspond to their values. (If you did not select Overwrite Existing Patches, previously edited data is not changed.)

   The exported JSON files are saved to *./SharedData/19/latest/TerrainData/Terrain/World/Features/osm-all-features-patch*.

ⓘ Due to how ImGui features such as the Developer Debug Overlay are integrated. clicking within ImGui menus and dialog boxes also registers as a click on the terrain below the point you clicked. If you select Overwrite Existing Patches, you might need to click the map again to reset the position.

2.5.3 Editing Feature Patches

The tile patches are standard GeoJSON files. You can edit them with a tool such as QGIS, or you could edit them by hand. Be sure to save the results.

2.5.4 Using Patches

The tile patches are live. Once you make the changes, delete your cache and reload your terrain to see the changes.

2.6. Rendering Features

Once you load feature data, you might want to render it. Once example is to render the data as symbols (points), lines, or polygons (areas) displayed on a map. Another example is to use the features to place 3D models on the terrain.

Model layers are used to load and style features for rendering. There are some specialized models, but the general earth file syntax is to define a `<model>` block that contains style elements and may reference a predefine library of graphical definitions and resources.

```
<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 19 describes the `model` element, its attributes, and some of its sub-elements. (A particular `model` element might not have all of the sub-elements described.)

Table 19. 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	Older format for defining the driver to use for creating models. Currently supported, but may be deprecated in a future release.

Table 19. model element attributes (continued)

Element	Attribute	Description
	vrfsim:enabled	Optional. For use in terrains used by VR-Forces only when the simulation engine needs a different value for an attribute than that used by the GUI. If present, it is ignored by VR-Vantage and the VR-Forces GUI. For details, see "2.19.1 Differential Processing of model Elements" on page 150.
ⓘ 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 are cut into tiles. This attribute specifies how aggressively to tile it.
level	name	The level specifies the level of detail (LOD) beyond which not to display the features.
	max_range	Distance beyond which the feature is not displayed.
selector	class	Specifies the type of feature.
styles		The `styles` element and sub-elements specify how to render the scene. This element is mandatory.
library		A texture library.
	name	Name of the texture library.
	url	The location of the library.
style		A specific style. The name, which precedes the curly brace ({), can be any text you want. For information about the attributes that you can specify in a style, see http://docs.osgearth.org/en/-latest/references/symbology.html.
lighting		Specifies whether to enable lighting. Usually true.

The <model> element can be one of the ones defined in Table 20 or can be a specialized rendering like <buildings> detailed in later sections.

Table 20. Feature renderer data

Earth File	Description
`FeatureImage`	Rasterizes vector data into an image layer.
`FeatureModel`	Renders vector data as geometry.
`TiledFeatureModel`	Similar to a `FeatureModel` layer, but optimized for pre-tiled vector datasets.

FeatureModel is the most common osgEarth syntax. However, the MAK ONE applications have some custom model implementations. These use the old syntax:

```
<model driver="feature_custom">
```

These custom rendering models include:

- Dynamic windsocks
- Dynamic flags
- Dynamic wind turbines
- SpeedTrees
- S-57 models including buoys, beacons, daymarks, and lights
- Breaking waves

2.6.1 Configuring Point Features

This example renders a symbol (icon) at the location of a feature.

```
<OGRFeatures name="point-data" build_spatial_index="true">
  <url>../Terrain/World/Features/points.shp</url>
</OGRFeatures>
<FeatureModel name="icons" features="point-data">
  <feature_indexing enabled="false"/>
  <styles>
    <style type="text/css">
      country {
        icon:          "../../images/warning.png";
        icon-placement: centroid;
        icon-declutter: true;
        icon-align:     center-bottom;
        text-content:   feature.properties.name;
        text-encoding:  utf-8;
        text-align:     center-bottom;
        text-halo:      #1f1f1f;
        text-size:      12;
        text-priority:  parseInt(feature.properties.priority);
      }
    </style>
  </styles>
</FeatureModel>
```

First, we define the feature (OGRFeatures). Next, we define the FeatureModel, referencing the feature. For the style, it renders an icon from a *.png* image. Additionally, it displays text from the feature name attribute. Finally, it sets a rendering priority. As the observer zooms out, automatic decluttering renders those with higher priorities over those with lower priorities when symbols and text overlap.

2.6.2 Configuring Linear Features

You can also render linear features, such as roads, rivers, or railways. The following code, from *roads.alamoana.xml*, configures roads:

```
 <OGRFeatures name="AlaMoana-Roads-Lines" build_spatial_index="true">
   <url>../../TerrainData/Terrain/Hawaii/Features/AlaMoanaRoadL.shp
</url>
   <mak_vrf_layer>Roads</mak_vrf_layer>
 </OGRFeatures>

 <FeatureModel name="AlaMoana Road Lines
(VRFGUI)" overlay="true" enabled="true" visible="false"
vrfsim:enabled="false">
   <feature_indexing enabled="false"/>
   <features>AlaMoana-Roads-Lines</features>
   <feature_name>feature.properties.name</feature_name>
   <styles>
     <!-- Appearance details -->
     <style type="text/css">
       AlaMoana_RoadL {
         stroke: #ff0000;
         stroke-width: 3.0;
         altitude-offset: 10;
         altitude-clamping: terrain-drape;
       }
     </style>
   </styles>
 </FeatureModel>
```

In this example, we define the linear feature, along with a mak_vrf_layer, allowing it to be used with VRFSIM.

Next, we define the FeatureModel, referencing the feature.

In the style, we define the stroke color as a hex RGB (or RGBA) value and a stroke width of 3.0 (pixels). You can use the altitude-offset to offset (raise) the rendered feature above the terrain skin to avoid coplanar artifacts.

Also, in the FeatureModel definition, we have the tags `enabled="true" visible="false" vrfsim:enabled="false"`. In this example, the layer is active (enabled) but not visible by default. Using GUI controls in MAK ONE applications, we can enable the layer for rendering. Additionally, the VRFSIM-specific enabled tag is set to false. Because VRFSIM cares only about the feature, the FeatureModel is not enable for VRFSIM and will be ignored.

2.6.3 Configuring Areal Features

You can also render area features, such as lakes, forests, or other polygonal areas. The following code, from *Bakersfield Topo Map (online).earth*, configures the Lake:

```
 <WFSFeatures name="Lake-Bakersfield">
   <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
   <typename>82</typename>
   <mak_vrf_layer>Lake</mak_vrf_layer>
 </WFSFeatures>
  <FeatureModel name="Bakersfield Lake" overlay="true" node_
caching="true" visible="false"
vrfsim:enabled="true"       enabled="false">
   <feature_indexing enabled="false"/>
   <features>Lake-Bakersfield</features>
   <styles>
     <!-- Appearance details -->
     <style type="text/css">
       Bakersfield-Lake {
       stroke: #4B0082;
       stroke-width: 3.0;
       fill: # 0fafd559;
       altitude-offset: 10;
       altitude-clamping: terrain-drape;
       }
     </style>
   </styles>
 </FeatureModel>
```

In this example, we define the area feature from a VR-TheWorld Feature Layer, along with a mak_vrf_layer, allowing it to be used with VRFSIM.

Next, we define the FeatureModel, referencing the feature.

In the style, we define the stroke color as a hex RGB value and a stroke width of 3.0 (pixels). We also specified the fill color as a hex RGBA value, meaning it will be semi-transparent. You can use the altitude-offset to offset (raise) the rendered feature above the terrain skin to avoid coplanar artifacts.

Also, in the FeatureModel definition, we have the tags `enabled="true"` `visible="false"` `vrfsim:enabled="false"`. In this case, the layer is active (enabled) but not visible by default. Using GUI controls in MAK ONE applications, we can enable the layer for rendering. Additionally, the VRFSIM-specific enabled tag is set to false, Because VRFSIM cares only about the feature, the FeatureModel is not enabled for VRFSIM and will be ignored.

2.6.4 Rendering Features as an Image

Rendering a large number of features can be slow and affect overall framerate. Typically, it is not suitable if you want to see them in 3D. Also, there are often times when the required data must be a raster dataset but you want to use feature data to define the necessary information. You can use a FeatureImage to rasterize a particular feature.

This example is from *MAK Earth Air and Space (online).earth*.

```
<OGRFeatures name="data:countries" build_spatial_index="true">
  <url>../../TerrainData/Terrain/World/Features/ne_10m_admin_0_
countries.shp</url>
</OGRFeatures>
<FeatureImage name="Countries">
  <enabled>true</enabled>
  <feature_indexing enabled="false"/>
  <features>data:countries</features>
    <styles>
      <style type="text/css">
        default {
          stroke: #6a5e69;
          stroke-width: 4.0;
          fill:  #575757;
          altitude-clamping: terrain-drape;
        }
      </style>
    </styles>
</FeatureImage>
```

The example loads a shapefile, specifically area features representing the countries of the world. A FeatureImage is created with the country features as the source.

The style then defines stroke and fill values. However, the end result is a rasterized image. Because the data is cached at the different LODs, once rasterized, the data renders much quicker than the dynamically rendered FeatureModel graphics. You could further optimize the image layer by setting the min_level and max_data_level to limit the number of LODs generated from the features.

2.6.5 Configuring Point-Model Replacement

In some cases, you might want to render 2D symbols. However, you can also use the point feature to place 3D models. This example loads a point feature and uses it to place a specified model.

```
<FeatureModel name="model-demo">
  <feature_indexing enabled="false"/>
  <clustering>true</clustering>
  <lighting>true</lighting>
  <OGRFeatures name="point-data" build_spatial_index="true">
    <url>
      ../datapath/myPoints.shp
    </url>
  </OGRFeatures>
  <layout>
    <tile_size_factor>3</tile_size_factor>
  </layout>
  <styles>
```

```
      <style type="text/css">
        Statue {
          model:              "../datapath/Statue.medf";
          model-scale:        1;
          model-placement:    vertex;
          altitude-clamping:  terrain;
          altitude-resolution: 5m;
        }
      </style>
    </styles>
  </FeatureModel>
```

This example loads a shapefile embedded as part of the model block. The style then uses the specified *.medf* file as the model. When rendered, the statue model is placed at the geographic location (vertex) of the point feature, and clamps to the terrain skin. Note that, by specifying the `altitude-resolution: 5m`, it uses the LOD that corresponds to 5 m rather than a possibly higher LOD for very high-resolution elevation data. This is an important performance optimization to ensure that the application does not need to load more elevation tiles and the results are usually indistinguishable from using the maximum LOD.

2.6.6 MAK ONE Custom Feature Processing

MAK ONE applications have several software modules, which we generically call "processors", that search through streamed-in data looking for things such as SpeedTree trees, objects that need effects textures attached: buoys, beacons, and so on. 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 *example-ComposedTerrain.trees.xml*, SpeedTrees are configured as follows:

```
 <model name="Composed Terrain Trees" driver="feature_custom"
vrfsim:enabled="true">
   <feature_indexing enabled="false"/>
   ...
   <OGRFeatures name="Vegetation">
     <url>
../../TerrainData/Terrain/ComposedTerrainExample/Features/TreesCompose
dTerrainP.shp</url>
   </OGRFeatures>
   <layout>
     <tile_size_factor>3</tile_size_factor>
   </layout>
   <styles>
     <style type="text/css">
       Vegetation {
         model:              feature.properties.type;
         model-scale:        1;
```

```
          model-placement:     vertex;
          altitude-clamping:   terrain;
          altitude-resolution: 5m;
        }
      </style>
    </styles>
    <clustering>true</clustering>
    <lighting>true</lighting>
  </model>
```

The feature name is Vegetation. This means that the speedTreeManager will process it.

Similarly, the featureDataBroker, which is responsible for creating 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">
  <feature_indexing enabled="false"/>
  <OGRFeatures name="s57-data" build_spatial_index="true">
    <url>../../TerrainData/Terrain/Hawaii/Features/US5HA52M
        /BOYLAT.shp</url>
  </OGRFeatures>
  ...
```

and (in *airports.worldwide.features.xml*):

```
<model name="InfrastructureCustom"
       driver="feature_custom"
       enabled="true"
       vrfsim:enabled="false"
  <clustering>false</clustering>
  <lighting>true</lighting>
  <WFSFeatures name="dynamicFeatures" >
    <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
    <typename>226</typename>
  </WFSFeatures>
...
```

The *registrant_feature_list.csv* file is fully commented and provides additional information about this scheme. Refer to that file to learn more.

> The MAK ONE custom processor uses the old `<model driver = "feature_custom >`, whereas osgEarth models use the newer `FeatureModel` notation.

2.7. Procedural 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 is an arduous and expensive task. Within an earth file, you can define layers to procedurally generate textured geometry from areal features to create a representative environment with relatively little work. Publicly available GIS features that represent the building footprint and have attributes such as height, type, and other data can provide the input.

2.7.1 Procedural Building Elements

Procedural Buildings are defined in a `<buildings>` block and are composed of the following elements:

- Areal feature. An areal feature defines the building footprints and various attributes that describe the building.
- Catalog. The building catalog uses tags and size constraints to map a feature to a specific model or geometric properties.
- Styles library. The buildings style library defines all the skins (textures) and models used by the building catalog.
- Styles and selectors. The styles and selectors map the feature to a style block with logic that can use relevant attributes to define different models or tags.

2.7.2 Building Style Properties

There are four main style properties that determine how MAK ONE applications render each feature:

- `building-model`. This property enables a specific model to be used for a feature. It could be a geospecific or a geotypical model based on an attribute of the feature. For example, there could be a feature that has an attribute "name" with a value "Eiffel Tower". In that case, you would use selector logic to map to a specific 3D model. Models are positioned on the centroid of the area feature. If a matching model is found here, processing ends.
- `building-height`. This property sets the height of the geometry, in meters. This could be a value from an attribute, hard-coded, or set by some logic.
- `building-tags`. This property defines a string that contains one or more tags that will be used when matching available building definitions in the catalog. Tags are not predefined but are simply the set of strings used in the catalog.
- `building-library`. The library to be used for the style.

2.7.3 Procedural Building Configuration

The following sample code is from *example-ComposedTerrain.buildings.xml*, located in *./SharedData/19/latest/TerrainData/TerrainConfiguration*. You can also review *buildings.worldwide.osm.online.xml* for a more complex example using OSM data or *buildings.oahu.procedural.online.xml* using a Feature Layer from VR-TheWorld, both used in the MAK Earth (online) terrain.

In the example, you can see that:

- It uses an `OGRFeatures`, which is a shapefile that contains the areal features and attributes.
- It sets the `<mak_vrf_layer>` to Building, allowing that vrfsim can categorize the layer.
- It defines the `<building_catalog>`. The catalog used, *osgEarthCatalogs/default.buildings-catalog.xml*, is one of the MAK catalogs used by most of the procedural building examples. It also defines a catalog specific to the VR-Force sim engine using the `vrfsim:building_catalog` notation. When VR-Vantage or the VR-Forces GUI starts, it uses the standard definition. However, when the VR-Forces sim engine starts, it uses a different catalog. This catalog uses simulation models that are simplified versions of the visual models.
- The `<settings>` block defines a number of rendering properties to specify how far from the viewer different models and parts will be displayed.
- A `<library>` of skins (textures) and models is defined. There could be multiple libraries defined. This example uses *osgEarthCatalogs/default.buildings-style-library.xml*, a library used by most of the MAK example terrains.
- There are two named styles defined: `commercial` and `residential`. Both use the same library and set the `building-height` from the value of the `height` attribute for each feature. The only difference is that commercial block uses `building-tags` with a value `"building Commercial"` and residential uses `building-tags` with a value `"building Residential"`.
- The selector uses two logic expressions to determine which style to use. Each looks at the value of the feature `type` attribute. If the value is `commercial`, then the commercial style is used. If the value is `residential`, then the residential style is used

```
<buildings name="Composed Terrain Example Buildings" additive_
lods="true" vrfsim:enabled="true">
   <enable_cancelation>false</enable_cancelation>
   <building_catalog>
     osgEarthCatalogs/default.buildings-catalog.xml
   </building_catalog>
   <vrfsim:building_catalog>
     osgEarthCatalogs/default.buildings-catalog-vrfsim.xml
   </vrfsim:building_catalog>
   <OGRFeatures name="buildings" build_spatial_index="true">
      <url>
        ../../TerrainData/Terrain/ComposedTerrainExample/Features/Buil
dingsComposedTerrainExampleA.shp
      </url>
   </OGRFeatures>
   <mak_vrf_layer>Building</mak_vrf_layer>
   <settings>
      <bins>
         <bin tag="clutter"      lod_scale="0.25"/>
         <bin tag="singlefamily" lod_scale="0.5"/>
         <bin tag="parapet"      lod_scale="0.25"/>
```

```
      </bins>
      <range_factor>4</range_factor>
   </settings>
   <styles>
      <library name="buildings">
         <url>
           ./osgEarthCatalogs/default.buildings-style-library.xml
         </url>
      </library>
      <style type="text/css">
         commercial {
         building-height:   feature.properties.height;
         building-library:  "buildings";
         building-model:    "";
         building-tags:     "building Commercial";
         altitude-clamping: terrain;
         }
         residential {
         building-height:   feature.properties.height;
         building-library:  "buildings";
         building-model:    "";
         building-tags:     "building Residential";
         altitude-clamping: terrain;
         }
      </style>
      <selector class="commercial">
         <query>
            <expr>feature.properties.type = 'commercial'</expr>
         </query>
      </selector>
      <selector class="residential">
         <query>
            <expr>feature.properties.type = 'residential'</expr>
         </query>
      </selector>
   </styles>
</buildings>
```

2.7.4 Building Catalog

The building catalog defines all the available geometry that can be mapped to a feature. Note that configuring the catalog is complex, therefore the best practice is to use the existing catalog and make modifications, if needed. In the catalog, three sets of values are used to determine whether a feature maps to a particular definition:

- Tags. Can match any of all.
- Minimum and Maximum Height. The specified height must be between the values.

- Minimum and Maximum Area. The calculated area of the areal feature must be between the values.

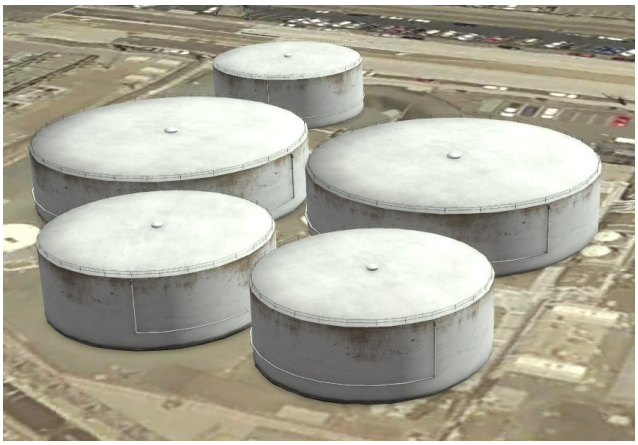
There are two types of Procedural Building definitions: Instanced and Parametric.

Instanced Models

Instanced models are 3D models that are defined in the library. They are typically MEDF, OpenFlight, or FBX models. Models have tags that match those in the instanced model definition. When rendered, the models are placed at the centroid of the areal feature and may or may not be stretched to the footprint or the specified height, depending on the model definition.

For example, there are storage tanks in some MAK ONE terrains.

Figure 4. Storage tank instanced models in San Diego, CA, USA



The XML below is one of the storage tank definitions. To match this model, the feature must:

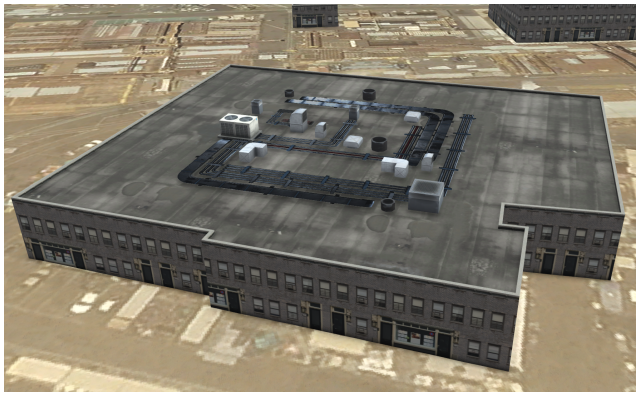
- Have an area less than 350 (m^2), between 0 and 350
- Have a height between 2 and 200 (m)
- Match some portion of the tags "`man_made American StorageTank Narrow`"

```
<Building name="American StorageTank Narrow INSTANCED MODEL "
    tags="man_made American StorageTank Narrow"
    min_height="2"
    max_height="200"
    max_area="350"
    instanced="true">
  <elevations>
    <elevation simplify="true">
    </elevation>
  </elevations>
</Building>
```

Parametric Models

Parametric models are geometry generated from the definition, extruded from the areal feature footprint, so that the geometry matches the footprint to the specified height. The model definition uses `<elevations>` to define different levels of the buildings, enabling you to define different textures for different parts of the building. For example, the first floor can have a texture with doors and windows, subsequent levels could have a texture that has only windows, and the top level could have another texture without windows. You can also define a roof texture for a flat roof, a parapet (the low wall or railing on the edge of a flat roof), and optional roof clutter (HVAC units, pipes, vents, and so on).

Figure 5. Parametric building with roof clutter and a parapet



Parametric buildings can use multiple `<elevations>`. You can think about the height like a stack. That first elevation says "draw that part of the building with the specified skin texture to the specified height". Subsequent elevations build from previous elevations and can either continue to the full height or a specified `height` or `height_percentage`.

Elevations have these properties:

- `height`. The height, in meters, of the specified elevation.
- `skin_name`. The skin (texture) to use from the library.
- `height_percentage`. The percentage of the entire building height to use for this elevation. All elevations should total 100.
- `inset`. How far for the edge of the wall the elevation will be displaced.

Any elevation can also have a `<roof>`. The final elevation must have one, but if you use inset on any elevation, the previous elevation should also have a roof. An elevation has these properties:

- `type`. Currently, only "flat" is a valid option.
- `skin_name`. The skin (texture) to use from the library.
- `model_tags`. A string with a set of tags, which can be used to add roof clutter.

An elevation can also have a `type="parapet"`, which has these properties:

- `height`. The height, in meters, of the parapet.
- `width`. The width, in meters, of the parapet.
- `inset`. How far for the edge of the wall the parapet will be displaced.
- `skin_name`. The skin (texture) to use from the library.

A parapet can also have a `<roof>`.

The following XML example is one definition for a Commercial Parametric building. To match this model, the feature must:

- Have an area between 300 and 1600 (m^2)
- Have a height between 3 and 30 (m)
- Match some portion of the tags `"building Generic American German French Spanish Asian Commercial"`. Typically, the tags would include `"building"` and `"Commercial"` and then one of `"Generic American German French Spanish Asian"`, which could be used to denote a country-specific model.

```
  <Building name="Commercial Grey Brick Small PARAMETRIC"
      tags="building Generic American German French Spanish Asian
Commercial "
      min_area="300"
      max_area="1600"
      min_height="3"
      max_height="30">
    <elevations>
      <elevation height="4" inset="0.0" skin_
name="ResidentialApartmentGrayBrickFirstFloor">
        <elevations>
          <elevation height="-4" skin_
name="ResidentialApartmentGrayBrick">
            <roof type="flat" skin_name="Roof2" model_tags="Roof
Clutter Tiny"/>
            <elevations>
              <elevation
type="parapet" inset="0.0" height="0.5" width="0.4" skin_
name="CorniceStone">
                <roof type="flat" skin_name="CorniceTopStone"/>
              </elevation>
            </elevations>
          </elevation>
        </elevations>
      </elevation>
    </elevations>
  </Building>
```

For this example, any matching feature will have geometry generated that follows its footprint with these characteristics:

- The first floor will have a skin "ResidentialApartmentGrayBrickFirstFloor", which will be the texture with the matching name in the specified building-library.
- If the height is greater than 4 m, then subsequent floors will use the skin "ResidentialApartmentGrayBrick". The height for this elevation is set to "-4". This essentially resets the height to 0, necessary because the first elevation set it to 4. If no height was specified, the overall building height would be the specified height + 4. You could also specify a height value, and the building height would then be the combined height of the two elevations, overriding the specified building height.
- It will have a flat roof with the skin "Roof2" and will include roof clutter that matches "Roof Clutter Tiny".
- It will have a parapet, with an inset="0.0", height="0.5", and width="0.4", in meters, and will use the skin "CorniceTopStone".

2.7.5 Building Library

The building library contains a list of all skins (textures) and models used by the building catalog.

Textures

Skins are textures. The optimal format is to use the converted MEIF format, but any supported image format should work.

Skins can have these properties:

- tags. String containing tags that are used to match the skin.
- name. Name of the skin.
- url. The path and image file to use.
- tiled. Boolean indicating whether the texture is tiled or not.
- image_width. The real-world width of the image, in meters.
- image_height. The real-world height of the image, in meters.
- texture_mode. Defines how the texture is applied. Can be one of: modulate, decal, blend. The default is modulate.
- min_object_height. The minimum acceptable real-world height (in meters) for which the image would make an appropriate texture. Essentially, a hint to select this image only to texture things that are in that height range.
- max_object_height. The maximum acceptable real-world height (in meters) for which the image would make an appropriate texture.

Models

Skins are 3D models. The optimal format is to use the converted MEDF format, but any supported model format should work.

Models can have these properties:

- `tags`. String containing tags that will be used to match the model.
- `name`. Name of the model.
- `url`. The path and model file to use.
- `can_scale_to_fit_xy`. Boolean to specify whether a model will be scaled (in X and Y) to the footprint area. Default is true.
- `can_scale_to_fit_z`. Boolean to specify whether a model will be scaled (in Z) to the input height. Default is true.

2.8. Extruded Buildings

Extruded buildings are an older, simpler approach for rendering geometry from building footprints. As with procedural buildings, geometry can be extruded from areal features with applied textures. The preferred approach is to use procedural buildings as described in "2.7. Procedural Buildings" on page 101, but extruded buildings are still supported.

The following example is from *buildings.washingtondc.online.xml*, located in *./SharedData/19/latest/TerrainData/TerrainConfiguration*.

For this example, the building footprints are coming from a VR-TheWorld Feature Layer and access as WFSFeatures. The footprints are then extruded to height, based on an attribute, `hgt`, from that feature. Each rendered building is defined by a style, `building`. The `building` style references two other styles to compose the geometry from the sides (`building-wall`) and roof (`building-rooftop`). All skins (textures) are defined in the library us_resources, which is an external file: *osgEarthCatalogs/default.buildings-style-library-atlas.osgb.xml*.

```
<FeatureModel name="DC Buildings" vrfsim:enabled="true" node_
caching="true">
  <feature_indexing enabled="false"/>
  <lighting>true</lighting>
  <cluster_culling>false</cluster_culling>

  <WFSFeatures name="buildings" build_spatial_index="true">
    <url>http://vr-theworld.com/vr-theworld/features/wfs</url>
    <typename>21</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:      feature.properties.hgt;
```

```
            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;
         }
         building-rooftop {
            skin-library:     us_resources;
            skin-tags:        rooftop;
            skin-tiled:       true;
            skin-random-seed: 1;
            fill:             #ffffff;
         }
      </style>
   </styles>
</FeatureModel>
```

2.9. Procedural Bridges

Procedural bridges are similar to procedural buildings, where features and attributes are used to generate geometry for that feature. For bridges, linear features are used. Feature attributes can be used to select different visual attributes using styles and JavaScript.

2.9.1 Procedural Bridge Properties

Rendering bridges uses a library of assets, which are textures (skin) that are applied to different parts of the geometry.

The `<Bridges>` layer can use these properties:

- `name`. Name string for the specific `<Bridges>` layer.
- `min_level`. Minimum LOD of the feature, usually used with the OSM MBTiles datasets.
- `max_level`. Minimum LOD of the feature, usually used with the OSM MBTiles datasets.
- `constraint_min_level`. By default, a bridge creates terrain constraints (that is, locked vertices) that conform the terrain to the ends of the bridge. This is the minimum terrain LOD at which these constraints appear. To disable the feature, use 99.

Each bridge style can use these properties:

- `library`. The `<library>` and set of `<skin>` textures available.
- `bridge-deck-width`. Width of the bridge top.
- `bridge-deck-skin`. Texture name from library to apply to the deck.
- `bridge-girder-skin`. Texture name from library to apply to the girder (sides).
- `bridge-railing-skin`. Texture name from library to apply to the railings.
- `bridge-girder-height`. Height of the bridge sides.
- `bridge-railing-height`. Height of the bridge railings.
- `bridge-span-lift`. Amount of uplift applied to each bridge end to avoid coplanar affects.
- `render-backface-culling`. Boolean to specify whether to cull backface polygons or not. Default is true.

2.9.2 Configuring a Bridge Layer

The following examples are from *example-ComposedTerrain.bridges.xml*, located in the *./SharedData/19/latest/TerrainData/TerrainConfiguration*. You can also look at *osm.bridges.xml* for a more complex example using OSM data.

The first step is to define the feature that has linear data describing the bridges. You can do this externally from the `<Bridges>` block and reference it or define it inside the block.

In this example, we use a shapefile that contains roads and bridges linear features:

```
<OGRFeatures name="RoadsAndBridgesExample"
      build_spatial_index="true" >
  <url>../../TerrainData/Terrain/ComposedTerrainExample/Features
        /RoadsComposedTerrainExampleL.shp</url>
</OGRFeatures>
```

Next, we define the `<Bridges>` block, referencing the OGRFeatures. Some key elements of this example include:

- The `<filters>` block reduces the data to only non-pedestrian bridge features.
- The `<styles>` section defines an asset `<library>` with all the textures to be used.
- In the `<style>` section, the properties are defined. There are separate definitions for primary and secondary highways, with a common definition used by both.
- The `<script>` section defines some functions, used by the `<style>`, to determine whether a feature is a primary or secondary highway.
- Also, in the `<filters>` block, the `<resample>` sets the line so that no segment is longer than specified value. The default maximum is 10 m, but these example bridges are too short to benefit from that. The data is in global-geodetic, so the value is in degrees (approximately 1.1 m)

```
<Bridges name="Example Bridges" nvgl="true">

  <features>RoadsAndBridgesExample</features>
```

```
    <constraint_min_level>99</constraint_min_level>

    <filters>
      <script>
        <![CDATA[
          feature.properties.bridge === 'TRUE' &&
          feature.properties.highway !== 'pedestrian';
        ]]>
      </script>

      <resample max_length="0.00001"/>
    </filters>

    <styles>
      <library name="assets">
        <skin name="deck-secondary-road">
          <url>../../ModelData/Environment/Land/Asphalt_png.meif</url>
        </skin>
        <skin name="deck-primary-road">
          <url>../../ModelData/Environment/Land/AsphaltwDashedLines_
png.meif</url>
        </skin>
        <skin name="girder-default">
          <url>../../ModelData/Environment/Land/asphalt_jpg.meif</url>
        </skin>
        <skin name="railing-default">
          <url>../../ModelData/Structures/Barriers/fence2_png.meif</url>
          <image_width>4</image_width>
          <image_height>1</image_height>
        </skin>
      </library>

      <style type="text/css">
        <![CDATA[
          common {
            library: assets;
            bridge-girder-skin: "girder-default";
            bridge-railing-skin: "railing-default";
            bridge-girder-height: 0.75m;
            bridge-railing-height: 0.5m;
            bridge-span-lift: 0.25m;
          }

          SecondaryRoad : common {
            select: is_secondary_road();
            bridge-deck-width: 6m;
            bridge-deck-skin: "deck-secondary-road";
          }
```

```
        PrimaryRoad : common {
          select: is_primary_road();
          bridge-deck-width: 8m;
          bridge-deck-skin: "deck-primary-road";
        }
      ]]>
    </style>

    <script>
      <![CDATA[
        function is_primary_road() {
          return feature.properties.highway === 'primary';
        }
        function is_secondary_road() {
          return feature.properties.highway === 'secondary';
        }
      ]]>
    </script>
  </styles>
</Bridges>
```

2.10. 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 6). This code enables specular reflections:

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

Figure 6. Specular reflections on water

2.11. Including Azure Maps Imagery Data in *earth* Files

Terrains can use imagery and map data from Microsoft's Azure Maps Service.

> The Azure Maps Service replaces the previous Bing Maps service, which is being retired by Microsoft. Bing services will continue until the end of the service date, supporting any existing Bing Maps Enterprise accounts. See https://learn.microsoft.com/bingmaps/coverage for details.

To use maps from the Azure Maps server, you must:

- Have a connection to the internet.
- Have an Azure Maps subscription key. This is listed in your Azure portal, Azure Maps, Authentication.
- Add your subscription key to your terrain file or in an environment variable as described below.

To learn about the available services, refer to the Azure Maps documentation: https://azure.microsoft.com/products/azure-maps.

2.11.1 Specifying the Subscription Key in the Terrain File

The *./SharedData/19/latest/TerrainData/TerrainConfiguration /imagery.worldwide.AzureMaps.online.xml* file is an example of how to configure an Azure Maps layer. You can include this XML in your terrain, or copy the `AzureMapsImage` block to use in your earth file. Either way, add your key between the `<subscription_key>` `</subscription_key>` tags.

The `AzureMapsImage` block syntax is:

```
<AzureMapsImage name="Azure Maps layer">
  <subscription_key>your-key</subscription_key>
  <tileset_id>microsoft.imagery</tileset_id>
  <profile>spherical-mercator</profile>
</AzureMapsImage>
```

AzureMapsImage layers work like any other image layer; you can add it to a CompositeImage, drape it over elevation, or use it within a mapgroup.

Supported options for the `tileset_id` include only the PNG (raster) services listed in https://learn.microsoft.com/rest/api/maps/render/get-map-tile?view=rest-maps-2025-01-01&tabs=HTTP#tilesetid. The PBF format tilesets are not supported.

Typical `tileset_id` values include:

- microsoft.imagery
- microsoft.base.road
- microsoft.base.hybrid.road
- microsoft.base.darkgrey

Other options include:

- map_tile_endpoint - The default value is https://atlas.microsoft.com/map/tile.
- api_version - The default value is 2024-04-01.

2.11.2 Specifying the Subscription Key in An Environment Variable

Create an environment variable OSGEARTH_AZURE_KEY and enter your subscription key as the value. Note that you still must add the Azure Maps imagery layer as shown above; this simply removes the need for entering the subscription key in that layer.

2.12. Using Cut-in Sites for High Resolution Insets

Users of terrain servers, such as VR-TheWorld, often want to add high resolution sites of particular interest while using the relatively 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, and *./SharedData/19/latest/Terrain/Hawaii/Geometry/OpenFlightAlaMoana/AlaMoanaGeocentricProxy.osg*, which is a locally stored high resolution inset.

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

```
<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 21 describes the `mask` element and its sub-elements.

Table 21. 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.

2.12.1 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 to:

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 Table 21 above.)
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:

```
  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. The syntax is:

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

Table 22 describes the options.

Table 22. 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.

2.13. Procedural Vegetation

Procedural vegetation uses land cover data to generate imagery and geotypical ground cover. The next generation rendering capability for MAK ONE applications, Procedural Engine 2 (PE2), enhances the realism of the natural environment. PE2 uses a biome-based procedural vegetation engine and provides expanded capabilities to avoid the limitations of the legacy procedural vegetation approach. The following sections describe the PE2 components and configuration options.

One change from previous versions is that procedural imagery is intended to be used in addition to imagery, not as a replacement. Although it is possible to use only procedural imagery, PE2 does not define imagery that is optimized for low LODs. You should use PE2 only if you set an appropriate min_level for the ProceduralImage layer.

2.13.1 The Basics

PE2 is comprised of three parts:

- Assets. A set of vegetation models and surface textures (images).
- LifeMap. A set of values representing Lush, Rugged, and Dense values for a location.
- Biome. The subset of assets for a specific geographic area.

Assets

There are two parts to the assets: Vegetation Models and Surface Textures.

Vegetation Models

Vegetation Models are a group of models and images that represent specific vegetation. This includes a 3D model (MEDF format, usually converted from FBX), top and side images (for the imposter billboard), associated images for the PBR effects, a simulation model (MEDF), and a thumbnail image (JPG).

For the models and images, MAK has defined naming guidelines to add consistency to the creation process. To learn about the specifics about the model and image creation process and guidelines, see "2.13.6 Defining Assets" on page 133.

When rendered, the PE2 models perform similarly to SpeedTrees, where 3D models render when the observer is close to the models and transition to imposter billboards when the observer is further away.

Surface Textures

Surface Textures are images that represent different surface materials. They are organized as a matrix of images, with rows for Substrate (ground) and Overlay (vegetation) layers. The images have full physics-based rendering (PBR) effects. Images will be blended based on the LifeMap values.

LifeMap

The LifeMap is the primary layer that defines the amount and type of vegetation cover for area. It is a Raster (gridded) dataset that contains Lush, Rugged, Dense values for a location. Values range from 0.0 to 1.0 and are used to choose the specific procedural imagery and vegetation assets.

- Lush
  - How moist (lush) the vegetation is.
  - Value defines an index into overlay textures to use and an index into list of vegetation models.
- Rugged
  - How rough the terrain is.
  - Defines an index into Substrate textures.
- Dense
  - How thick (dense) vegetation is at that location.
  - Defines how much of Overlay textures are blended with Substrate textures and density of ground cover at location.

Biome

The Biome is an area that defines a subset of ground cover assets. Each biome is pageable: assets seamlessly page in and out as needed. Biomes can be an arbitrary size and shape. Multiple biomes can be active.

Procedural Imagery

The procedural image for a location is defined by combining the various texture assets based on the Lush, Rugged, and Dense values for the location. Texture assets are organized as matrix with substrate and overlay rows. The Rugged value will choose the

substrate image, mixing two adjacent textures depending on value. The Lush value will choose the overlay image, mixing two adjacent textures depending on value. The Dense value determines how much of overlay texture is used to mix with substrate.

Texture images also include the associated PBR effects images.

- Displacement defined for the substrate image can show through the overlay vegetation image.
- Other effects are rendered in the resulting scene.

MAK ONE applications include a base configuration for a default matrix and images.

Procedural Ground Cover

The base configuration includes a catalog of assets, which are a grouping of 3D models and imposter (top and side) images. The assets are classified as trees, bushes and undergrowth, which are essentially large, medium and small groupings. The Biome determines the subset of assets for a specific location.

The lush value specifies which assets are chosen. The Lush value determines matching assets based on their set min and max values. The Dense value specifies how many in area. Trees, bushes, and undergrowth asset lists are processed separately, which allows assets from each group to appear in the same area.

Emulating the Real World

You can use the LifeMap and assets (texture matrix and asset list) alone for procedural vegetation in fictional areas. This essentially uses a single asset list for entire area. However, most users want to use real-world data to have a simulated world that more closely emulates the real world. This is where you would use Biomes and the ability to define specific subsets of assets.

Default Biome

Each biome lists ground cover assets for specific area. You can use any dataset to define the specific areas. The base configuration uses a global terrestrial ecoregion dataset used by the One Earth Bioregions Framework (https://www.oneearth.org/bioregions-2023). This schema divides the earth surface into 8 biogeographic realms. Each Biome is classified into one of 14 major habitat types or biomes containing 847 smaller ecoregions.

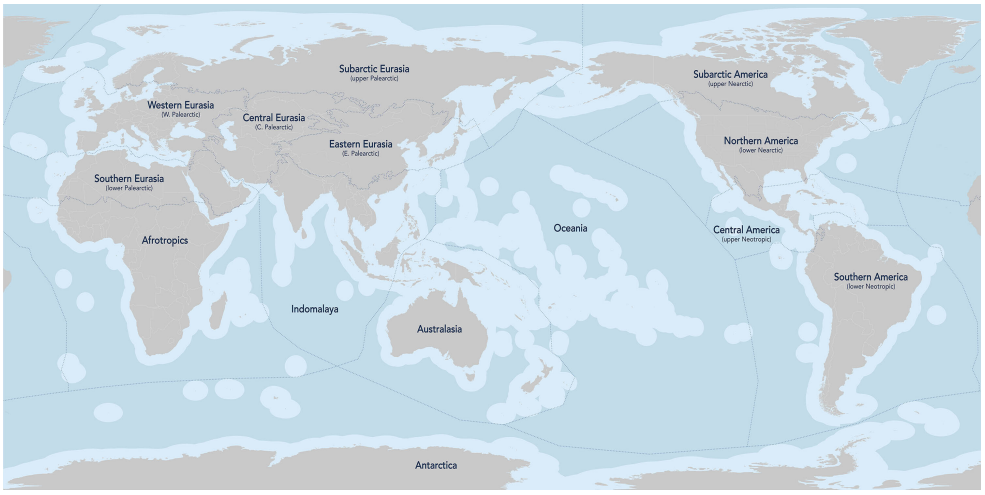
Realm

A biogeographic realm is the division of the Earth's land surface into 8 categories based on distributional patterns of terrestrial organisms These include:

- Australasia (AA)
- Antarctic (AN)
- Afrotropic (AT)
- Indo-Malay (IM)
- Nearctic (NE)
- Neotropic (NT)

- Oceania (OC)
- Palearctic (PA)

Figure 7. Realms[1]



Biomes

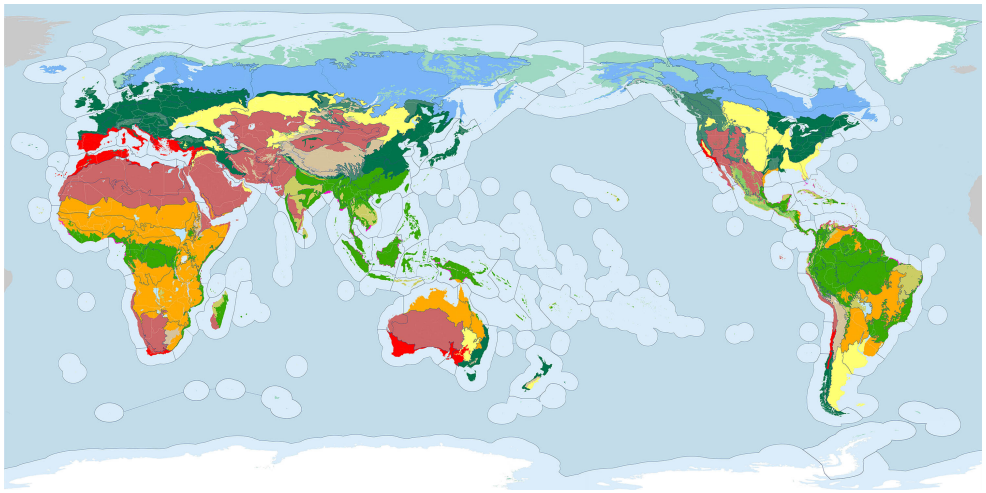
A Biome, sometimes referred to as a major habitat type (MHT), is a biogeographical unit consisting of biological community that has formed in response to physical environment, which is characterized according to predominant average conditions.

These include:

- Tropical and Subtropical Moist Broadleaf Forests (01)
- Tropical and Subtropical Dry Broadleaf Forests (02)
- Tropical and Subtropical Coniferous Forests (03)
- Temperate Broadleaf and Mixed Forests (04)
- Temperate Conifer Forests (05)
- Boreal Forests/Taiga (06)
- Tropical and Subtropical Grasslands, Savannas and Shrublands (07)
- Temperate Grasslands, Savannas and Shrublands (08)
- Flooded Grasslands and Savannas (09)
- Montane Grasslands and Shrublands (10)
- Tundra (11)
- Mediterranean Forests, Woodlands and Shrub (12)
- Deserts and Xeric Shrublands (13)
- Mangroves (14)

[1] Image from "One Earth Bioregions Framework". (2024, November 6). https://www.oneearth.org/bioregions-2023/

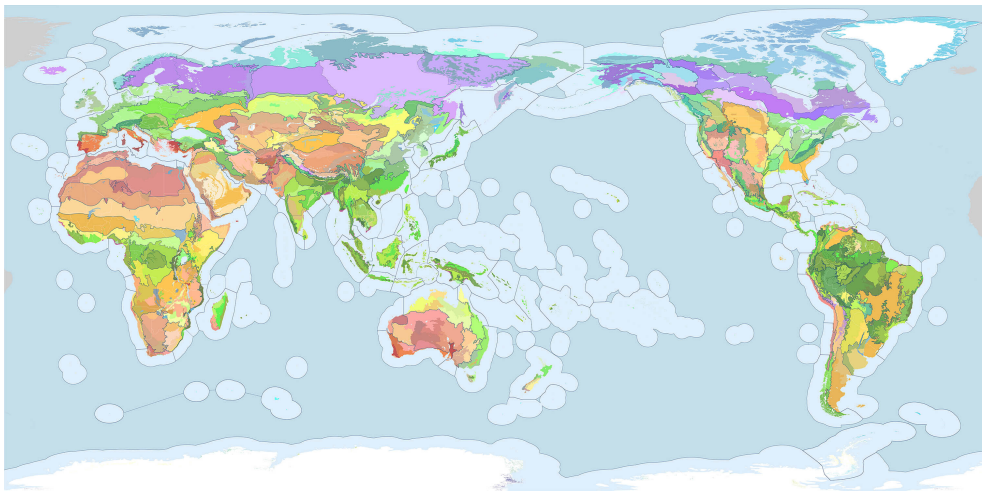
Figure 8. Biomes (or major habitat types)[2]



Ecoregion

An ecoregion is one of 847 areas classified by major global plant communities determined by rainfall and climate, organized by realm and Biome/MHT.

Figure 9. Ecoregions[3]



Using the LifeMap Biome

The default biome uses an Areal feature (shapefile) dataset. Each ecoregion can have list of geotypical assets for a geographic area. The Ecoregion-Realm-MHT definitions have a hierarchical relationship; if no assets are defined, then it defaults to its parent. If there are no assets defined for a specific ecoregion, it then checks its parent Realm-MHT. If there are no assets defined for the associated Realm-MHT, it checks its parent MHT.

---

[2]Image from "One Earth Bioregions Framework". (2024, November 6). https://www.oneearth.org/bioregions-2023/

[3]Image from "One Earth Bioregions Framework". (2024, November 6). https://www.oneearth.org/bioregions-2023/

Southern California Example

As an example, let's look at the specifics for the Southern California Coast. The details for this area include:

- Ecoregion: California coastal sage and chaparral (422)
- Realm: Nearctic (NE)
- MHT/Biome: Mediterranean forests, woodlands and scrub (12)

In this example, the specific BIOME_ID would be 12-NE-422.

We could define very specific assets definitions for Southern CA coast (CA coastal sage and chaparral, 12-NE-422) or default to more general mapping for all Nearctic, Mediterranean forests, woodlands and scrub (NE-12) or default to general global Mediterranean forests, woodlands and scrub (12).

Biomes can also tailor the density (fill) and comparative number (weight) for each asset.

Benefits of Biomes

Biomes enable seamless support for different vegetation anywhere in world. Each biome provides a pageable subset of vegetation models, which means that only those currently needed are resident in memory. The hierarchical approach ensures global coverage of reasonable vegetation, and users can define assets for each MHT and refine with regional vegetation by defining assets by ecoregion.

Using Land Cover and Biomes with the LifeMap and Assets

Land cover is "real" data about the earth, defined as raster or features data according to a specific set of classes. The PE2 approach can use land cover data to define specific LifeMap Lush, Rugged, Dense values. As with the legacy procedural vegetation approach, PE2 supports a hierarchy of datasets with varying resolutions.

2.13.2 More Mechanisms to Control LifeMap Values and Asset Selection

These sections describe other mechanisms to refine how the LifeMap values are generated and how the associated assets are chosen.

Traits

Ecoregions still can cover large areas. Also, some land cover classes do not necessarily fit into the Lush, Rugged, Dense schema. When defining the mapping of land cover classes to the Lush, Rugged, Dense values, you can also specify Traits to make areas more specific and match details provided by land cover. Essentially this creates a new sub-biome with only those assets marked with the trait. Traits are also applied to specific model assets, which enables specific assets to be used for specific land cover classes.

Traits can be any string. The MAK default configuration includes some standard traits to better parallel Copernicus classes. These include:

- ENL. Evergreen Needleleaf.
- EBL. Evergreen Broadleaf.
- DBL. Deciduous Broadleaf.

- DNL. Deciduous Needleleaf.
- SHRUB. The default for the bushes group. Enables land cover classes to be defined to specifically include assets from the bushes group.
- GRASS. The default for the undergrowth group. Enables land cover classes to be defined to specifically include assets from the undergrowth group
- CROP. Cultivated agriculture crops. Crop vegetation does not fit neatly into the Lush value selection. The CROP trait enables vegetation to be identified as only crops and could be ignored in normal processing but used for specific land cover classes.
- URB. Urban. Typically used to simulate mowed grass. The URB trait enables land cover classes to essentially have no models assets, while still allowing the Lush and Dense values to select a vegetated surface texture.

Example

We can define a White Spruce model asset using the ENL trait:

```
<asset name="WhiteSpruce" traits="ENL" >
  <url>../../ModelData/Vegetation/Biome/Trees/EvergreenNeedleLeaf
      /Spruce/WhiteSpruce/WhiteSpruce.medf</url>
  <side_url>../../ModelData/Vegetation/Biome/Trees
      /EvergreenNeedleLeaf/Spruce/WhiteSpruce
      /WhiteSpruceSide_png.meif</side_url>
  <top_url>../../ModelData/Vegetation/Biome/Trees
      /EvergreenNeedleLeaf/Spruce/WhiteSpruce
      /WhiteSpruceTop_png.meif</top_url>
</asset>
```

Then, for our land cover class—Copernicus, Closed Forest, Evergreen Needleleaf (111)—we can also add the ENL trait:

```
<mapping value="111" dense="0.8" lush="0.026" rugged="0.3"
    traits="ENL" />
```

For our Biome, we have an asset list defined for Temperate Broadleaf and Mixed Forests:

```
<biome id="04" parent="00" name="Temperate Broadleaf and
    Mixed Forests">
  <assets>
    <asset name="WhiteSpruce"/>
    <asset name="RedMapleSpring" fill="0.1"/>
    <asset name="HolmOak" fill="0.3"/>
  </assets>
</biome>
```

For any areas of the land cover class that falls into the Biome area, we have essentially created a new biome—Temperate Broadleaf and Mixed Forests-ENL—which would only have the single White Spruce asset.

```
<biome id="04" parent="00" name="Temperate Broadleaf and
    Mixed Forests-ENL">
  <assets>
    <asset name="WhiteSpruce"/>
  </assets>
</biome>
```

Required Traits

If the land cover does not specify a trait, any asset will be used even if it has a trait defined. However, we can require an asset to only be considered if the land cover also has the trait defined. We can add `traits_required="true"` to the asset along with the trait. If land cover also has the associated trait, the asset is used. Otherwise, it is ignored.

Multiple Biomes

The Biomes definition can have layers, which allows you to have multiple biome datasets. Biomes specify the specific assets lists for the Biome. As with any layer definition, the last layer defined takes precedence. The default approach used is the TNC/WWF scheme, but you can use any dataset to define different biomes containing different subsets of assets.

MAK has used the CA Fveg land cover in the past and continues to do so for PE2. However, the CA Fveg dataset uses a WHR Type classification scheme that defines 59 classes (see https://wildlife.ca.gov/Data/CWHR/Wildlife-Habitats). Each class defines 15 m land cover, which also includes a list of specific vegetation for area. Since it defines a list of specific vegetation, we can also use it as a biome.

Coastal Scrub (CSC) Example

CSC is one of the primary land cover areas in the Camp Pendleton region. It defines a number of primary plant species (Coyotebush, California Buckwheat, Sage Species) as well as many other secondary species. Therefore, we can define a configuration that uses CA Fveg for 15 m ecoregions to override more general TNC/WWF biome areas.

Weighted Effects on LifeMap

The LifeMap can use a number of data sources to determine the Lush, Rugged, or Dense values. The effects can be defined in the configuration, and each can have a weight, or how much that specific data will affect the resulting values.

Currently, the MAK ONE default configuration files use land cover as the primary mechanism for setting the Lush, Rugged, or Dense values, which ensures that coverage has the highest weight. However, other factors can modify the resulting values.

Terrain Effect

The terrain effect is a LifeMap setting for the percentage of how actual terrain can modify the values. It uses slope from the specified elevation layer to adjust LifeMap values at that location. As slope increases in steepness, Rugged increases while Dense and Lush both decrease. As a result, a steep slope shifts choices for textures, vegetation density, and choice of vegetation.

Noise Effect

The noise effect inserts random noise to add variation and help compensate for lower-resolution land cover. The MAK ONE default configuration currently uses a low 10% effect so as to not deviate too far from real world land cover data.

Imagery Effect

The imagery effect uses the specified imagery layer color to modify Dense, Lush and Rugged values. Imagery color is decomposed into "greenness" and "redness". Dense increases with greenness. Rugged increases with redness. Lush increases with greenness multiplied by darkness. As an example, for a green spot, the darker it is, the more lush it is.

> The imagery effect is not currently used in standard configuration. Use of the imagery effect would force the requirement that the VR-Forces simulation engine also download all imagery, which would negatively affect performance. If you are using VR-Vantage as a visualization-only solution, then enabling the effect could tailor the procedural imagery and ground cover to match the imagery more closely.

Density Mask Layer

Like the legacy procedural vegetation approach, you can use a mask layer to clear areas of vegetation to compensate for lower-resolution land cover. For example, you can remove vegetation for roads, parking lots, buildings, and so on. This is defined as the LifeMap mask_layer to specify Density Mask image. The image is a FeatureSDF (or CompositeImage comprised of multiple FeatureSDF layers). The FeatureSDF specifies a transition range to blend with surrounding LifeMap values. The mask layer sets Lush, Dense, and Rugged to 0.

For procedural imagery, the Lush value is set to 0 value and dense is 0, which means that no overlay texture is used and only the first (0) overlay image is used. Ground cover uses the 0 Lush index, but Dense is 0, therefore no assets are rendered.

Override Textures

Override textures are used to map land cover values to textures that do not fit LifeMap Lush, Rugged, Dense concept (for example, water, snow, ice, or man-made surfaces). The images have same PBR effects as normal materials.

However, the current limitation is that edges of different areas are not blended with adjacent materials. As a result, the images are rendered at the resolution of the land cover data. This can result in a "blocky" visual, depending on the resolution. You can up-sample the layer to reduce jagged edges, but it is still not perfect and definitely blockier than rest of procedural textures.

2.13.3 PE2 for the VR-Forces Simulation Engine

PE2 is used for the visuals but also has mechanisms to support features needed by the simulation engine.

LifeMap Soiltype

The VR-Forces simulation engine uses soiltype and roughness as part of the movement system. Therefore, in addition to Lush, Rugged, and Dense, the coverage layers define an associated surface material. The values used are CDB Base Materials. For example:

- BM_VEGETATION
- BM_VEGETATION-BRUSH
- BM_LAND-GRASS
- BM_ASPHALT
- BM_SAND
- BM_LAND-MARSH
- BM_WATER

The VR-Forces simulation engine maps the CDB base materials values to current the list of soiltypes. The use of CDB base materials enables the simulation engine to expand the list of available surface materials in the future without requiring changes to the PE2 configuration files.

In the VR-Forces sim engine, soiltypes such as BM_VEGETATION are remapped to VR-Forces basic soiltype (see Table 35 on page 253). This mapping is defined in the *./appData/settings/vrfSim/landCoverDataSurfChar.map* file, which is used for the global mapping of all terrains. In addition to this mapping, you can create a terrain base mapping.

For example, you could create a file named *Ala Moana.landCoverDataSurfChar.map* in the *./SharedData/19/latest/TerrainData/TerrainConfiguration* directory that would update and append to that global *landCoverDataSurfChar.map* file. If *landCoverDataSurfChar.map* has the line:

```
Match       BM_WATER      deepriver
```

and *Ala Moana.landCoverDataSurfChar.map* contains

```
Match       BM_WATER      shallowlake
Match       BM_WATER-MUD  mud
```

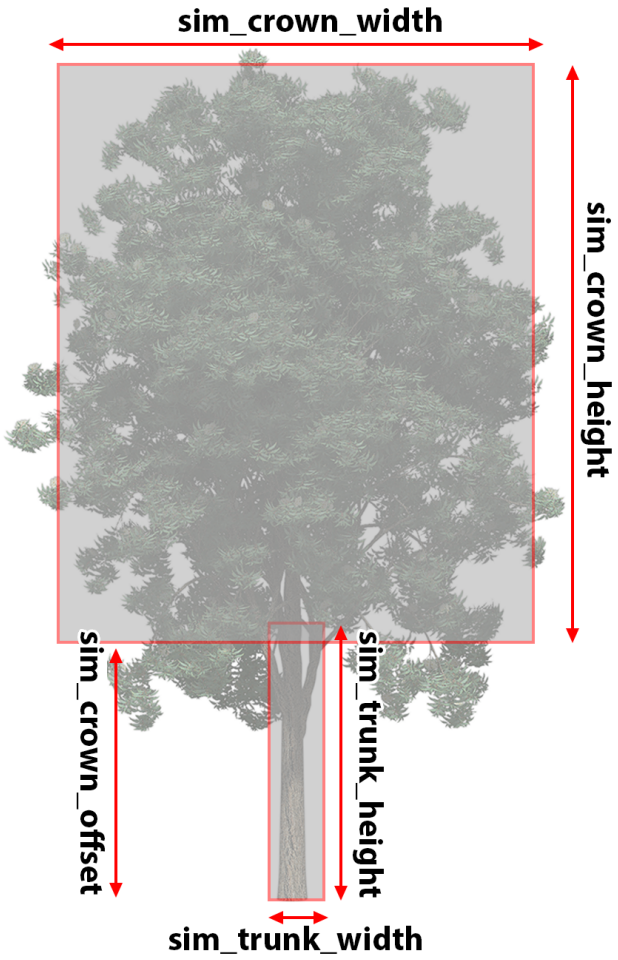
then the soiltype in Ala Moana is `shallowlake` and remains `deepriver` in other terrains such as MAK Earth, and Ala Moana also has the `mud` soiltype while other terrains do not.

SIM Models

Model assets can include a simplified simulation-specific model which, when placed as props, are used for simulation calculations. For a model, *mymodel.medf*, there can be a corresponding *mymodelSIM.medf*. For PE2, these models are converted into a set of values used in the calculations. These include:

- sim_trunk_height
- sim_trunk_width
- sim_crown_offset
- sim_crown_height
- sim_crown_width

Figure 10. Billboard attributes



The values are derived from the 3D model and stored in *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/simVegetation.xml*. They are generated from the SIM model. If there is no SIM model, they are generated from the bounding box of the full 3D Model. If you add additional models, you can regenerate this file by running *simTreesTool.exe* to reprocess all the models in the *./SharedData/19/latest/ModelData/Vegetation/Biome* directory. See "The simTreesTool" on the facing page.

Assets that have a corresponding SIM model must also have a `<width>` defined in the catalog. The width is derived from SIM model values. If `sim_crown_width` > 0.0, then use that value. Otherwise, use the `sim_trunk_width` value. This is required to ensure the VR-Forces simulation engine and GUI place visual and simulation models at same location.

Configuring Vegetation to Affect Line-of-Sight and Movement

By default, procedural vegetation models are not configured to affect line-of-sight (LOS) and movement. If you want the models to have an effect, edit *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs /biome.vegetation.config.xml*. In the Vegetation block, you will see `vrfsim:enabled="false"`. If you change this to true, the models will have an effect.

```
<Vegetation name="Vegetation" vrfsim:enabled="true">
```

Note that only the tree group affects the simulation. Currently, the bushes and undergrowth groups do not have any affect. The tree models affect LOS and the tree model information is included when calculating the navigation mesh (navmesh). Without a navmesh, entities will not avoid trees. Trees can block movement, but this requires the SMS to be configured to support it. By default, vehicles will drive through mostly-vertical polygons in the SIM (such as buildings). There is a flag that can be enabled in the SMS to check for this and block movement.

> Configuring trees to affect line-of-sight and movement loads them in the simulation engine. This can impact performance.

Using the Terrain Index to Reduce Polygon Counts

When you enable procedural trees, entities such as aircraft load all the vegetation in the tiles under its path. This causes the sim engine to load a high number of triangles that represent the trees which might not be needed for your use case. To improve performance, you can reduce the polygons for these "high flyers".

For example, *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/ biome.vegetation.config.xml* has this section at the end:

```
<mak_terrain_index name="worldwide90m_bathymetry">
  <url>../../../TerrainData/Terrain/World/Elevation
       /earthnew/VRTW-Worldwide-90m-Elevation-with-
       Bathymetry-Collection.index.bin</url>
  <max_lod>7</max_lod>
  <max_tolerance>500.0</max_tolerance>
  <min_tolerance>-500.0</min_tolerance>
</mak_terrain_index>
```

The simTreesTool

The simTreesTool is a command-line tool that gets the bounding box dimensions for geometry within an OpenFlight or FBX model.

The default behavior is to process files in the directory with simulation models, which are files that end with "sim". For example, if the tool finds both a *Tree.FBX* and a *TreeSim.FBX* file, it processes only *TreeSim.FBX*. The *Tree.FBX* file is skipped unless you specify that the tool process only non-sim model files or both sim and non-sim model files. (See Table 23 on the next page to learn about the options for the simTreesTool.)

The geometry must be commented as:

```
"@dis sim_trunk"
"@dis sim_crown"
```

The tool then writes out the following values in *./SharedData/19/latest /TerrainData/TerrainConfiguration/osgEarthCatalogs/simVegetation.xml*.

```
sim_trunk_height
sim_trunk_width
sim_crown_offset
sim_crown_height
sim_crown_width
```

For example:

```
<asset name="TreeName" sim_trunk_height="#.##" sim_trunk_width="#.##"
sim_crown_offset="#.##" sim_crown_height="#.##" sim_crown_
width="#.##"/>
```

where *#.##* is the measurements for the tree, if any.

The VR-Forces sim engine then uses the *simVegetation.xml* file to cause those sim tree assets to affect entities' line-of-sight and movement when users enable vegetation for the simulation engine; see "2.13.3 PE2 for the VR-Forces Simulation Engine" on page 125.

The syntax for simTreesTool is:

```
simTreesTool.exe  --XmlOutputfile outputfilename --TreesPathFolder
sourcefilepath [--nonSimFiles --simAndNonSimFiles --ignore_rest
--version --help]
```

Table 23 describes the command-line options.

Table 23. simTreesTool command-line options

Option	Description
(--nonSimFiles | -n)	Processes only files without simulation models. For example, if there is a *Tree.FBX* file and no corresponding file ending with "sim" (that is, there is no *TreeSim.FBX*), the tool processes the *Tree.FBX* file and calculates the bounding volume from the visual model.
(--simAndNonSimFiles | -s)	Processes all files in the directory unless there is a file that ends with "sim", at which point is skips the first file. For example, if the tool finds both a *Tree.FBX* and a *TreeSim.FBX* file, it processes only *TreeSim.FBX*. If there is no corresponding file ending with "sim", then the tool processes only the first file, which in this example is *Tree.FBX*.

Table 23. simTreesTool command-line options (continued)

Option	Description
(--TreesPathFolder | -f) *sourcefilepath*	The path to the folder where your sim tree files are located. In a typical MAK data installation, this is *C:/MAK /SharedData/19/latest/ModelData/Vegetation/Trees*.
(--XmlOutputfile | -x) *outputfilename*	The file path and filename for the XML output file that VR-Forces reads. In a typical MAK ONE data installation, this is *C:/MAK/SharedData/19/latest/TerrainData /TerrainConfiguration/Trees/osgEarthCatalogs /simVegetation.xml*.
(--ignore_rest | --)	Ignores all command-line options that are listed after this argument.
(--version | -v)	Displays version information.
(--help | -h)	Displays command-line usage information.

2.13.4 Configuration Files

The MAK Data Terrain Configuration includes a number of terrain (.earth) files that are configured to use the PE2 resources. MAK Earth (online) and Ala Moana are the primary terrains to review. However, the primary PE2 configuration is defined in *biome.config.online.xml*, which makes use of online (VR-TheWorld-hosted) resource, and *biome.config.local.xml*, which makes use of only local resources.

biome.config.online.xml

The *./SharedData/19/latest/TerrainData/TerrainConfiguration/biome.config.online.xml* file contains the top-level configuration for the PE2 resources. It defines the LifeMap and Biomes, and references the supporting Coverage, density mask, Biomes, BiomeCatalog, and Vegetation layers.

The Coverage layers are defined in *./SharedData/19/latest/TerrainData/ TerrainConfiguration/osgEarthCatalogs/biomes.landcover.coverage.online.xml*. The land cover starts with the global 100 m Copernicus dataset and ranges to high-resolution 0.5 m and 1 m datasets for a number of insets areas.

One technique to ensure consistency across the various land cover datasets is the use of presets to define the Rugged, Lush, Dense values, any traits, soiltype, and any override material, and then reference that preset in the coverage mapping. A number of presets are defined in *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/ coverage/presets.xml* and these values are used across the mappings for the various land cover datasets.

biome.config.local.xml

The *./SharedData/19/latest/TerrainData/TerrainConfiguration/biome.config.local.xml* file contains the top-level configuration for the PE2 resources using only local resources, which requires no connection to VR-TheWorld online or access to a local VR-TheWorld terrain server.

The Coverage layers are defined in *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/biomes.landcover.coverage.local.xml*. The land cover starts with the global 300 m Globcover dataset and includes a 2.5 m dataset for Hawaii.

2.13.5 MAK Implementation Guidelines

The PE2 approach is completely configurable and could support a wide variety of configurations. However, given the interdependency between the various settings, MAK defined a number of default configurations and guidelines for values that you should use when defining the Rugged, Lush, Dense values, and when using these defaults. The following sections detail the specific configurations and guidelines.

Procedural Imagery Assets

The AssetCatalog defines all the Materials, which are the individual procedural images used for surface materials, the LifeMapMatrix, which is the table that defines the substrate and overlay textures using the defined Materials, and the Models, which are the set of 3D vegetation assets, grouped by trees, bushes, and undergrowth. The default configuration is defined in osgEarthCatalogs/BiomesAssetCatalog.xml.

For the LifeMapMatrix, MAK defined a 2x6 table to use for the Procedural Imagery.

Table 24. LifeMapMatrix

	1	2	3	4	5	6
Rugged	sand	rough_sand	dirt	dirt_sm_ rocks	rocks_ dirt	more_ rocks_dirt
Lush	sparse	patchy_ grass_brn	patchy_grass_ grnbrn	patchy_ grass_grn	full_ grass	swamp_veg
Values	**0**	**0.2**	**0.4**	**0.6**	**0.8**	**1**

The Rugged line is the substrate, where images are chosen by the specified Rugged value. The Lush line is the overlay, where images are chosen by the specified Lush value. Given the 6 columns, the Values line provides the corresponding Rugged or Lush value that would choose that column. Values between those numbers would choose a blended percentage of each image for the 2 columns.

Lush and Trait Values for Copernicus Classes

Using the LifeMapMatrix table (Table 24 above), we can define a guideline for Lush values to use for various land cover classes so that we have an appropriate vegetation coverage for the surface materials. We can then use that value to configure the 3D assets min and max values and define the order of the assets as they are listed in the Biome definitions.

In addition to the Lush value, we can define a set of traits for each class. If there is a specified trait, we can select the specific tree, bush, or undergrowth trait to use. That ensures the appropriate selection of vegetation models.

The guideline defines this for Copernicus, but you can map it to similar classes in other datasets.

Table 25. Copernicus class mappings

Lush	Copernicus Class	Overlay Textures	Traits
0.000	60, Barren (bare or sparse vegetation)	sparse	
0.025	61, Desert-Sand (desert with bare or sparse vegetation)	sparse/patchy_ grass_brn	
0.050	113, Forest-Closed Deciduous Needleleaf (closed forest, deciduous needleleaf. tree canopy greater than 70%)	sparse/patchy_ grass_brn	DNL SHRUB GRASS
0.050	123, Forest-Open Deciduous Needleleaf (open forest, deciduous needleleaf. top layer are trees 15-70% and second layer is mix of shrubs and grassland)	sparse / patchy_ grass_brn	DNL SHRUB GRASS
0.100	20, Shrubs (Shrubs, Plants with persistent and woody stems, and without any defined main stem being less than 5 m tall)	sparse / patchy_ grass_brn	SHRUB GRASS
0.300	111, Forest-Closed Evergreen Needleleaf (closed forest, evergreen needleleaf. tree canopy greater than 70%.)	patchy_grass_brn / patchy_grass_ grnbrn	ENL SHRUB GRASS
0.350	121, Forest-Open Evergreen Needleleaf (open forest, evergreen needleleaf. top layer are trees 15-70% and second layer is mix of shrubs and grassland)	patchy_grass_brn / patchy_grass_ grnbrn	ENL SHRUB GRASS
0.500	115, Forest-Closed Mixed (closed forest, mixed)	patchy_grass_ grnbrn / patchy_ grass_grnbrn	
0.500	116, Forest-Closed (closed forest, not matching any of the other definitions)	patchy_grass_ grnbrn / patchy_ grass_grnbrn	
0.550	125, Forest-Open Mixed (open forest, mixed)	patchy_grass_ grnbrn / patchy_ grass_grnbrn	
0.550	126, Forest-Open (open forest, not matching any of the other definitions)	patchy_grass_ grnbrn / patchy_ grass_grnbrn	
0.550	40, Crops (Cultivated & managed vegetation or agriculture)	patchy_grass_ grnbrn / patchy_ grass_grnbrn	CROP and traits_ required=true
0.600	114, Forest-Closed Deciduous Broadleaf (closed forest, deciduous broadleaf. tree canopy greater than 70%.)	patchy_grass_grn	DBL SHRUB GRASS

Table 25. Copernicus class mappings (continued)

Lush	Copernicus Class	Overlay Textures	Traits
0.650	124, Forest-Open Deciduous Broadleaf (open forest, deciduous broadleaf. top layer are trees 15-70% and second layer is mix of shrubs and grassland)	patchy_grass_grn / full_grass	DBL SHRUB GRASS
0.700	30, Grassland (herbaceous vegetation, tree and shrub cover is less than 10%)	patchy_grass_grn / full_grass	GRASS
0.850	122, Forest-Open Evergreen Broadleaf (open forest, evergreen broadleaf. top layer are trees 15-70% and second layer is mix of shrubs and grassland)	full_grass / swamp_veg	EBL SHRUB GRASS
0.850	112, Forest-Closed Evergreen Broadleaf (closed forest, evergreen broadleaf. tree canopy greater than 70%)	full_grass / swamp_veg	EBL SHRUB GRASS
0.950	100, Moss (moss and lichen)	full_grass / swamp_veg	GRASS
1.000	90, Wetland (herbaceous wetland. lands with a permanent mixture of water and herbaceous or woody vegetation)	swamp_veg	GRASS
The following are override values. The textures do not come from the matrix.			
0.000	50, Urban (urban or built up)	NONE; separate asphalt layer from OSM features	URB
0.000	70, Snow-Ice (permanent snow and ice)	override	
0.000	80, Water (permanent water bodies: lakes, reservoirs, and rivers)	override, but for OSM-based water	
0.000	200, Water (oceans, seas)	override, but for OSM-based water	

Land Cover, Traits, and Assets

Using the Copernicus classes as the guideline, we can apply this to other land cover datasets. Coverage data mappings must specify lush, rugged, dense and soiltype values and, optionally, traits or material.

If the coverage defines a trait, only assets with that trait are included in the list. This is done separately for each of the tree, bush, and undergrowth categories.

Multiple traits can be defined. For example, the Deciduous Broadleaf Forest with shrub and herbaceous understory can have "DBL SHRUB GRASS", where assets used would include those that match any of specified traits. For example, there are land cover classes that have only shrubs and undergrowth layers which could use traits="SHRUB GRASS", or forest classes that do not have an understory which could use traits="ENL GRASS".

If the coverage specifies `trait` and `traits_required=true`, then any asset will be processed if the trait is also specified for that asset. For example, Crops (cultivated and managed vegetation or agriculture) uses CROP and has `traits_required=true`. For any biome asset list, only assets with the `trait=CROP` specified would be considered as part of the list. If multiple assets are in the list, the specific assets and amounts are chosen based on the lush value, the asset's min and max settings, and a normal distribution.

2.13.6 Defining Assets

The MAK ONE applications include a number of predefined image and model assets that have been configured for the various land cover and biome configurations. However, these can be replaced or extended by creating your own assets. "Procedural Imagery Assets" on page 130 describes the specifics for creating each, including MAK guidelines for naming the files and directory locations.

2.14. Legacy Procedural Vegetation

This information applies to legacy Procedural Earth 1 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 11 illustrates a portion of Sanibel Island, Florida, USA, using the MAK Earth (online) terrain. This terrain uses relatively low resolution imagery. Figure 12 illustrates the same location using the Procedural Earth (online) terrain. Notice that the procedurally generated vegetation makes this a much more attractive terrain. Figure 13 illustrates the same location with OpenStreetMap data included in the terrain. Figure 14 shows a closeup of the terrain.

Figure 11. Sanibel Island, MAK Earth (online)

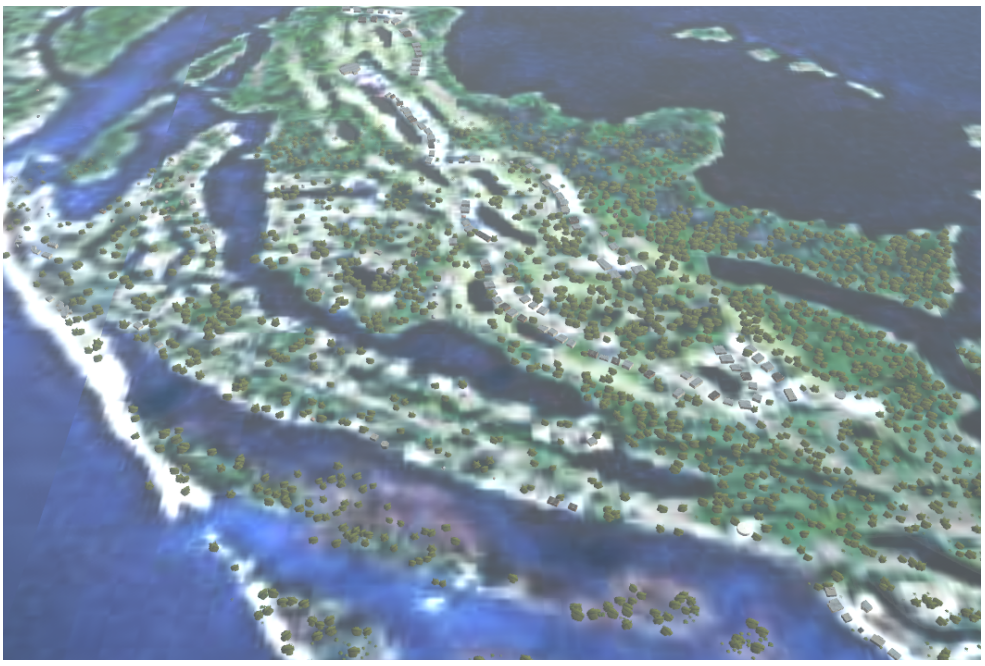


Figure 12. Sanibel Island, Procedural Earth (online)

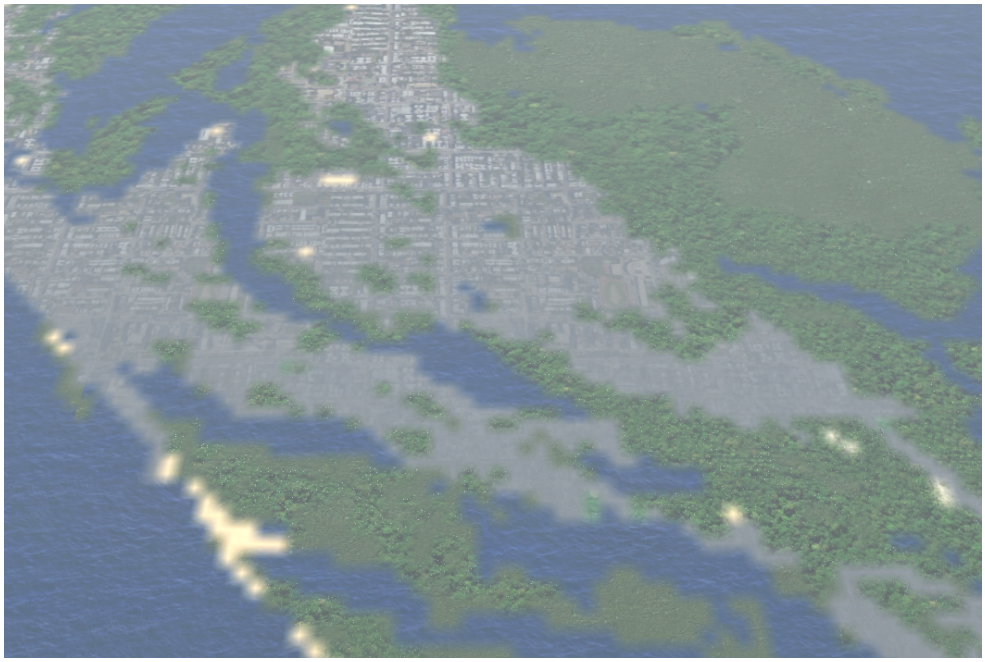


Figure 13. Sanibel Island, ProceduralEarth - OpenStreetMap (online)

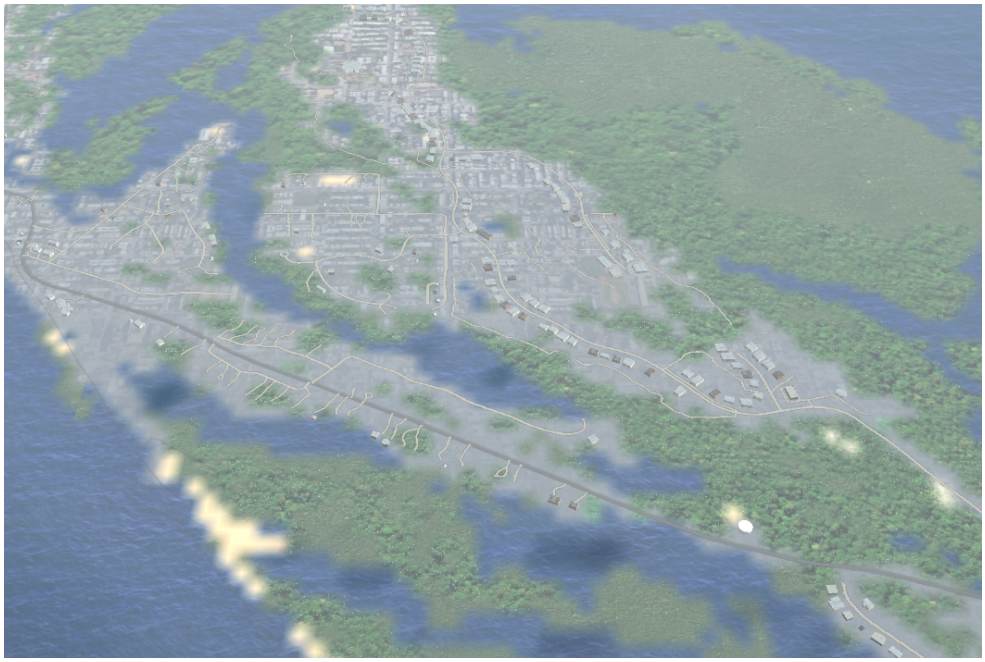
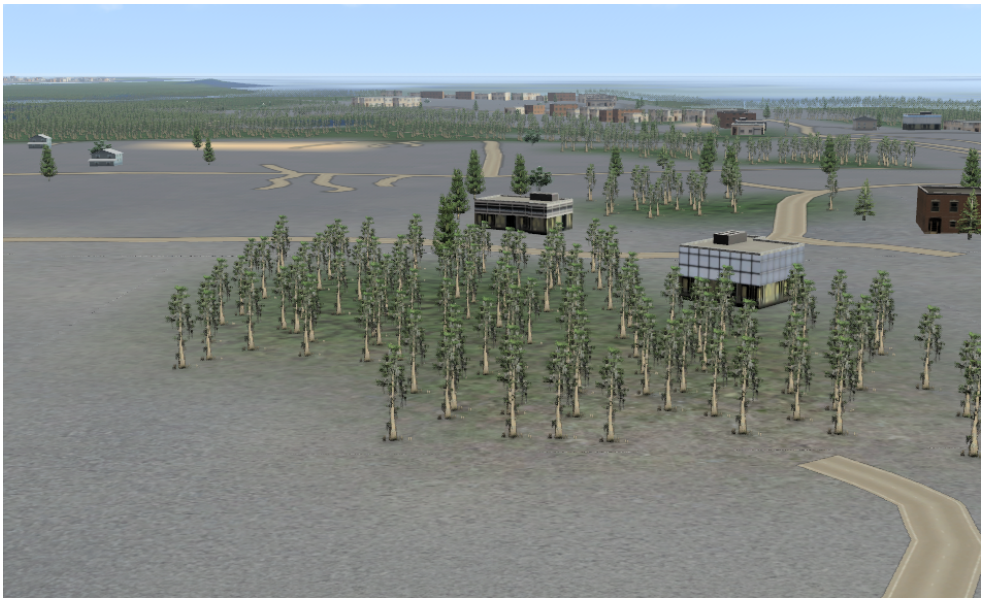


Figure 14. Sanibel Island closeup, ProceduralEarth - OpenStreetMap (online)



2.14.1 Configuring Procedural Trees for VR-Forces

VR-Forces can support procedural trees. This allows the simulation engine to understand the location of the trees, causing them to affect line-of-sight and invoke 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. See Figure 10 on page 126.

- **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).

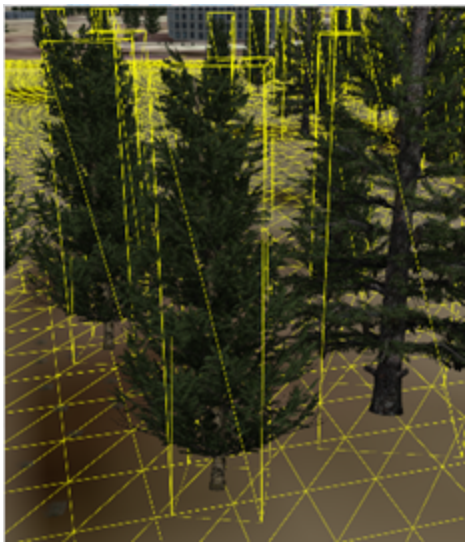
Although most of the attributes are optional, if you do not set any of them, the bounding box around the tree will be too large to be useful. At a minimum, you should set the `sim_crown_height` and `sim_crown_width`, for example:

```
<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"/>
```

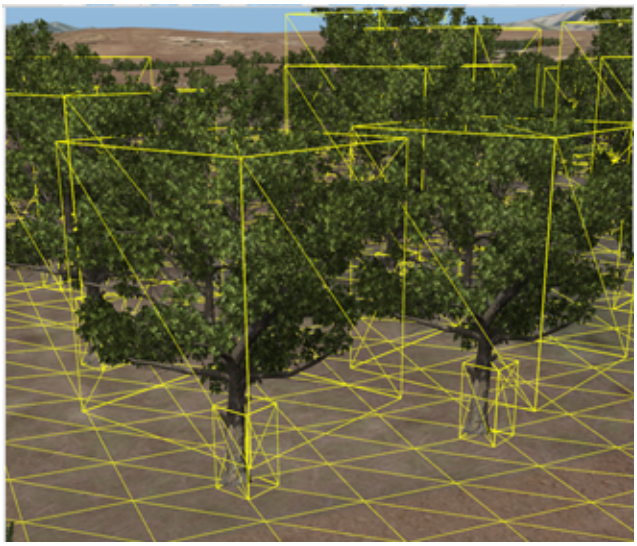
With these two optional parameters, VR-Forces creates one bounding box for the entire tree using the height and width values (Figure 15).

Figure 15. 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 16).

Figure 16. 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*.

2.14.2 Configuring Whether Procedural Vegetation Loads in the Simulation Engine

You can choose whether the VR-Forces sim engine loads procedural trees and other vegetation from the terrain.

When you load procedural vegetation in the sim engine, you can regenerate the navigation mesh in VR-Forces such that entities will avoid the plants. The vegetation can also block line of sight.

On the other hand, loading procedural trees in the sim engine can impact performance. If you are not concerned about your entities interacting with vegetation, you might want to disable procedural vegetation to improve performance.

This feature requires an NVIDIA graphics card.

In the *./appData/settings/vrfSim/terrainInterfaceConfig.mtl* file, edit the `procedural_sim_vegetation_enabled` parameter and set it to either True or False.

2.15. 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. Procedural terrains, on the other hand, 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 tools 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. Which tool you use depends on which procedural vegetation approach you used in your terrain configuration.

2.15.1 Exporting Terrains that use the Biome-Based Procedural Vegetation Engine (PE2)

For terrains using the PE2 approach for procedural vegetation, use *osgearth_exportvegetation.exe*. It is a command-line tool located in *./bin64* and its syntax is:

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

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 area to process, in degrees.
- *shapefile*.shp is the output shapefile.
- *name* is an asset property name as an attribute. You can use the `--include-asset-property` command-line option multiple times for multiple asset property names.

You most likely need to append *.template* to your specified earth file. This is necessary because most MAK earth files use templates (specifically the {%...%} notation), and the template preprocessor is an OSG plug-in. Therefore, you must append *.template* for the preprocessor to navigate all the template "include" directives.

The tool exports only features for the Trees group. No bushes or undergrowth are included.

The shapefile contains point features for each procedural tree. Each feature has these 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 assets attributes that you want to include as feature attributes, you can use the `--include-asset-property` option to name them. This might include the URL for the full model path, or name for the asset name.

2.15.2 Exporting Terrains that use the Legacy Procedural Vegetation (PE1)

For terrains using the legacy (PE1) approach for procedural vegetation, use *osgearth_exportgroundcover_splat.exe*. This command-line tool is located in *./bin64*and its syntax is:

```
osgearth_exportgroundcover_splat.exe 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.
- *swlongswlatnelongnelat* are the extents of the terrain in degrees.
- *shapefile*.shp is the output shapefile.
- *property* is a billboard property name as an attribute. You can use `--include-billboard-property` multiple times for multiple billboard property names.

The shapefile contains point features, one per instance of GroundCover. Each feature has these 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 to include as feature attributes, you can use the `--include-billboard-property` option to name them.

2.16. Rendering 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.

Rendered features (including buoys) are organized into layers. In the earth file, the names in the `styles` block specify the layer to which the streamed features belong. 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 must correspond to the style names you used in the earth file.

This 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" build_spatial_index="true">
    <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: 0.000045;
      altitude-offset:     0;
      }
  </style>
  <selector class_expr="getStyleClass()"/>
</styles>
```

2.16.1 Specifying Model Definitions for 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 S-57 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 needing to specify many model definitions.

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

This 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 such as:

```
"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:

```
BOYSHP,,Nun,Can,Spherical,Pillar,Spar,Barrel,Super,Ice
COLOUR,,White,Black,Red,Green,Blue,Yellow,Grey,Brown,Amber,
    Violet,Orange,Magenta,Pink
COLPAT,,HorizontalStriped,VerticalStriped,DiagonalStriped,
    Squared,Striped,BorderStriped
```

Therefore, the model definition would be BuoysBlueSpherical, since the enumerated COLOUR value `5` (the fifth value in the list) is mapped to `Blue`, and the BOYSHP value `3` is mapped to `Spherical`. Some of these attributes, such as COLOUR, may be lists of enumerated values. The strings for enumerated lists are concatenated. The 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 also be all numerics, all capital letters, or a mix of both. 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. When the OBJNAM attribute value is provided in the marker string and 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.

2.16.2 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 that you understand 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 they ship with.

Each buoy shape (can, pillar, nun, and so on) is a separate model. Each model has a texture atlas that has all of 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 141 when mapping attributes to model definition substrings:

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

In this example, all of 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 can 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, see "2.16.6 Daymarks" on page 146 and "2.16.7 Topmarks" on page 146.) The model definition for the buoy specifies a key (found in the *.dtxc* file) to be used for the main texture (`mainTextureKey`) and for the topmark's texture (`topmarkTexturekey`).

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

2.16.3 Configuring Beacons

You can specify beacon feature styles the same way you specify buoys. You can add any number of beacon styles names. The S-57 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" build_spatial_index="true">
    <!-- From S-57 data -->
    <url>../../TerrainData/Terrain/Hawaii/Features/US5HA52M/BOYLAT.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: 0.000085830688476562 50;
        altitude-offset:     0;
        }
    </style>
  </styles>
</model>
```

Model definitions for beacons work the same way as those 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
```

2.16.4 Configuring 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`. This example shows a `model` block for a light:

```
 <model name="LightsUS5MA10M" driver="feature_custom"  enabled="true">
   <OGRFeatures name="s57-data" build_spatial_index="true">
     <!-- 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: 0.000045;
         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:

- 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()
```

2.16.5 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. For example:

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

2.16.6 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/19/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`).

2.16.7 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.

2.16.8 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. That allows developers to implement it. The assumption is that not many of the other sequence types will be used, but it is data-specific. Once we know what sequences are needed for the S-57 data in use, it should be relatively straightforward to implement those additional sequence generation methods.

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

2.17. 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.

The following is an example for Fort Hood, TX, USA.

```
<!-- 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.

2.18. Options for earth Files

An earth file can have `<option>` elements that affect the entire map. The terrain options elements are values to configure the REX Terrain Engine, which is osgEarth's primary terrain rendering system.

Table 26 describes the terrain option elements.

Table 26. terrain option elements

Element	Description
tile_size	Number of vertices, in each dimension, per tile. Must be a power-of-2 plus 1. More tiles result in more polygons, both for rendering and what is used by VR-Forces and VR-Engage. Default: 17.
morph_ terrain	Perform vertex position blending across LODs. Default: true.
morph_ imagery	Blend image layers across LODs. Default: true.
progressive	Whether to load terrain tiles in LOD order, low to high. Default: false.
skirt_ratio	Edge skirt height, as a percentage of tile width. Default 0.001.
normal_maps	Whether to generate normals from elevation data. Default: true.
min_expiry_ time	This is the minimum number of seconds before terrain tiles will page out if not used. For fast moving eyepoints, where you do not want to run out of video memory like flight sims, you may want to have this fairly low, especially if your memory usage is approaching 100%. Default: 0 (immediately available to page out when not in view).
min_expiry_ frames	The number of frames that a terrain tile has not been needed before it can be considered for expiration. Default: 0.
merges_per_ frame	Controls the number of new tiles that will merge each frame. Default: 1.
min_tile_ range_factor	The range between when different tile LODS become visible. When smaller, it means you get a lower LOD closer to the camera, for ground operations lower is better. Lower also helps performance but setting this lower will also mean less data is visible farther out. From the air this looks bad. A good range seems to be between 2 and 5, with 5 looking better from the air but performing worse. Default: 7.0.
expiration_ range	Minimum range (distance from camera) beyond which unused terrain data is eligible to expire. You can set this to force terrain tiles within the range to remain in memory. Default: 0 (unused tiles expire based on min_expiry_time).
first_lod	This is the lowest LOD that the terrain will render. Default: 0. Terrain starts at this LOD and subdivides from there.

Table 26. terrain option elements (continued)

Element	Description
cast_shadows	Whether the terrain should cast shadows on itself. Default: true. If false, non-terrain objects will cast shadows on the terrain, but terrain elements (such as mountains) will not. Default: false.
color	The color of the terrain skin if there is no imagery on it. This is mostly irrelevant. Default: #ffffff.

The `feature_fading` option element provides default settings for fading for the model and building layers. 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.

The feature_fading option has the following attributes:

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

It only applies to layers that use indirect rendering. The defaults can be overridden in model and building layers directly.

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.

The `screen_space_error` option element sets the value (in pixels) for the minimum size for when procedural vegetation models will appear.

The example terrains include a sample `<options>` configuration that can be included in any earth file. The file is *options.terrain-rex-default.xml* and has the following values.

```
<options>
  <terrain driver         = "rex"
    tile_size             = "17"
    morph_terrain         = "true"
    morph_imagery         = "true"
    progressive           = "true"
    skirt_ratio           = "0.1"
    normal_maps           = "true"
    merges_per_frame      = "1"
    min_tile_range_factor = "3"
    expiration_range      = "10000"
    min_expiry_time       = "60"
    first_lod             = "3"
    cast_shadows          = "true"
    color                 = "#ffffff" >
  </terrain>
```

```
  <feature_fading duration  = "0.3"
                  max_range = "6000"
                  attenuation_distance = "1000" />
  <screen_space_error>15</screen_space_error>
</options>
```

There is also a file, *options.terrain-default.xml*, for the old MP Terrain Engine. However, this remains only for a few legacy configurations and is not what you should use for any new configuration.

2.19. Differential Processing of Elements and Attributes

VR-Vantage and VR-Engage share earth files with VR-Forces. The differential processing described in this section would not be used by VR-Vantage nor VR-Engage. They ignore `vrfsim:` syntax. This topic simply explains why the earth files provided with VR-Vantage and VR-Engage are written this way.

There are times when VR-Forces users want the GUI and sim engine 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 sim engine. For example, if you want different values for the `name` attribute in a model element, specify it:

```
<model name="gui_name" vrfsim:name="sim_engine_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, this code specifies that the application use the `<trees>` element in the GUI and the `<vrfsim:trees>` element in the sim engine:

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

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

2.19.1 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 `FeatureModel` element.

The following example of differential feature processing is from a terrain *.xml* file. It is the section used to extrude buildings. `vrfsim:enabled` is set to true. The value for the `typename` attribute is applied to GUI processing and the value for the `vrfsim:typename` attribute is applied to sim engine processing.

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

If a model element uses the `simple` driver, it uses the value of the `enabled` attribute to determine whether 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. Paged and Streaming Terrains

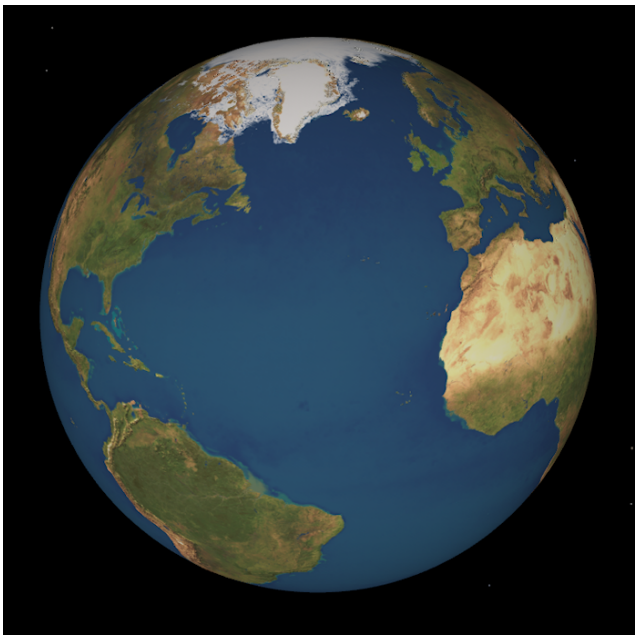
This chapter explains how to use terrain servers for paging and streaming terrains. It also provides information about preparing paged terrains for use with MAK ONE applications.

3.1. Connecting to Terrain Servers 154
3.1.1 Adding Terrain Server Connections 154
3.1.2 Connecting to Terrain Servers Through a Proxy 155
3.2. Preprocessing Paged Terrains 155
3.3. Manually Paging In Terrain in VR-Forces 156
3.4. 2D View Layers 157
3.4.1 2D View Layers Overview 157
3.4.2 Defining 2D View Layers in earth Files 157
3.5. Procedural Airports 158
3.5.1 Configuration for MAK ONE Applications 158
3.5.2 Data Files 159
3.5.3 Editing Procedural Airport Data 162
3.5.4 Editing Airport Lighting 163
3.5.5 Airport Terminals 166
3.5.6 Other Procedural Airport Data 166
3.5.7 Adding Custom Airfield Lighting 167
3.5.8 Airfield Lighting for Inset Airports and Runways 172
3.5.9 Runway Offset to Avoid Z-Fighting 173

3.1. Connecting to Terrain Servers

MAK ONE applications use the osgEarth libraries for loading terrains from remote or local data sources. To access terrain servers, you must configure the appropriate layers in an earth file. Terrain servers support streaming of terrain data-on-demand, which makes them ideal for displaying terrains that have large amounts of data.

Figure 17. Geocentric terrain



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 that server. One of the terrains is MAK Earth (online), which is a whole earth terrain featuring data for anywhere on the globe.

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.

3.1.1 Adding Terrain Server Connections

Remote data from a terrain server is managed in the same manner as a local dataset. Data can be added as Layers. Layers can be combined, both using local or remote data sources. Remote data can be rendered in a variety of ways.

For example, instead of a GDALImage to define a layer that accesses a local file, you could use:

- `TMSImage` to access an OSGeo Tile Map Service (TMS) server hosted on VR-TheWorld or other terrain services.
- `WMSImage` to access an OGC Web Map Service (WMS) server hosted on VR-TheWorld or other terrain services.
- `XYZImage` to access a pretiled image source, either on a terrain server or local.

As with a GDALImage, there is a URL option. Instead of referencing a local file, however, the URL would be a web address. For example:

```
<TMSImage name="Worldwide High-Resolution Imagery">
  <url>http://vr-theworld.com/vr-theworld/tiles/1.0.0/142/</url>
</TMSImage>
```

The same applies to elevation or feature data.

Unlike local data, remote data is streamed when needed. Only the tiles for the current view, which includes the geographic extents and level of detail, are accessed. As tiles are streamed from the terrain server, they are cached locally. Subsequent requests for those tiles are accessed from the cache rather than requiring another request to the server.

All this is contained in an earth file, which can mix remote and local data as required.

> All earth files must be placed in *./SharedData/19/latest/TerrainData/TerrainConfiguration*.

3.1.2 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. They support basic authentication and may support domain authentication if your proxy allows it to be accessed through basic authentication. To set up authentication, set these 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.

3.2. Preprocessing Paged Terrains

MAK ONE applications support preprocessing external references used by paged terrains. They 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, you might experience significant performance issues the first time you load one of these terrains. 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/19/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, see "15.19. Compressing Model Files (medfTool)" on page 381.

3.3. 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, when 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, and whether it blocks waiting for the terrain to finish paging in. You can also create an area that does not load the vegetation in the terrain data.

Terrain page-in areas are created in the same manner as any other tactical graphic area. You can create them as the scenario progresses or you can include them in plans, which means you can specify that the terrain pages in only when the scenario needs it. You can also attach a terrain page-in area to a simulation object to ensure that terrain in front of the simulation object pages 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 sim engine.

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

1. In VR-Forces, open the Create Object Palette.
2. Optionally, click the Terrain filter ( ).
3. Optionally, click the Areal tag.
4. On the Force line, click the button that represents the force you want this area to be assigned to.
5. In the list of control objects, select the area you need:
   - Terrain Page-in Area or Terrain Page-in Area (no vegetation). Does not block waiting for the terrain to finish paging in, with or without vegetation.
   - Blocking Terrain Page-in Area or Blocking Terrain Page-in Area (no vegetation). Blocks until the terrain finishes paging in, with or without vegetation.
6. Create the area.

3.4. 2D View Layers

Terrains can include multiple layers to display in the 2D projection (plan view). There is a terrain, *./SharedData/19/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.

3.4.1 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, street 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 either:

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

MAK ONE applications track which layers you are rendering and, after the length of time specified in the General Application Settings, unload 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.

3.4.2 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.

This 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" lod_blending="true">
  <mapgroup>air</mapgroup>
  <enabled>false</enabled>
  <url>../../TerrainData/Terrain/HawaiiTifExample/otw
      /AlaMoanaNewRetouched.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" enabled="true"
     vrfsim:enabled="true" >
  ...
  <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.

3.5. Procedural Airports

MAK ONE applications support global procedurally generated airports. You can add procedural airports to earth files and include them with any terrain. Procedural airports can include both visual components as well as features to support aerodromes simulated by VR-Forces.

Airports in terrains use feature data derived from open source data. They are configurable, and you can add other airports as needed.

3.5.1 Configuration for MAK ONE Applications

Most of the configuration for procedural airports is done as part of the Aerodrome catalog. This is defined in *./SharedData/19/latest/TerrainData/TerrainConfiguration /osgEarthCatalogs/aerodrome_catalog.xml*. The catalog uses art assets and features to define the various components of the airports.

In addition to the catalog, there are several other features used to define flattened elevation for the airports, add windsocks, add NAVAID (navigational aid) models, and define FeatureModels used by the VR-Forces simulation engine.

Use the *MAK Earth (online)* terrain file as a reference for how to configure procedural airports. The earth file defines the procedural airports by referencing two primary files in the *./SharedData/19/latest/TerrainData/TerrainConfiguration* directory:

- airports.worldwide.aerodrome.xml
- airports.worldwide.features.xml

The configuration in *MAK Earth (online)* is more complex because it uses both procedural airports and inset (OpenFlight) airports, augmented with elements from the procedural airport configuration.

3.5.2 Data Files

The data used to render the procedural airports consists of shapefiles generated from the X-Plane 11.53 data. Most of the files are derived from the Airport Data (APT.DAT) file, but there are also shapefiles associated with the EARTH_NAV.DAT, EARTH_AWY.DAT and EARTH_FIX.DAT files.

The shapefiles are located in *./SharedData/19/latest/TerrainData/Terrain/World/Features* and the subfolders *apt_shapes* and *apt_shapes/NAVAIDS* (Table 27).

> You can find the Airport Data file specification at the X-Plane Developer website, https://developer.x-plane.com.

Table 27. Shapefiles for procedural airports

Filename	Details	Notes
./SharedData/19/latest/TerrainData/Terrain/World/Features		
APTBoundary_all	Area feature for Airport boundary	Used for airport flattening, which means some boundaries have been smoothed or simplified for better results. Used to define the FeatureModel for The VR-Forces simulation engine.
aeroway_terminal_v4	Area feature for the OSM Aerodrome terminal buildings footprints	Referenced from the Aerodrome catalog terminals.
./SharedData/19/latest/TerrainData/Terrain/World/Features/apt_shapes		
APT	Point features that define all available airports	The VR-Forces simulation engine uses this file to identify any airport to be controlled.
APTLightBeacon	Point features that define various beacons	Referenced from the Aerodrome catalog light_beacons.

Table 27. Shapefiles for procedural airports (continued)

Filename	Details	Notes
APTLinearFeatureLTC	Linear features that define various taxiway markings	Referenced from the Aerodrome catalog linear_features.
APTWindsock	Point feature that defines location of windsocks	Used by the custom Feature to place associated models.
Helipad	Point feature that defines location of helipads	Not currently used.
HelipadPolygon	Area feature that defines outline of helipads	Not currently used.
Pavement	Area feature that defines outline of paved surfaces	Referenced from the Aerodrome catalog pavement.
RunwayAccessLights	Point feature for the locations and contexts of runway access lights.	Referenced from the Aerodrome catalog runway_access_lights.
RunwayLine	Linear feature that defines the runways	Not currently used.
RunwayPolygon	Area feature that defines the runways	Referenced from the Aerodrome catalog runways and used to define the FeatureModel for the VR-Forces simulation engine.
RunwayThreshold	Point feature that defines various aspects of runways including the approach lighting	Referenced from the Aerodrome catalog runway_thresholds.
StartupLocation	Point feature that defines named starting locations	Referenced from the Aerodrome catalog startup_locations.
Stopway	Area feature that defines stopways	Referenced from the Aerodrome catalog stopways.

Table 27. Shapefiles for procedural airports (continued)

Filename	Details	Notes
TaxiwaySign	Point feature that defines the locations and contents for taxiway signs	Referenced from the Aerodrome catalog taxiway_signs.
VASI_PAPI_WIGWAG	Point features that define locations for the PAPI, VASI, and Wig-Wag lights	Referenced from the Aerodrome catalog light_indicators.
WaterRunwayPolygon	Area feature that defines water runways	Not currently used.
WaterRunwayThreshold	Point feature that defines various aspects of water runways	Not currently used.
./SharedData/19/latest/TerrainData/Terrain/World/Features/apt_shapes/NAVAIDS		
Airways	Linear features that defines airway segments	Not currently used.
DME	Point feature that defines locations of NAVAIDs	Used by the FeatureModel to place associated 3D models.
Fix	Point feature that defines all fixes	Not currently used.
GS	Point feature that defines glideslopes	Used by the FeatureModel to place associated 3D models.
ILS	Point feature that defines ILS components	Used by the FeatureModel to place associated 3D models.
Marker	Point feature that defines marker beacons	Not currently used.

Table 27. Shapefiles for procedural airports (continued)

Filename	Details	Notes
NDB	Point feature that defines NDBs (DB-records)	Not currently used.
VOR	Point feature that defines VORs (D-records, including VORTACs and VOR-DMEs)	Not currently used.

3.5.3 Editing Procedural Airport Data

All shapefiles are installed as part of the MAK ONE data packages. The files are editable, allowing you to change or add details as needed. In *MAK Earth (online)*, windsocks and NAVAIDs data come from VR-TheWorld, but you can update the configurations to use the local shapefiles instead if you need to edit those files. Note that the data used in the aerodrome catalog does not currently support WFSFeatures hosted on VR-TheWorld.

When you add any new features, you must update the attribute index for the APT_ICAO attribute for any file that contains an attribute for the airport ICAO code. Both the VR-Forces GUI and simulation engine use this for access to data. To do this, after you have made your edits, run:

```
ogrinfo -sql "create index on APT USING APT_ICAO" <updated-file>.shp
```

The ogrinfo tool is available as part of the QGIS tool, or you can use other GIS tools to perform the same operation. If you add any new airports, update the spatial index using QGIS or other GIS tools.

Editing Airports

To add new airports, update all relevant files to add details for the appropriate ICAO code. You must define an appropriate airport boundary to flatten the elevation, which allows for visible rendering of runways, pavement, and other objects.

This includes updating:

- *./SharedData/19/latest/TerrainData/Terrain/World/Features/APTBoundary_all.shp*
- And these files in *./SharedData/19/latest/TerrainData/Terrain/World/Features/apt_shapes*:
  - *APT.shp*
  - *APTBoundary.shp*

Editing Runways

To add new runways, update all relevant files and ensure that you add details for the appropriate ICAO code. Runway numbers need to be consistent across the multiple files.

This includes updating these files in *./SharedData/19/latest/TerrainData/Terrain /World/Features/apt_shapes*:

- *RunwayPolygon.shp* (and *RunwayLine.shp* if you want to be consistent, but the file is not currently used)
- *RunwayThreshold.shp*
- *Stopway.shp*
- *StartupLocation.shp*

Editing Other Airport Features

You may want to edit additional files to expand the visual aspects of any added airports and runways. However, these are not required for the VR-Forces sim engine.

This includes updating these files in *./SharedData/19/latest/TerrainData/Terrain /World/Features/apt_shapes*:

- *Pavement.shp* to add any paved areas outside the runways, including taxiways and parking areas.
- *APTLinearFeature.shp* to add any visual markings to the pavement areas and also to add any light beacon models.
- *TaxiwaySign.shp* to add any taxiway signs.

3.5.4 Editing Airport Lighting

A number of different files define the various airport lights. The following sections describe the shapefiles and attributes for each. In general, all lights require an associated airport ICAO code and associated runway number.

Approach Lights

For procedural airports, the various approach lights are defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/ aerodrome_catalog.xml* as part of the runway_thresholds features using RunwayThreshold.

Associated with airport ICAO code (apt_icao) and runway (rwy_num), the specific approach light type (approach_l) is defined. Values can be:

- None
- ALSF-I
- ALSF-II
- Calvert
- Calvert ISL Cat II and III
- MALS
- MALSF
- MALSR
- ODALS
- RAIL
- SALS

- SSALF
- SSALR

Runway Centerline Lights

For procedural airports, centerline lighting is defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as part of the runways features using RunwayPolygon.

For each runway, if the centerline attribute is 1 (true), then the centerline lighting is calculated from the runway polygon and rendered.

Displaced Threshold Lights

For procedural airports, threshold lighting is defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as part of the runway_thresholds features using RunwayThreshold.

Associated with airport ICAO code (apt_icao) and runway (rwy_num), each threshold feature has an attribute to determine whether the threshold is displaced or not (is_displac). If the value is 1, then the threshold is displaced. The threshold has an associated runway with length and heading.

Edge Lights

For procedural airports, edge lighting is defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as part of the runways features using RunwayPolygon.

Associated with airport ICAO code (apt_icao) and runway number (rwy_num1 and rwy_num2), the specific edge light type (edge_light) is defined. Values can be:

- None
- HIRL
- LIRL
- MIRL

Runway End Identifier Lighting (REIL)

For procedural airports, Runway End Identifier lighting is defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as part of the runway_thresholds features using RunwayThreshold.

Associated with airport ICAO code (apt_icao) and runway (rwy_num), the Runway End Identifier lighting is determined by using the REIL attribute. Values include:

- None
- Omni-directional
- Unidirectional

End Lights

For procedural airports, end lighting is defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as part of the runways features using RunwayPolygon.

The ends of runways are calculated and marked with runway end lighting.

Runway Guard (Wig-Wag), PAPI, and VASI Lights

For procedural airports, Wig-Wag, PAPI, and VASI Lights are defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as light_indicators using VASI_PAPI_WIGWAG.

Associated with airport ICAO code (apt_icao), there is an attribute (type) that defines the specific light type. Other attributes define specifics about the light. Values for the type can be:

- PAPI Left
- PAPI Right
- Tri-colour VASI
- VASI

Note: Space Shuttle PAPI (deprecated) still exists in the data but should not be used.

Threshold Lights

For procedural airports, threshold lighting is defined in the *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as part of the runway_thresholds features using RunwayThreshold.

Associated with airport ICAO code (apt_icao) and runway (rwy_num), each threshold feature has an attribute to determine whether the threshold is displaced or not (is_displac). If the value is 0, then the threshold is not displaced. The threshold has an associated runway with length and heading.

Touchdown Zone Lightbar

For procedural airports, Touchdown Zone (TDZ) lighting is defined in the *./SharedData/19/latest/TerrainData/Terrain/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml* as part of the runway_thresholds features using RunwayThreshold.

Associated with airport ICAO code (apt_icao) and runway (rwy_num), each threshold feature has an attribute to determine whether the runway has TDZ lighting (touchdown_). If the value is 1, then the runway has TDZ lighting. There may be multiple entries for the runway if the runway has a displaced threshold. One entry would define the TDZ lighting and the other would define the corresponding displaced threshold lighting similar to the TDZ lighting.

Notes on Taxiway Lights

Originally, the X-Plane-derived shapefiles did not have information for placing taxiway lights and, as a result, the functionality to automatically place lights from the Linear Features could not be implemented. Instead, these must be placed using newly created

shapefiles and specific attributes to place the taxiway lights.

See section "3.5.7 Adding Custom Airfield Lighting" on the facing page.

Note that the X-Plane data processing has been updated and the shapefile does now contain an X-Plane Line Type Code (LTC) attribute, which could eventually be used to render different line styles as well as embedded lights including those for Taxiway center line and edge lights. However, in the 11.53 data, very few airports actually have Taxiway center line and edge lights information defined.

3.5.5 Airport Terminals

The procedural airport catalog includes a terminals definition that uses *./SharedData/19/latest/TerrainData/Terrain/World/Features/aeroway_terminal_v4.shp* for the terminal building footprints associated with the respective airport ICAO codes. The source of the data was an extract from OpenStreetMap (OSM) for area features with the tag `aeroway=terminal`.

If you are using only the procedural airport catalog, this is a good way to render airport terminals. However, if you are also rendering OSM buildings, you may have duplicate buildings. In *MAK Earth (online)*, because OSM buildings and other inset airports are used, *./SharedData/19/latest/TerrainData/TerrainConfiguration/airports.worldwide.aerodrome.xml* is referenced with `build_terminals="false"` to prevent procedural airport terminals from rendering.

Depending on how you choose to render the terminals, you can edit the *aeroway_terminal_v4.shp* or update your other building data.

3.5.6 Other Procedural Airport Data

The primary configuration for procedural airports is done using the catalog defined in *./SharedData/19/latest/TerrainData/TerrainConfiguration/osgEarthCatalogs/aerodrome_catalog.xml*, referenced from *./SharedData/19/latest/TerrainData/TerrainConfiguration/airports.worldwide.aerodrome.xml*. In addition to the catalog, there are several other features used to add windsocks, add NAVAID models, and define FeatureModels used by the VR-Forces simulation engine. These are defined in *airports.worldwide.features.xml*.

Windsocks

Windsocks are dynamic 3D models that react to the wind speed and direction of the simulated weather. They are defined as feature_custom models. The model uses a point feature, currently hosted on VR-TheWorld.

If you need to change or add windsocks, you can use *./SharedData/19/latest/TerrainData/Terrain/World/Features/apt_shapes/APTWindsock.shp* to make the changes. You must either update the configuration to use your local shapefile as a OGRFeatures or make a new VR-TheWorld layer on your own server with the updated data and then modify the layer typename in the configuration file.

See "15.11. Adding Wind-based Controls to Models" on page 367 to learn about windsock models.

NAVAIDs

The procedural airport configuration uses several features to define 3D models for various NAVAIDs. These are defined in *./SharedData/19/latest/TerrainData/TerrainConfiguration/aiports.navaids.xml*, referenced by *airports.worldwide.features.xml* in the same folder. These include models for VORTAC, TACAN, VOR, DME, Glide Slope Indicator, and ILS.

Model definitions are in files located in *./SharedData/19/latest/TerrainData/Terrain/World/Features/apt_shapes/NAVAIDS*.

- VORTAC, TACAN, VOR, and DME positions are defined in *dme.shp*.
- Glide Slope Indicator positions and heading are defined in *gs.shp*.
- ILS positions and heading are defined in *ils.shp*.

If you need to change or add NAVAIDs, you can use the respective shapefiles to make the changes. You must either update the configuration to use your local shapefile as a OGRFeatures or make a new VR-TheWorld layer on your own server with the updated data and then modify the layer typename in the configuration file.

VR-Forces Simulation Engine Data

There are several FeatureModel objects defined specifically for the simulation engine and are used to support the VR-Forces aerodrome and entity tasks. These include:

- sim_boundary (*./SharedData/19/latest/TerrainData/Terrain/World/Features/APTBoundary_all.shp*)
- And files in *./SharedData/19/latest/TerrainData/Terrain/World/Features/apt_shapes*:
  - sim_runways (*RunwayPolygon.shp*)
  - sim_airports (*APT.shp*)

These use the same shapefiles as described previously for the visuals, therefore any changes made will be reflected in VR-Vantage, VR-Engage, and the VR-Forces GUI and simulation engine.

3.5.7 Adding Custom Airfield Lighting

The X-Plane data does not currently contain any data for Taxiway (Aerodrome) Lights or Runway Access Lights. To add or modify any of the existing data, you must configure the files as described below.

Runways

All supported light group types for runways are defined in *./appData/importConfig/runway_lightgroup_by_name_map.csv*.

Table 28. Runway light group types

Light Group Name	Description
TDZ	Touchdown Zone Lightbar
CNTR	Centerline Light
Edge	Edge Light
END	End Light
THR	Threshold Light
DTHR	Displaced Threshold Light
REIL	REIL (runway end identifier lights)
REIL	Unidirectional REIL
VGSI	PAPI (precision approach path indicator) left
VGSI	PAPI right
VGSI	VASI (visual approach slope indicator)
RGL	Wig-Wag lights
DTG	Distance To Go Lights
APP	Approach Light ALSF-I (approach lighting system with sequenced flashing lights)
APP	Approach Light ALSF-II
APP	Approach Light Calvert
APP	Approach Light Calvert ISL Cat II and III
APP	Approach Light MALS (medium-intensity approach lighting system)
APP	Approach Light MALS-SSALS
APP	Approach Light MALSF
APP	Approach Light MALSF-SSALF
APP	Approach Light MALSR
APP	Approach Light ODALS (omnidirectional approach lighting system)
APP	Approach Light RAIL
APP	Approach Light SALS
APP	Approach Light SSALF
APP	Approach Light SSALR

Aerodromes

Support light group types for aerodromes are defined by *./appData/importConfig/aerodrome_lightgroup_by_name_map.csv*. The supported types are listed in the table.

Table 29. Taxiway (aerodrome) light group types

Light Group Name	Description
AL	Apron Light
AFL	Apron Flood Light
CBL	Clearance Bar Light
OL	Obstruction Light
TCL	Taxiway Center Light
TEL	Taxiway Edge Light
TGL	Taxiway Guard Light
TIML	Taxiway Identifier MarkerBoard Light
UISL	Undercarriage Inspection System Light
TLONL	Taxiway Lead On Light
TLOFL	Taxiway Lead Off Light
TREL	Taxiway Rapid Exit Light
SBL	Stop Bar Light

Model Definitions

Model definitions that map to light group types for aerodromes are defined in *./appData/importConfig/runway_light_modeldef_by_name_map.csv*.

The model definitions reference 3D models with light points, light sources, and geometry. The geometry can also have emissive textures.

Light Point Configuration

For light point configuration, use the CDB Lightpoint Palette Name in the OpenFlight File. In an FBX file, using a Standard Omni or Spot Light source, you can add the comment in the Object properties dialog `@dis lightpoint` plus the CDB Lightpoint Palette Name, which can also be found in Hierarchy in the *./appData/importConfig/Lights_DefaultValues_3_2.csv* file.

For example:

```
@dis \Light\Cultural\Airport_Lighting\Taxiway\Edge
@dis LightCulturalAirport_LightingTaxiwayEdge
```

As shown above, you can write it with or without backslashes.

For more information, see "15.5. Configuring Light Points" on page 355.

If you want to control the intensity for a light, not simply configure it for either on or off, you must specify a Light Group Name in a comment.

For emissive textures and light source intensities that you need to control, you must define a light point with the same CDB Lightpoint Palette Name as the emissive texture geometry or dynamic light.

For example:

```
@dis lightpoint \Light\Cultural\Airport_Lighting\Taxiway\Edge
```

Example of emissive texture geometry:

```
@dis emissive \Light\Cultural\Airport_Lighting\Taxiway\Edge
```

Example of light source:

```
@dis dynamiclight  \Light\Cultural\Airport_Lighting\Taxiway\Edge
```

Table 30. Model definitions for light group types

Light Group type	Model Definition
Touchdown Zone Lightbar	LightsRunwayTouchownZoneLight
Centerline Light	LightsRunwayCenterlineLight
Edge Light	LightsRunwayEdgeLight
End Light	LightsRunwayEndLight
Threshold Light	LightsRunwayThresholdLight
Displaced Threshold Light	LightsRunwayDisplacedThresholdLight
REIL	LightsRunwayEndIdentifierLightUnidirectional
Unidirectional REIL	LightsRunwayEndIdentifierLightUnidirectional
Approach Light ALSF-I	LightsRunwayApproachLightingALSF-I
Approach Light ALSF-II	LightsRunwayApproachLightingALSF-II
Approach Light Calvert	LightsRunwayApproachLightingCalvert
Approach Light Cavert ISL Cat II and III	LightsRunwayApproachLightingCavert ISL Cat II and III
Approach Light MALS	LightsRunwayApproachLightingMALS
Approach Light MALS-SSALS	LightsRunwayApproachLightingMALS-SSALS
Approach Light MALSF	LightsRunwayApproachLightingMALSF
Approach Light MALSF-SSALSF	LightsRunwayApproachLightingMALSF-SSALSF
Approach Light MALSR	LightsRunwayApproachLightingMALSR
Approach Light ODALS	LightsRunwayApproachLightingODALS
Approach Light RAIL	LightsRunwayApproachLightingRAIL
Approach Light SALS	LightsRunwayApproachLightingSALS

Table 30. Model definitions for light group types (continued)

Light Group type	Model Definition
Approach Light SALSF	LightsRunwayApproachLightingSALSF
Approach Light SSALR	LightsRunwayApproachLightingSSALR
PAPI Left	LightPAPILeft
PAPI Right	LightPAPIRight
VASI	LightVASI
Wig-Wag lights	LightWigWag
Apron Flood Light	LightsFloodLightPole
Apron Light	LightsTaxiwayLight
Clearance Bar Light	LightsTaxiwayClearanceLight
Obstruction Light	GetModelFromFile
Taxiway Center Light	LightsTaxiwayCenterlineLightAligned
Taxiway Edge Light	LightsTaxiwayLight
Taxiway Guard Light	LightsLightPoint
Taxiway Identifier MarkerBoard Light	LightsRunwayEndLight
Undercarriage Inspection System Light	GetModelFromFile
Taxiway Lead On Light	LightsTaxiwayCenterlineLightAligned
Taxiway Lead Off Light	LightsTaxiwayCenterlineLightAligned
Taxiway Rapid Exit Light	LightsTaxiwayCenterlineLightAligned
Stop Bar Light	LightsTaxiwayCenterlineLightAligned

You can add lights that are not in the X-Plane data to procedural airports as points in the respective shapefiles, one point per light.

Each point feature must have the associated Attributes:

- type - Light Group Type
- apt_icao - Airport ICAO code
- rotation – orientation of light model
- model – 3D model override if configured with GetModelFromFile
- modelpath – 3D modelpath override if configured with GetModelFromFile

In the Aerodrome Catalog:

```
<taxiway_lights>
  <OGRFeatures name="taxiway_lights" build_spatial_index="true">
    <url>../../../TerrainData/Terrain/World/Features
        /apt_shapes/TaxiwayLights.shp</url>
  </OGRFeatures>
</taxiway_lights>
```

```
  <runway_access_lights>
    <OGRFeatures name="runway_access_lights" build_spatial_
index="true">
      <url>../../../TerrainData/Terrain/World/Features
          /apt_shapes/RunwayAccessLights.shp</url>
    </OGRFeatures>
  </runway_access_lights>
```

Filename	Details	Notes
Taxiway Lights	Point feature that the locations and contents for taxiway lights. Apron Flood Light, Apron Light, Clearance Bar Light, Obstruction Light, Taxiway Centre Light, Taxiway Edge Light, Undercarriage Inspection System Light.	Referenced from the Aerodrome catalog taxiway_lights
Runway Access Lights	Point feature that the locations and contents for Runway Access lights, including: Taxiway Lead On Light, Taxiway Lead Off Light, Taxiway Rapid Exit Light, Stop Bar Light.	Referenced from the Aerodrome catalog runway_access_lights

3.5.8 Airfield Lighting for Inset Airports and Runways

Aerodrome configuration can substitute runways or airports with custom OpenFlight inset models. The insets can work in conjunction with the procedural aerodromes and can be mixed and matched, allowing part of an aerodrome to be an inset and part to be procedural.

To control lights in an OpenFlight airport runway model, light points must be organized into discrete light groups. These light groups are identified by a string in the Comment tab of a Lightpoint System node or Group node.

The string must consists of `@dis light-group` then the ICAO airport code, the runway identifier and the lightgroup type. The resulting format is `@dis light-group` *`ICAO`*`:`*`Runway Identifier`*`:`*`Lightgroup`*, for example:

```
@dis light-group KBOS:22L:APP
```

To display light points properly, they should be configured to CDB Lightpoint Palette Names. MAK ONE applications map CDB Lightpoint Palette names to column D, Hierarchy, in *./appData/importConfig/Lights_DefaultValues_3_2.csv*.

For more information, see "15.5. Configuring Light Points" on page 355.

3.5.9 Runway Offset to Avoid Z-Fighting

You can set a runway offset elevation to avoid Z-fighting (rendering artifacts resulting from co-planar polygons). You simply need to specify a non-zero `offset`. In your *airports.worldwide.features.xml* file, set something similar to:

```
<FeatureElevation name="elevation-worldwide-airports"
   min_level="10"
   max_data_level="15"
   vrfsim:max_level="13"
   vrfsim:worldwide="true"
   enabled="true"
   offset="-0.005"
   attr="ELEVATION">
```

The optional keyword: `vrfsim:worldwide="true"` bypasses the extents for airports in a terrain. VR-Forces assumes that the extent is the world. This improves performance when loading terrains.

4. Managing Terrains Using the MAK ONE Application GUI

This chapter explains how to manage aspects of the terrain using features in the MAK ONE application GUI, including 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.

4.1. Introduction to the Terrain Editor 176
4.1.1 Opening the Terrain Editor 176
4.2. Adding Data to a Terrain 177
4.3. Editing Terrain Properties in the Terrain Editor 177
4.3.1 Enabling and Disabling Layers 177
4.3.2 Editing the Properties of a Layer 178
4.3.3 Setting Properties for Terrain Patches 178
4.4. Testing Terrain Components Using the Terrain Editor 180
4.4.1 Adding Elevation Data to a Terrain 180
4.4.2 Adding Images to a Terrain 182
4.4.3 Feature Layers 183
4.4.4 Saving a Terrain 185
4.5. Adding Dynamic Oceans 185
4.5.1 Adding a Dynamic Ocean Layer 185
4.5.2 Adding a Dynamic Ocean Layer Directly 188
4.5.3 Setting the Ocean LOD Elevation 189
4.5.4 Configuring the Ocean Height Map 189
4.6. Specifying Simulation Ocean Settings 192
4.7. Adding Props to a Terrain 193
4.7.1 Extracting Props from a Terrain Patch 193
4.7.2 Adding Props from a Feature Layer 195
4.7.3 Viewing a List of Props 197
4.7.4 Selecting Props 197
4.7.5 Changing a Prop's Type 197

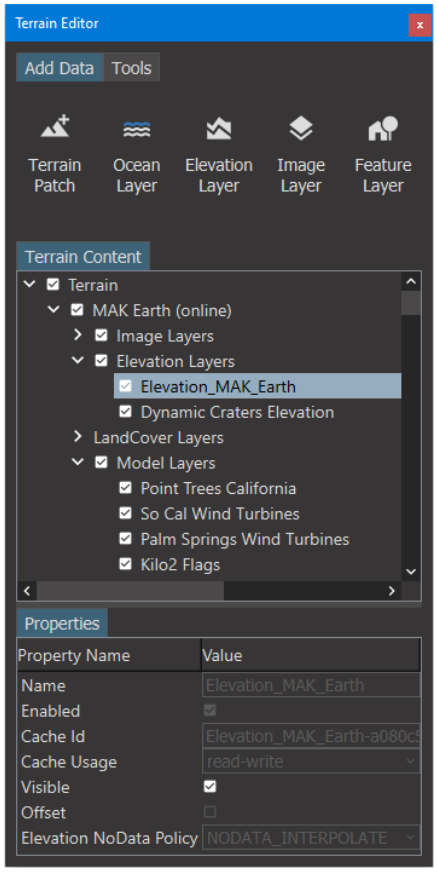
4.7.6 Changing a Prop's Position or Orientation 198
4.7.7 Changing a Prop's Model Definition 198
4.7.8 Exporting Props and SpeedTrees to a Shapefile 198
4.7.9 Importing Props and SpeedTrees from Shapefiles 199

4.1. Introduction to the Terrain Editor

VR-Vantage, VR-Forces, and VR-Engage let you define and load terrains by creating earth files that specify all of the terrain data to be loaded. Approaches to creating terrains using earth files are described in "Chapter 2. Composing a Terrain from Source Data" on page 69.

The Terrain Editor (Figure 18) is a graphical user interface that allows you to add terrain patches and 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.

Figure 18. 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

You can use the Terrain Editor to 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 2. Composing a Terrain from Source Data" on page 69 and "3.1. Connecting to Terrain Servers" on page 154.

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 described in "4.4.4 Saving a Terrain" on page 185.

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 are not saved to the earth file. If you make a change that you want to keep, you must edit the earth file. The advantage of using the Terrain Editor is that you can test many changes in real time instead of editing the earth file and then reloading 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 of a layer. This provides a quick way for you to examine the visual impact of the 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 edit a terrain that uses an earth file created in a text editor, the layer can have many more properties. You can use the Terrain Editor to change their values and see the effect in real time.

Changed properties are only valid during the current application session. You can use the information to update the terrain file.

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

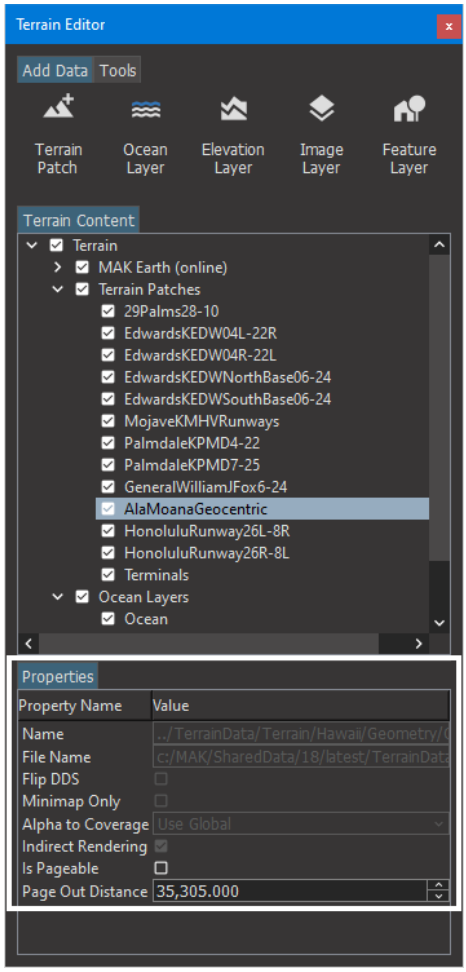
➤ **To edit a property's value:**

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.

4.3.3 Setting Properties for Terrain Patches

Using the Terrain Editor, you can set properties for the terrain content. For terrain patches, you can specify whether the patch is pageable and, if so, the page out distance.

Figure 19. Terrain properties for Terrain Patch



4.4. Testing Terrain Components Using the Terrain Editor

You can test terrain components by adding elevation, imagery, and feature data using the Terrain Editor Panel. You cannot use the panel to create or modify an earth file. It simply allows you to see how those individual files will appear in MAK ONE applications.

The overall process for composing a terrain through the GUI is:

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 "4.4.1 Adding Elevation Data to a Terrain" below.
3. Optionally, add imagery, as described in "4.4.2 Adding Images to a Terrain" on page 182.
4. Optionally, add feature data, as described in "4.4.3 Feature Layers" on page 183.
5. Optionally, add a dynamic ocean layer, as described in "4.5.1 Adding a Dynamic Ocean Layer" on page 185.
6. Optionally, add props or extract props, as described in "4.7.2 Adding Props from a Feature Layer" on page 195 and "4.7.1 Extracting Props from a Terrain Patch" on page 193.
7. Save the terrain, as described in "4.4.4 Saving a Terrain" on page 185.

For a list of supported terrain and feature formats, see "Supported File Formats" on page 30.

- For details about using the Terrain Editor, see "Chapter 4. Managing Terrains Using the MAK ONE Application GUI" on page 175.
- You can also see "Chapter 9. Terrain Tutorials" on page 271 for more guidelines on creating terrains.

4.4.1 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.

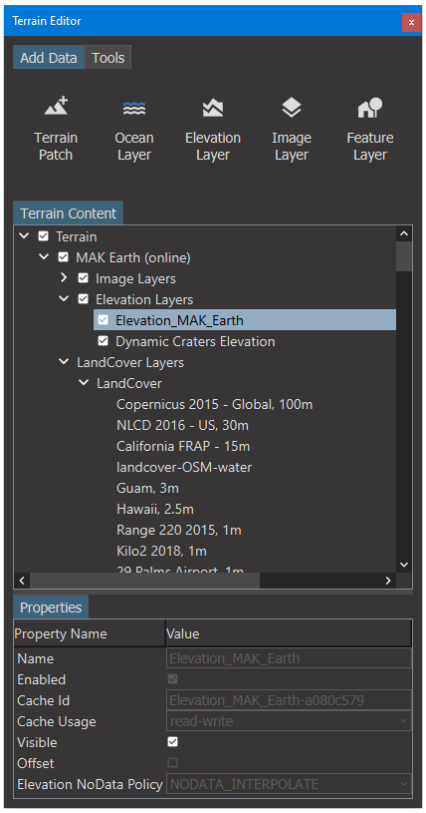
You can use this procedure to add a single earth file and other terrain files. Some are found in the *./SharedData/19/latest/TerrainData/TerrainConfiguration* directory.

- 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 load only one earth file 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:**

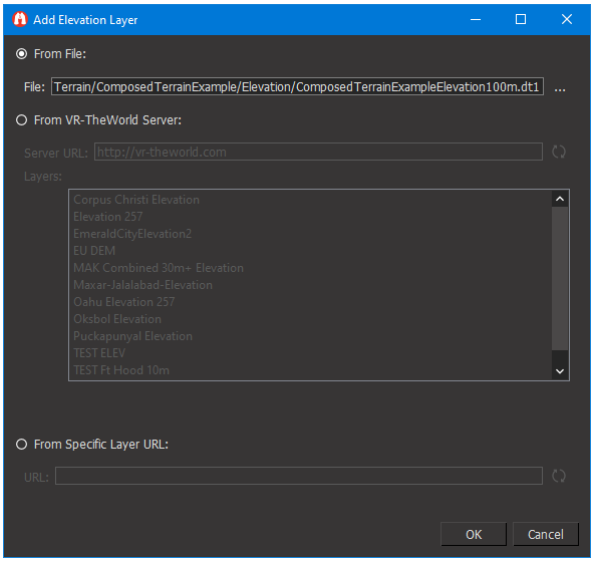
1. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.
2. Select the Add Data tab (Figure 20).

Figure 20. Terrain Editor Panel



3. Click Elevation Layer. The Add Elevation Layer dialog box opens.
4. Select the file, the layer from the VR-TheWorld, or the URL (Figure 21).

Figure 21. Add Elevation Layer dialog box



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

4.4.2 Adding Images to a Terrain

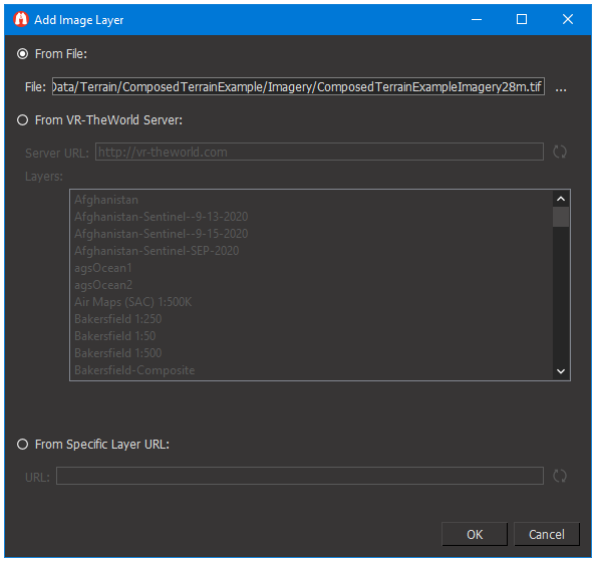
Depending on your terrain source, you may have terrain polygonal data, but no texture data or feature data to provide a realistic 3D display. You can drape images (raster maps) over the terrain to provide a more pleasing display.

> 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 22).

Figure 22. 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.

4.4.3 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 GUI) 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 that file.

> ⚠ In VR-Forces, loading large feature layers into the GUI 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, the application does not visualize the feature data, but the simulation engine handles 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, see "Chapter 7. Configuring Feature Data in VR-Forces and VR-Engage" on page 229.

The preferred method for adding feature data in the sim engine is via shapefiles. Loading shapefiles in the sim engine, 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, see "7.6. Extracting Feature Data from GDB to Shapefiles" on page 233.

- 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, see "4.7. Adding Props to a Terrain" on page 193.
- 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 when you load the earth file.
- MAK ONE applications support S-57 data. The recommended way to load S-57 data is to convert it to shapefiles and then load those 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, *Test-Massachusetts S-57 (online).mtf*.

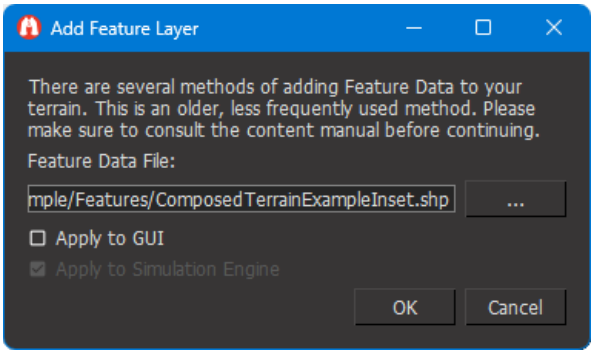
Adding a Feature Layer

You can use the GUI to add a feature layer, but other methods are preferred.

➤ **To add a feature layer:**

1. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.
2. Select the Add Data tab (Figure 20 on page 181).
3. Click Feature Layer. The Add Feature Layer dialog box opens (Figure 23).

Figure 23. Add Feature Layer



4. Enter 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 GUI, or both, select Apply to Front End. Clear Apply to Front End to add a feature layer only in the sim engine. (Apply to Simulation Engine is always selected and is not editable.)
6. Click OK.
7. Save the terrain.

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.

4.4.4 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. Terrains are saved with the extension *.mtf*.

When accelerated file loading is enabled, the data saves in a format that can be loaded more quickly than from its native format. (For details about accelerated file loading, see "8.1. Configuring File Caching" on page 238.)

> 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 as an MTF file:**

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.

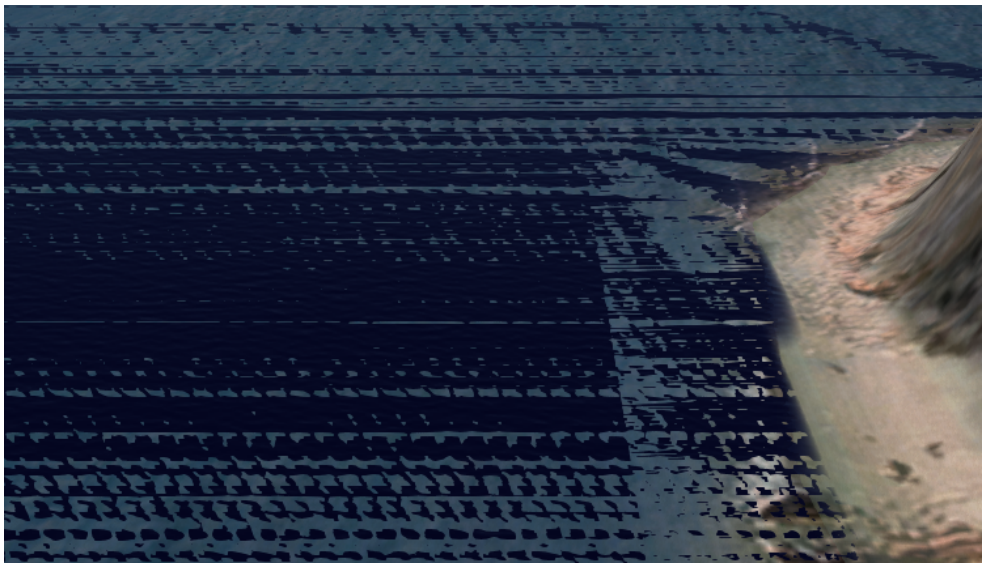
4.5. Adding Dynamic Oceans

4.5.1 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 24). This problem occurs because the ocean layer and the texture are both at the same height (0, or sea level).

Figure 24. 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, unless your elevation data includes the appropriate bathymetry, you can experience Z-fighting when the ocean layer is at the same elevation as the terrain texture. You may need to raise the ocean height to actually see the ocean effects.

Figure 25 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 25. Ocean height

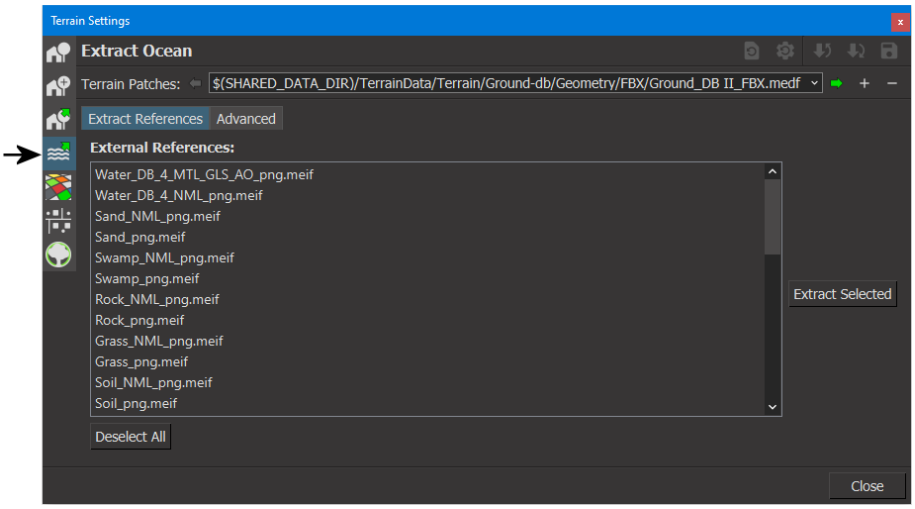
Extracting Water Textures and Adding an Ocean Layer

You cannot extract water textures when indirect rendering is enabled. You can temporarily disable indirect rendering as described in "8.5.3 Disabling Indirect Rendering for Terrain" on page 249.

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

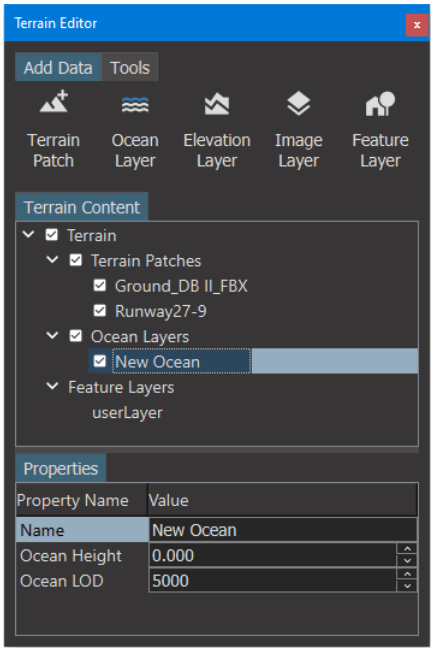
1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Extract Ocean page (Figure 26).

Figure 26. 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 27).

Figure 27. Ocean layer properties



7. Adjust the Ocean Height such 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, see "4.5.3 Setting the Ocean LOD Elevation" on the facing page.
9. Save the terrain.

4.5.2 Adding a Dynamic Ocean Layer Directly

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

➤ **To add a dynamic ocean layer:**

1. Choose **View > Terrain Editor Panel**. The Terrain Editor opens (Figure 27 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 such 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, see "4.5.3 Setting the Ocean LOD Elevation" on the facing page.

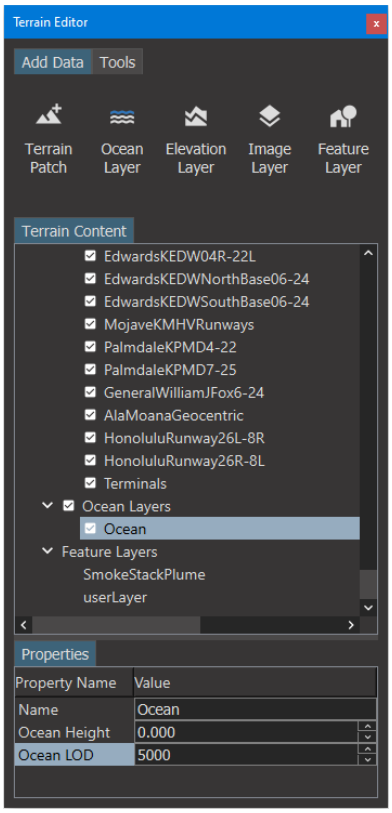
4.5.3 Setting the Ocean LOD Elevation

Enabling dynamic ocean affects performance. Since the effects are hard to see once the observer is high above the ocean, 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 28).

Figure 28. Ocean layer properties



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

4.5.4 Configuring the Ocean Height Map

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

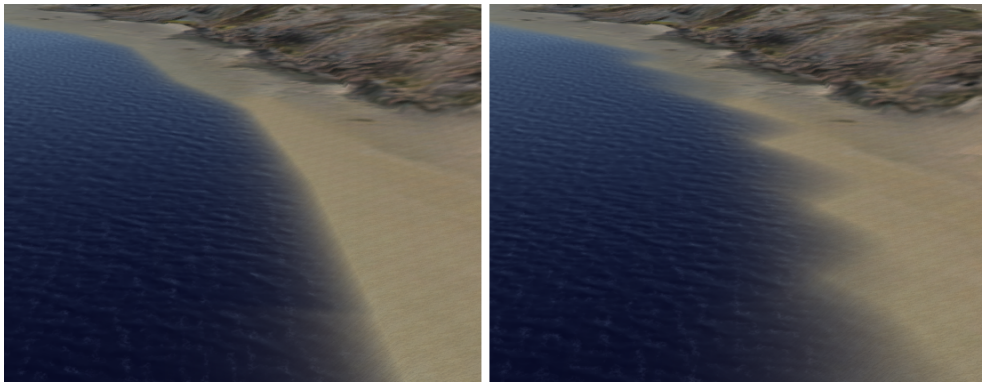
Height map resolution is specified as meters per pixel. The height map texture size is specified in pixels. Therefore, these two settings determine the length of one side of the

(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 40,960 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 should 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 29 shows a coastline at different levels of resolution.

Figure 29. Ocean height map resolution – 1.0 and 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 either "Configuring Marine Conditions" in the *VR-Vantage User Guide* or "Configuring Marine Conditions" in the *VR-Forces User Guide*.

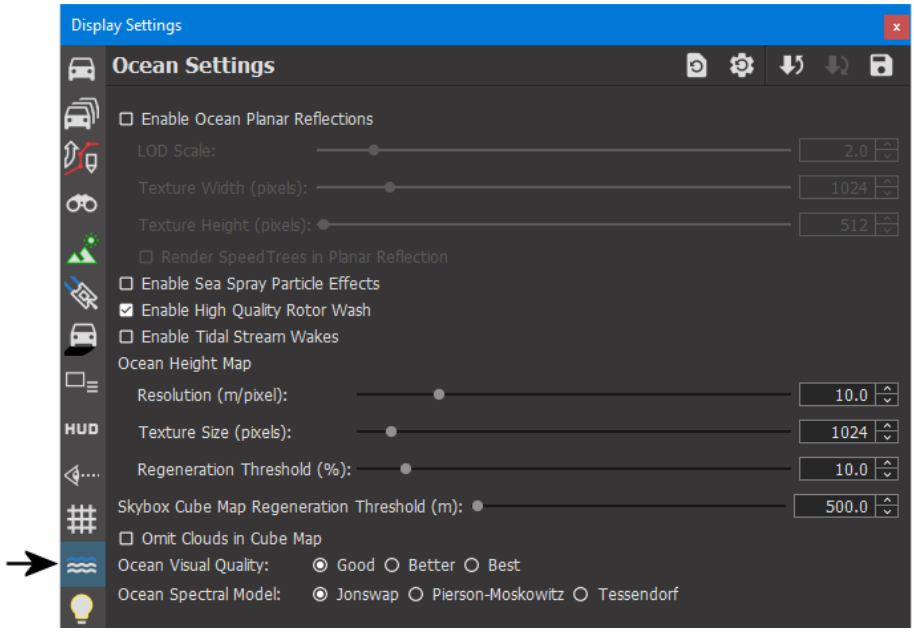
Setting the Ocean Height Map Resolution

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 30).

Figure 30. Ocean Settings



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

Setting the Ocean Height Map Texture Size

The ocean height map texture size is specified in pixels.

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

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 30 above).
3. Adjust the Ocean Height Map Texture Size slider or enter a value in the box.

Setting the Ocean Height Map Regeneration Threshold

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

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

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 30 above).
3. Adjust the Ocean Height Map Regeneration Threshold slider, or enter a value in the box.

4.6. Specifying Simulation Ocean Settings

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

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

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

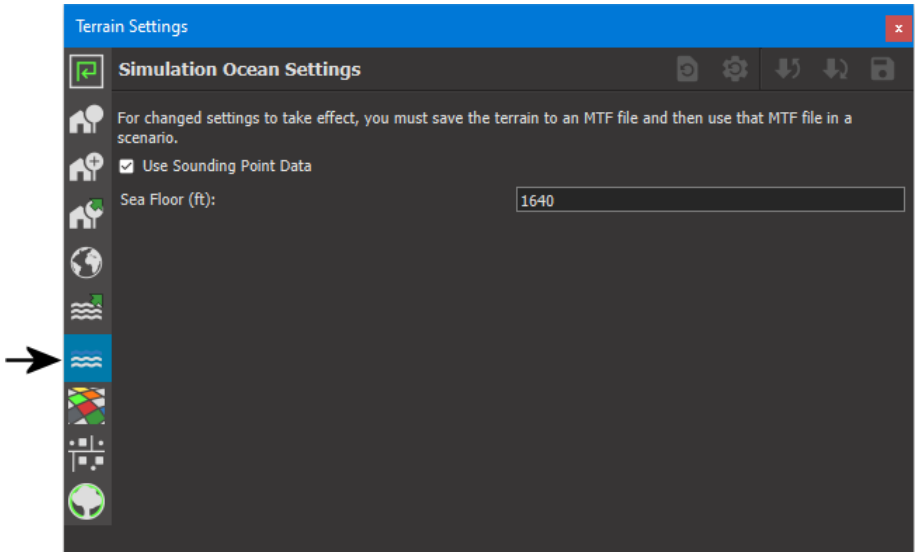
You can also specify that VR-Forces use a fixed depth for any area that does not have underwater polygons or feature data.

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

➤ **To specify ocean depth:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Simulation Ocean Settings page (Figure 31).

Figure 31. Simulation Ocean Settings page (VR-Forces)



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, enter a value in the Sea Floor text box.
5. Save the terrain.
6. Reload the scenario.

4.7. Adding Props to a Terrain

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

> In VR-Forces, you can create buildings and cultural feature objects from the Create Object Palette that, in the 3D Stealth view, look the same as props with similar names. Cultural features are not the same as props, however. 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.

4.7.1 Extracting Props from a Terrain Patch

When you extract props from a terrain patch, you must tell the MAK ONE application where to find prop data, what type of prop to use, and what model definitions to map the props to:

- 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*.

  If you select the Extract as Speed Tree check box, the application adds an extraction operation to the MTF file. This extraction operation replaces external references with SpeedTree models. Using this option avoids the need for each tree to be specified in the MTF file.
- 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.

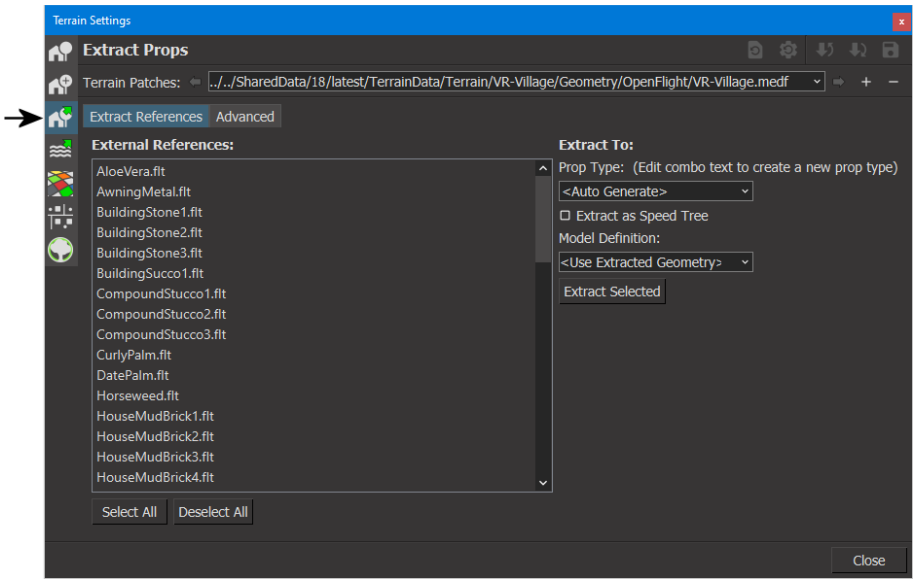
For an example of how to extract props from a terrain patch, see "9.2. Extracting Props from a Terrain" on page 278.

> You cannot extract props when indirect rendering is enabled. You can temporarily disable indirect rendering as described in "8.5.3 Disabling Indirect Rendering for Terrain" on page 249.

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

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Extract Props page (Figure 32).

Figure 32. 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, enter 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 enter 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 Extract References tab, click Extract Selected. If you are using the Advanced tab, click Extract. The props, if any, are extracted. You can view the list of props in the Props list on the Edit Existing Props page.

Extraction Operation Example

The following is an example of the extraction operation for SpeedTrees.

```
<myExtractionOperations class_id="10" tracking_level="0" version="0">
  <count>1</count>
  <item_version>1</item_version>
  <item class_id="11" tracking_level="0" version="1">
    <myExpression>^((CurlyPalm.flt))$</myExpression>
    <myRegExFlag>1</myRegExFlag>
    <myKey></myKey>
    <myNodeCount>88</myNodeCount>
  </item>
</myExtractionOperations>
```

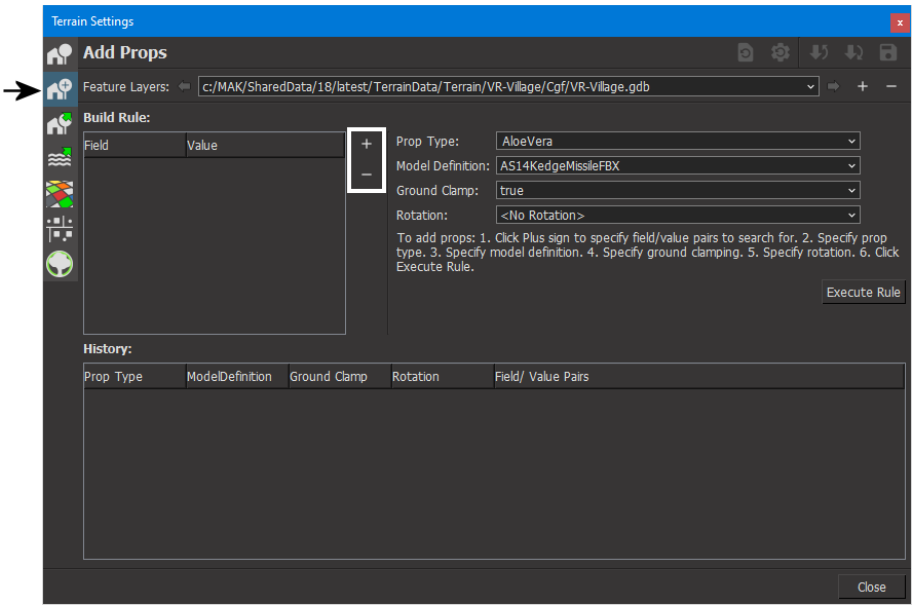
4.7.2 Adding Props from a Feature Layer

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 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:**

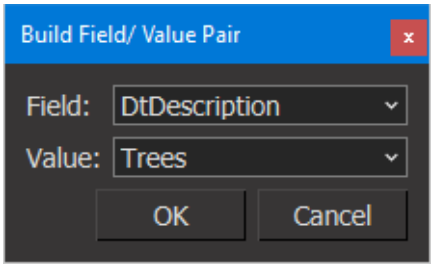
1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Add Props page (Figure 33).

Figure 33. 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 33). The Build Field/Value Pair dialog box opens (Figure 34).

Figure 34. Build Field/Value Pair dialog box

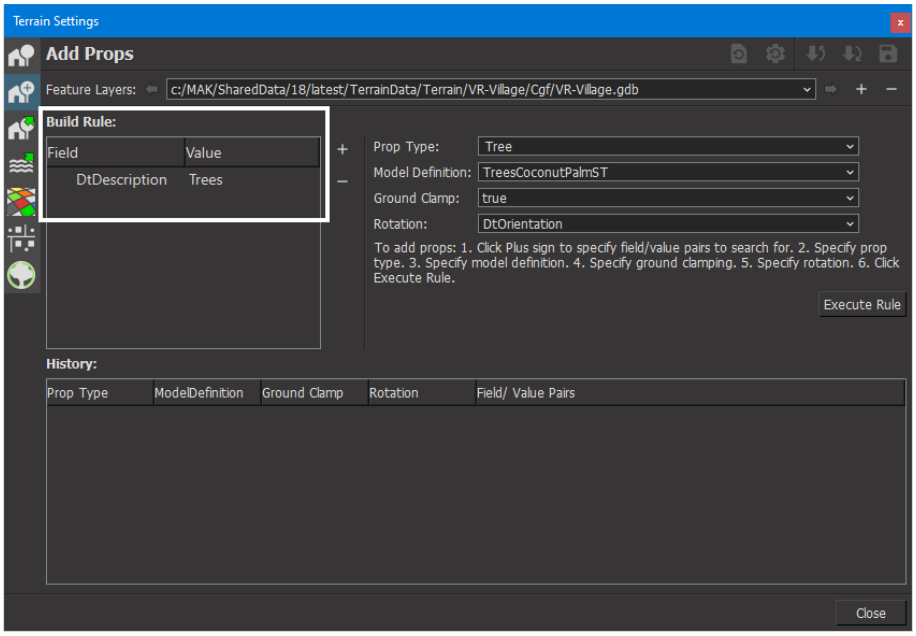


5. In the Field list, select a field. Fields are the column headings in the feature definition.
6. In the Value list, select a value. Values are one or more values from the definition table that are available for the column heading.

> 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 35).

Figure 35. 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.

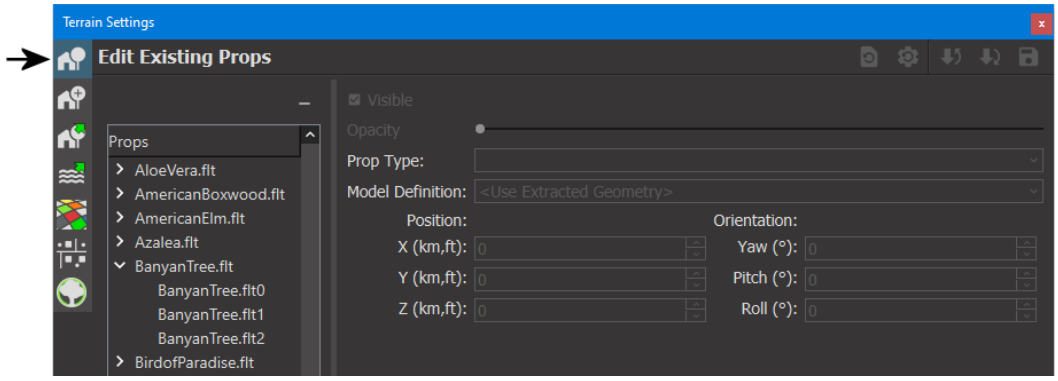
10. In the Model Definition list, select the model definition you want to use for this prop.
11. In the Ground Clamp box, specify whether to ground clamp this prop.
12. Optionally, set the rotation.
13. Click Execute Rule. The prop is extracted from the feature layer and is listed in the History window.

4.7.3 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 36).

Figure 36. Edit Existing Props page



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

4.7.4 Selecting Props

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

➤ To select multiple props, use **Ctrl**+click or **Shift**+click.

4.7.5 Changing a Prop's Type

Every prop has a type. Props are organized by type in 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 36 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 enter a new type name. The selected props are moved to the new prop type in the Props window.

4.7.6 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 36 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, select or enter new values in the X, Y, or Z boxes.
5. To change the orientation, select or enter new values in the Yaw, Pitch, or Roll boxes.

4.7.7 Changing a Prop's Model Definition

You can change the model definition used to render a prop. For information, see "12. Model Definitions" on page 303.

➤ **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 36 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.

4.7.8 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 current terrain. You can then add the props to a terrain using 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. Enter a name for the shapefile.
3. Click Save.

4.7.9 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.

5. Dynamic Terrain in MAK ONE Applications

This chapter describes the dynamic terrain feature in MAK ONE applications.

5.1. Introduction to Dynamic Terrain 202
5.2. Creating Dynamic Terrain Areas and Lines 202
5.3. Dynamic Terrain Switch Types 203
5.4. Craters, Ditches, Berms, and Tank Scrapes 203
5.4.1 Craters 203
5.4.2 Ditches, Berms, and Tank Scrapes 204
5.5. Destructible Buildings in Procedural Terrains 204
5.6. Manually Changing the Terrain 204
5.7. Resetting Dynamic Terrain Changes 205
5.8. Removing Dynamic Terrain Damage 205
5.9. Resetting All Changes in Dynamic Terrain 205

5.1. Introduction to Dynamic Terrain

VR-Forces and VR-Engage support 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, ditches, berms) in response to munition impact or the placement of dynamic terrain control objects.

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 and VR-Engage can automatically regenerate navigation data for the terrain. This allows simulation objects to take the damage into account as they move through the terrain. The priority of regeneration and the types of terrain change that trigger it are configured in *./appData/settings/vrfSim/navigationProfiles.mtl*. By default, navigation data is regenerated for damage, and damage_structure switch types. For information about navigation areas and navigation profiles, please see the *Generating Navigation Data* chapter in the *VR-Forces User Guide*. For information about switch types, see "5.3. Dynamic Terrain Switch Types" on the facing page.

- 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 can take 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 with the original navigation data for a brief period until navigation data updates.
- Dynamic terrain is supported on OpenFlight and FBX terrain patches, and Heightfields and Point Model Replace in osgEarth. MetaFlight, Terrapage, and CDB terrains, and also props do not support dynamic terrain.

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

5.2. Creating Dynamic Terrain Areas and Lines

If you disable the global dynamic terrain processor when you create a scenario, you can still enable dynamic terrain within dynamic terrain areas and lines. (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 User Guide*.)

Dynamic terrain areas and lines are terrain control objects that are available on the Create Object Palette. You create them in the same manner as any other areal or linear tactical graphic.

5.3. Dynamic Terrain Switch Types

Table 31 lists the dynamic terrain switches supported.

Table 31. 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.

5.4. Craters, Ditches, Berms, and Tank Scrapes

VR-Forces and VR-Engage can add craters, ditches, berms, and tank scrapes to terrain. This type of dynamic terrain only works on osgEarth terrains in parts of the terrain defined by `heightfield` elements. It does not work in most high fidelity inset areas. VR-Forces can, however, place a crater model on an airport's runway or taxiway and flatten the area on the airport. They are an added model, similar to a prop.

5.4.1 Craters

VR-Forces can add craters to the terrain in response to munition impact. The size of the craters can vary depending on the soiltype and munition type.

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 soiltype and munition type.

Crater models for runways and airports are in *./SharedData/19/latest/ModelData/Configuration/Definitions/Entity/Environment/Land/Craters*.

5.4.2 Ditches, Berms, and Tank Scrapes

VR-Forces users can add ditches and berms to the terrain using the Dynamic Terrain Ditch, Dynamic Terrain Berm, and Dynamic Terrain Tank Scrape terrain objects. The dynamic terrain objects are configured in the *./data/simulationModelSets/base/vrfSim/systems/other/dynamic-terrain.sysdef* file by the `linear-deformation` parameters in the `linear-deformation-list` block. Tank scrapes are directional, providing a place for tanks to take hull-down positions. The elevation deformation of tank scrapes is in *./SharedData/19/latest/ModelData/Environment/Land/TankScrape*.

5.5. Destructible Buildings in Procedural Terrains

For detonations of appropriate size near a vertex of a procedurally generated building, the building is replaced with a rubble model.

The detonation radius must intersect at least one vertex of a building in order to destroy it. For example, if there is a tall building with 4 vertices at the bottom of the building and 4 at the top but the explosion happens in middle of the building, it will not be destroyed.

This works for any osgEarth terrain using the *osm.features.xml* file and the debris models provided in the MAK ONE data package. You can find the *osm.features.xml* file in *./SharedData/19/latest/TerrainData/TerrainConfiguration* and the debris models in *./SharedData/19/latest/ModelData/Environment/Land/Debris*.

5.6. 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 objects with switches.

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

1. In VR-Forces, 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 where you can make changes.
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 included 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.

5.7. Resetting Dynamic Terrain Changes

If you have made changes to the terrain by hand, such as opening doors and windows, you can reset those changes. The Reset Changes feature also enables you to remove damage caused by a munition.

➤ **To reset dynamic terrain changes:**

1. In VR-Forces, 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 included in the list window.
4. For each option listed that you want to change, select Reset in the list.
5. Click OK.

5.8. Removing Dynamic Terrain Damage

You can remove terrain damage caused by munitions or manual changes to the terrain, including craters. Remove Damage does not affect changes to doors and windows.

➤ **To remove dynamic terrain damage:**

1. In VR-Forces, 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.

5.9. Resetting All Changes in Dynamic Terrain

In a dynamic terrain, you can reset all changes. That is, you can remove damage such as craters and ruined buildings, as well as changes like persistent tracks or doors and 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. In VR-Forces, choose **Simulation > Dynamic Terrain > Reset All Changes**.
2. Click OK.

6. Material Classification for SensorFX

SensorFX is a plug-in to MAK ONE applications. It provides physics-based sensor views for Long Wave Infrared (LWIR), Midwave Infrared (MWIR), and night vision goggles (NVG). In physics-based sensor views, the rendering of terrain and simulated objects takes into account the material of the object, air and material temperature, ambient light, and the characteristics of the sensor.

Physics-based sensor simulation starts with a materially classified scene. This chapter describes the concepts behind material classification for use with SensorFX.

MAK ONE does not support TerraPage for material classification.

6.1. Material Classification on Demand 208
6.1.1 Associating Styles and Features 212
6.2. Streaming Sensorized Imagery 213
6.3. Using SensorFX with Geocentric Terrains 214
6.4. Material System Concept 214
6.4.1 Thermal Phenomena Modeling 215
6.4.2 Common 1D Material Systems 219
6.4.3 Definitions of Assignment Descriptor 220
6.4.4 Boundary Conditions Thermal Equations 221
6.5. Material Classification of CDB Terrains 225
6.5.1 Configuring CDB Raster Material 225
6.5.2 Configuring CDB to Sensorized CDB 3D Models (Geotypical or Geospecific) 225
6.5.3 CDB Base Material Mapping to Sensor Material Classification 226

6.1. Material Classification on Demand

SensorFX material classification on demand (MCOD) provides automatic material classification using vector shape-files, and combines them with an intensity-modulated image to produce material encoded textures used in the real-time sensor rendering. The material encoded textures use the Eigen material technique, which provides excellent rendering capabilities with the ability to mimic an unlimited number of material systems through the combination of multiple eigen materials.

Eigen materials can be thought of as a basic material group that can be combined to form specific material systems. These textures are in a continuous parameter space that is not dependent on time-of-day and other environmental conditions and can be Mipped/interpolated and compressed. The textures can be used to simulate all the different sensor bands (NVG, IR, and so on) without being regenerated.

The process is done automatically as the data is paged in from the server, and once the material textures have been generated, the texture array images are saved to the standard osgEarth file cache ready for fast loading on subsequent runs.

This section explains how to define generation of a material classified texture in an earth file.

The MCOD process:

- Processes any number of vector shape-files as raster images for each LOD. These shape-files represent material features (for example, roads, building footprints, water, and so on.)
- Combines the image textures to produce a per-pixel intensity modulation that maintains the spatial variation.
- Generates the material classified texture layers that are processed in real-time to calculate sensor wavelength-specific output using the sensor suite.

MCOD:

- Supports an unlimited number of material types by the use of texture arrays. Individual Eigen materials are stored as distinct components in each of the texture array layers.
- Stores individual materials as distinct components in each of the texture array layers.
- Specifies material components using osgEarth earth files. A feature component's color is specified using the standard style symbol set. This section describes how to use the styles necessary to define features plus any custom styles particular to MCOD.
- Caches files using the standard osgEarth pipeline to accelerate run-time load performance.

The MCOD plug-in uses standard osgEarth terminology and also these terms:

- texture layer. Corresponds to an index of an output texture array (0 to number of available layers-1).
- channel. An image color component (for example, R,G,B or A).
- default material. The material to use in the areas that are not defined by a shapefile.
- Intensity Modulation (IM). An optional channel to hold a gray scale representation of the visual imagery.
- feature level. Extends the osgEarth concept of level, but is particular to features, such as shapefiles.

This code is a template for setting up an MCOD layer in an earth file:

```
<map name="Name of the Map" type="geocentric" version="2">
    <!-- [Optional] Visible Image layers go here -->

    <!-- [Optional] Elevation source can be specified here -->
    <image
name="MaterialClassificationOnDemand" driver="mcod" visible="false"
shared="true">
        <!-- optional file cache id -->
        <cacheid>mcod</cacheid>
        <!-- defines the depth of the texture array, must be 1 or more
-->
        <num_texture_layers>2</num_texture_layers>

         <!-- for now texture compression MUST be set to none -->
         <texture_compression>none</texture_compression>
         <!-- assign default material channel (RGBA) and texture layer
-->
         <default_material_channel>#0000ff00</default_material_
channel>
         <default_material_texture_layer>0</default_material_texture_
layer>

         <!-- assign intensity modulation color channel (RGBA)
and             texture layer -->
         <intensity_modulation_channel>#000000ff</intensity_
modulation_channel
        <intensity_modulation_texture_layer>0</intensity_modulation_
texture_layer>

        <!-- specify minimum level where features (shapefiles) will
begin             to render -->
        <min_feature_level>10</min_feature_level>
        <!-- [Optional] image layers - to compute intensity modulation
-->
        <image>
            ...
```

```
        </image>
        <image>
            ...
         </image>
         <!-- features layers - contain material shapefiles to render
-->
         <features>
            ...
         </features>
         <features>
            ...
         </features>
         <!-- define rendering styles, by name, for the features -->
         <styles>
            <style type="text/css">
                ...
            </style>
         </styles>
    </image>
</map>
```

Layer flags (osgEarth standard):

- driver="mcod". Specifies the MCOD plug-in.
- visible="false". Layer will not be rendered directly.
- shared="true". Textures are made available to shaders.

MCOD specific variables:

- num_texture_layers. The number of texture layers defining the creation of the output texture array.
- default_material_channel. Channel (color component, RGBA order) that will contain the default material. in the example above, default material channel is defined as #0000ff00 indicating that the 8-bit Blue channel will contain the default material.
- default_material_texture_layer. Texture array id [0 to num_texture_layers-1] for the default material.
- intensity_modulation_channel. Channel (color component, RGBA order) that will contain intensity modulation. In the example above, the intensity modulation channel is defined as #000000ff, indicating that the 8-bit Alpha channel will contain IM.
- intensity_modulation_texture_layer. Texture array id [0 to num_texture_layers-1] for intensity modulation.
- min_feature_level. Similar to the osgEarth variable min_level, but is specific to features (material shapefiles). Default: 0.

- splat_combine_mode. Specifies how the procedural imagery materials should be combined with the MCOD layer's materials and what to use as the intensity modulation. This can be one of the following options:
  - NO_COMBINE. Default value for splat_combine_mode. Use this mode if there is not a procedural imagery/splatting layer that can be combined with the MCOD layer.
  - IM_SPLAT. Uses the visual procedural image layer as the intensity modulation for the SensorFX sensor modes. This mode requires that both the procedural imagery material layer and the MCOD layer use the same *.ms* file. See *./SharedData./19/latest/TerrainData/TerrainConfiguration/Test-MAK Earth - MCOD - Splat - Im Image Layer (online).earth* for an example of how to configure this mode.
  - IM_IMAGE_LAYER. Uses an image layer as the intensity modulation for the SensorFX sensor modes. This mode can use two separate *.ms* files, one for the procedural imagery material layer and one for the MCOD layer. See the *Test-MAK Earth - MCOD - Splat - Im Image Layer (online).earth* file for an example of how to configure this mode.

This code shows the MCOD setup for the Oahu, HI, area:

```
<image name="features"
       driver="mcod"
       opacity="1.0"
       visible="false"
       shared="true"
       min_filter = "LINEAR_MIPMAP_LINEAR"
       mag_filter = "LINEAR"
       max_range  = "100000000">
    <cacheid>mcod</cacheid>

    <!-- specify the minimum level that features (shapefiles) will
begin to render -->
    <min_feature_level>10</min_feature_level>

    <num_texture_layers>2</num_texture_layers>

    <texture_compression>none</texture_compression>

    <!-- assign default material channel (RGBA) and texture layer -->
    <default_material_channel>#0000ff00</default_material_channel>
    <default_material_texture_layer>1</default_material_texture_layer>

    <!-- assign intensity modulation color channel (RGBA) and texture
layer -->
    <intensity_modulation_channel>#000000ff</intensity_modulation_
channel>
    <intensity_modulation_texture_layer>0</intensity_modulation_
texture_layer>
```

6.1.1 Associating Styles and Features

Styles specify how features are rasterized into a layer. The feature rasterization is implemented using the osgEarth framework that wraps the AggLite renderer. Styles defined in the earth file control raster characteristics such as color and stroke width.

MCOD features must be associated with style. (Please see osgEarth documentation for information on setting up feature styling.) In addition to the osgEarth styling support, MCOD also supports a simple styling configuration by matching the feature name to the style name.

This code is for the Oahu, HI, area:

```
<TFSFeatures name="buildings-1">
  <url>http://vr-theworld.com/vr-theworld/mbtiles/osm/</url>
    <min_level>14</min_level>
    <max_level>14</max_level>
    <profile>spherical-mercator</profile>
    <format>pbf</format>
    <filters>
      ...
    </filters>
</TFSFeatures>

<TFSFeatures name="roads">
  <url>http://vr-theworld.com/vr-theworld/mbtiles/osm/</url>
    <min_level>14</min_level>
    <max_level>14</max_level>
    <profile>spherical-mercator</profile>
    <format>pbf</format>
    <filters>
      ...
    </filters>
</TFSfeatures>

<styles>
  <style type="text/css">
    buildings-1 {
      fill:          #ff0000;
      stroke:           #ff0000;
      stroke-min-pixels:  0;
      stroke-width:  0px;
      texture-layer:     0;
      altitude-clamping:  terrain-drape;
    }

    road{
      stroke:             #00ff00;
      stroke-width:       15m;
      stroke-min-pixels:  0;
```

```
      altitude-clamping:  terrain-drape;
      texture-layer:      0;
    }

    ...

</styles>
```

6.2. Streaming Sensorized Imagery

SensorFX supports streaming imagery. It is set up through the earth file. Create an image section with a name that contains emat texture. For example:

```
<GDALImage name="streaming emat2" "lod_blending="true" shared="true"
visible="false">
  <shared_sampler>jrmuSensorTex2</shared_sampler>
  <shared_matrix>jrmuSensorTexMatrix2</shared_matrix>
  <url>../../TerrainData/Terrain/DATABASENAME/emat2</url>
</image>

<GDALImage name="streaming emat3" "lod_blending="true" shared="true"
visible="false">
    <shared_sampler>jrmuSensorTex3</shared_sampler>
    <shared_matrix>jrmuSensorTexMatrix3</shared_matrix>
    <url>../../TerrainData/Terrain/DATABASENAME/emat3</url>
</image>
```

where *DATABASENAME* is the name of the terrain database.

emat2 and emat3 should be in separate directories. They must be GeoTIFF versions of the emat textures to ensure that osgEarth has the location information to correctly place the textures.

To specify an *.ms* file, include:

```
<external>
  <msfile>../../TerrainData/Terrain/DATABASENAME/
      databaseEmat.ms</msfile>
</external>
```

This *.ms* file can be used with streaming imagery, MCOD, or osgEarth terrains that are not material classified at all.

For information about *.ms* files and emat texture files, see also Chapter 16 on page 385.

The example HawaiiTifExample has a sample earth file called *HawaiiTifExample.earth* and an *.mtf* file called *HawaiiTifExample.mtf*. It shows how to set up the earth file and the terrain directory for using GeoTIFFs directly.

6.3. Using SensorFX with Geocentric Terrains

If you load a geocentric terrain without using an earth file, SensorFX does not work properly. You can work around this problem by creating an earth file that does not do anything and creating an MTF file that loads the earth file and the terrain.

The earth file should look like this:

```
<map name="MAK Earth" type="geocentric" version="2">
  <options>
    ...
  </options>
</map>
```

1. In SensorFX, load the earth file as a terrain server.
2. Load the geocentric terrain as a terrain patch.
3. Save the terrain as an MTF terrain.

By default, the observer will be in space, so create a saved view of the terrain at a useful location.

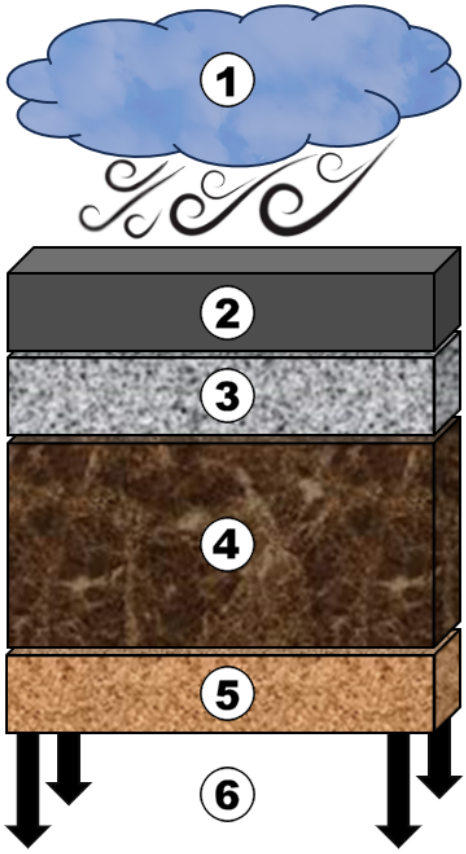
6.4. Material System Concept

SensorFX employs a "material system concept" to facilitate the classification of materials throughout a simulated scene.

There are three terms to understand about the material system concept:

- Material. Homogeneous composition of matter with specific context-independent physical properties.
- Layer. A single material with some specified thickness.
- 1D Material System. A set of layers of thicknesses that might vary, with context-dependent boundary conditions for the top layer and the bottom layer.

Figure 37. Example of 1D Material System



There are six parts to Figure 37, including four layers of ground (2–5 in the figure):

- 1 – Air convection of 19.2° C (66.5° F) at 0.44 m/s, comprised of direct and diffuse solar and radiative cooling.
- 2 – 0.05 m layer of asphalt.
- 3 – 0.05 m layer of gravel.
- 4 – 0.82 m layer of a homogenous composite material, 30% loam and 70% clay.
- 5 – 15 m layer of bedrock.
- 6 – Isothermal conduction at 15.5° C (60° F).

The top layer (2) is subjected to various thermal phenomena (1), such as air convection, and the bottom layer (5) is subjected to isothermal conduction (6).

6.4.1 Thermal Phenomena Modeling

The thermal phenomena applied to the top and bottom surface are configurable. SensorFX supports the modeling of the following thermal phenomena:

- Air convection
- Angle-dependent radiative loading from the sun, sky, and ground (Earthshine)
- Angle-dependent radiative emission (cooling) from the surface

- Rain loading (if rain present in environmental specification)
- Evapotranspiration
- Fixed, forced heat loads (Q1 & Qn)
- Conduction

The thermal phenomena used for each surface are defined by each surface's boundary condition type. The available boundary condition types differ slightly between the top and bottom surfaces:

- Top surface:
  - Return user-provided temperature
  - Natural air convection
  - Forced air convection with fixed user-provided temperature and airspeed
  - Forced air convection with ambient air temperature and fixed, user-provided airspeed
  - Aerodynamic skin heating
  - Return ambient air temperature
  - Insulated
  - Deep Sea
- Bottom surface:
  - Conduction
  - Natural air convection
  - Forced air convection with fixed user-provided temperature and airspeed
  - Forced air convection with ambient air temperature and fixed, user-provided airspeed
  - Aerodynamic skin heating
  - Return ambient air temperature
  - Insulated

Table 32 describes each boundary condition.

Table 32. Boundary conditions

ID	Name	Description
0	Return user-provided temperature	Simply returns the user's set temperature (T_1) and is used whenever an artificial heat source is driving the top face to a particular known temperature. This can be used for the top layer only. If used as a bottom-layer boundary condition, it is ignored.
	Conduction	Describes a solid conductive contact at a known, fixed temperature. Required user inputs are: • T_n $[degC]$ is the solid contact temperature • d is the material layer's thickness. For more information, see "Conduction" on page 221.
1	Natural air convection	Known as the "ambient" condition, it describes a typical surface subject to only natural convective and radiative effects. There are no required user-provided inputs for this boundary condition. For more information, see "Natural Air Convection" on page 222.
2	Forced air convection with fixed user-provided temperature and airspeed	Known as the "forced convection" condition, it describes a surface subject to artificial convective effects, e.g. as driven by an exhaust blower or air conditioning system. Natural static convection may also be present. When used with the top layer, required user-provided inputs are: • $T_1[degC]$ is the user-provided temperature. • $V_1[m/s]$ is the user-provided airspeed. When used with the bottom layer, required user-provided inputs are: • $T_n[degC]$ is the user-provided temperature. • $V_n[m/s]$ is the user-provided airspeed. For more information, see "Forced Air Convection (Fixed Temperature/Fixed Airspeed)" on page 222.

Table 32. Boundary conditions (continued)

ID	Name	Description
3	Forced air convection with ambient air temperature and fixed, user-provided airspeed	This condition is also a forced convection condition and has the same functional form as the previous case, except that here the forced air temperature is the same as the (computed) ambient air temperature, so that it is diurnally dependent and need not be supplied by the user. For more information, see "Forced Air Convection (Ambient Air Temperature/Fixed Airspeed)" on page 223.
4	Aerodynamic skin heating	This condition describes a surface traveling through the air at speed V, which is thus subject to air friction and subsequent heating of the surface. It is useful for application on aircraft nosecones and leading edges of wings or helicopter rotor blades. There are no required user-provided inputs for this boundary condition. For more information, see "Aerodynamic Skin Heating" on page 224.
5	Return ambient air temperature	This condition merely returns the current ambient air temperature, which can be useful for vegetation materials, but only in the case of no wind convection. There are no required user-provided inputs for this boundary condition.
6	Insulated	In this boundary condition, the surface is insulated from any external thermal influences, and cannot radiate, but is still subject to internal conduction. There are no required user-provided inputs for this boundary condition.
7	Deep sea	This boundary condition does not perform a thermal solution, but simply returns a top-layer surface temperature modeled from typical conditions for an ocean surface, at given latitude and date. The general trends are: • Exhibits a Gaussian decrease with absolute value of latitude. • Oscillates up to 8° C about mean with calendar month, with maxima/minima in February and August, depending on latitude. • Exhibits a small diurnal variation (approximately 2° C) with a peak at 2:00 PM.

A material system is defined in an *.ms* (material system) file. SensorFX provides default material system configuration files for various visuals, such as terrain, entities, and ocean. Material system files are tab-separated value files that you can open in a spreadsheet editor, such as Microsoft Excel.

6.4.2 Common 1D Material Systems

Table 33. Common 1D Material Systems

Layered Material	Top/Outermost Assignment Descriptor	Bottom/Innermost Assignment Descriptor
Roads, soils, rocky terrain	Ambient	Conduction (Low Temperature)
Small bodies of water	Ambient	Conduction (Low Temperature)
Large bodies of water	Forced air convection with ambient air temperature and fixed airspeed (medium)	Conduction (Low Temperature)
Road barrier, outdoor wall, boulder	Ambient	Ambient
Unoccupied or destroyed building	Ambient	Ambient
Occupied building walls and roof	Ambient	Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/High Airspeed)
Occupied building A/C units	Ambient	Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/Medium Airspeed)
Vegetation leaves, billboards, and signs	Ambient	Forced air convection with ambient air temperature and fixed air speed (Low Airspeed)
Vegetation bark/stem	Ambient	Ambient
Inactive or destroyed vehicle hull	Ambient	Forced air convection with ambient air temperature and fixed airspeed (Low Airspeed)
Active ground vehicle hull	Ambient	Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/High Airspeed)

Table 33. Common 1D Material Systems (continued)

Layered Material	Top/Outermost Assignment Descriptor	Bottom/Innermost Assignment Descriptor
Active ground vehicle exhaust, engine block	Ambient	Forced air convection with fixed, user-provided temperature and airspeed (High Temperature/Low Airspeed)
Active ground vehicle track or wheel	Ambient	Forced air convection with fixed, user-provided temperature and airspeed (Medium Temperature/Low Airspeed)
Active aircraft hull, wings	Forced air convection with ambient air temperature and fixed airspeed (High Airspeed)	Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/High Airspeed)
Active artillery	Ambient	Forced air convection with fixed, user-provided temperature and airspeed (High Temperature/High Airspeed)
Human	Forced air convection with ambient air temperature and fixed airspeed (Low Airspeed)	Conduction (Medium Temperature)
Ship deck or hull (above waterline)	Ambient	Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/High Airspeed)
Ship hull (below waterline)	Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/Low Airspeed)	Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/High Airspeed)

6.4.3 Definitions of Assignment Descriptor

Table 34. Assignment descriptor definitions

Descriptor	BC	T	V	Q
Conduction (Low Temperature)	0	15	-	-
Conduction (Medium Temperature)	0	37	-	-
Conduction (High Temperature)	0	150	-	-
Ambient	1	-	-	-
Forced air convection with ambient air temperature and fixed airspeed (Low Airspeed)	3	-	5	-

Table 34. Assignment descriptor definitions (continued)

Descriptor	BC	T	V	Q
Forced air convection with ambient air temperature and fixed airspeed (Medium Airspeed)	3	-	20	-
Forced air convection with ambient air temperature and fixed airspeed (High Airspeed)	3	-	175	-
Forced air convection with fixed user-provided temperature and airspeed (High Temperature/Low Airspeed)	2	150	20	-
Forced air convection with fixed, user-provided temperature and airspeed (High Temperature/High Airspeed)	2	150	100	-
Forced air convection with fixed, user-provided temperature and airspeed (Medium Temperature/Low Airspeed)	2	70	20	-
Forced air convection with fixed, user-provided temperature and airspeed (Medium Temperature/High Airspeed)	2	55	100	-
Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/Low Airspeed)	2	10	10	-
Forced air convection with fixed, user-provided temperature and airspeed (Low Temperature/High Airspeed)	2	22	20	-
Insulated	6	-	-	-
Fixed Low Heat Load	-/1	-	-	100
Fixed Medium Heat Load	-/1	-	-	1000
Fixed High Heat Load	-/1	-	-	10000

6.4.4 Boundary Conditions Thermal Equations

Conduction

The heat flux into the surface in the case of conduction is given by:

$$E = h_k \left(T_n - T_s\right) \left[W/m^2\right]$$

where:

- $T_s \left[degC\right]$ is the (computed) instantaneous temperature of the surface.
- $T_n \left[degC\right]$ is the (user-given) solid contact temperature.
- $h_k = kt/d$ is the ratio of the surface's thermal conductivity and material layer thickness.

Natural Air Convection

Natural air convection, specifically, is the convective effect is governed by the current wind speed and ambient air temperature, as well as the surface normal direction relative to the vertical ($\widehat{V}$). For surfaces with their `soil_type` flag set to `1`, the characteristic length is determined by the user's evapareamax setting; otherwise, it is an empirical constant.

The convective heat flux into the surface is given by:

$$E = h_c(T_a - T_s)\ [W/m^2]$$

where:

- $T_s\ [degC]$ is the (computed) instantaneous temperature of the surface.
- $T_a[degC]$ is the (computed) instantaneous ambient air temperature.
- $h_c\ [w/m^2/degC]$ is the convection coefficient, described as the sum of two contributions, a static contribution (h_o) present even when the air speed is zero, and a dynamic contribution (h_f) dependent upon wind speed. The general trends are:
  - Static ("ambient") contribution (h_o):
    - Increases slowly with $|T_a - T_s|$
    - Is smallest when $T_a - T_s$ and $\widehat{N} \bullet \widehat{Z}$ have the same sign
    - Is largest when $T_a - T_s$ and $\widehat{N} \bullet \widehat{Z}$ have opposite signs
    - Decreases slowly with evapareamax (if stype = 1)
  - Dynamic ("forced-air") contribution (h_f):
    - Increases slowly with V
    - Decreases slowly with evapareamax (if stype = 1)

Forced Air Convection (Fixed Temperature/Fixed Airspeed)

In particular, the forced convective effect is governed by the forcing mechanism's output airspeed V and temperature T, as well as the surface normal direction relative to the vertical ($\widehat{V}$). For surfaces with their `soil_type` flag set to `1`, the characteristic length is determined by the user's evapareamax setting; otherwise, it is an empirical constant.

The convective heat flux into the surface is given by:

$$E = h_c(T - T_s)\ [W/m^2]$$

where:

- $T_s\ [degC]$ is the (computed) instantaneous temperature of the surface.
- $T\ [degC]$ is the (computed) instantaneous ambient air temperature.

- h_c $[w/m^2/degC]$ is the convection coefficient, described as the sum of two contributions, a static contribution (h_o) present even when the airspeed is zero, and a dynamic contribution (h_f) dependent upon forcing airspeed. The general trends are:
  - Static ("ambient") contribution (h_o):
    - Increases slowly with $|T - T_s|$
    - Is smallest when $T - T_s$ and $\widehat{N} \bullet \widehat{Z}$ have the same sign
    - Is largest when $T - T_s$ and $\widehat{N} \bullet \widehat{Z}$ have opposite signs
    - Decreases slowly with evapareamax (if stype = 1)
  - Dynamic ("forced-air") contribution (h_f):
    - Increases slowly with V
    - Decreases slowly with evapareamax (if stype = 1)

Forced Air Convection (Ambient Air Temperature/Fixed Airspeed)

In particular, the forced convective effect is governed by the forcing mechanism's output airspeed V speed and ambient air temperature, as well as the surface normal direction relative to the vertical ($\widehat{V}$). For surfaces with their `soil_type` flag set to `1`, the characteristic length is determined by the user's evapareamax setting; otherwise, it is an empirical constant.

The convective heat flux into the surface is given by:

$E = h_c(T_a - T_s)$ $[W/m^2]$

where:

- T_s $[degC]$ is the (computed) instantaneous temperature of the surface.
- T $[degC]$ is the (computed) instantaneous ambient air temperature.
- h_c $[w/m^2/degC]$ is the convection coefficient, described as the sum of two contributions, a static contribution (h_o) present even when the air speed is zero, and a dynamic contribution (h_f) dependent upon forcing air speed. The general trends are:
  - Static ("ambient") contribution (h_o)
    - Increases slowly with $|T - T_s|$
    - Is smallest when $T - T_s$ and $\widehat{N} \bullet \widehat{Z}$ have the same sign
    - Is largest when $T - T_s$ and $\widehat{N} \bullet \widehat{Z}$ have opposite signs
    - Decreases slowly with evapareamax (if stype = 1)
  - Dynamic ("forced-air") contribution (h_f)
    - Increases slowly with V
    - Decreases slowly with evapareamax (if stype = 1)

Aerodynamic Skin Heating

The convective heat flux into the surface is given by:

$$E = h_a\,(T_{rec} - T_s)\,[W/m^2]$$

where:

- $T_s\ [degC]$ is the (computed) instantaneous temperature of the surface.
- $T_{rec}\ [degC]$ is an effective air temperature dependent on altitude, velocity, and surface normal.
- $h_a\ [w/m^2/degC]$ is a convection coefficient dependent on entity airspeed V. The general trends are:
  - Convection coefficient (h_a):
    - Increases linearly with V at high V
    - Increases slowly with V at low V
    - Increases slowly at $|T_a - T_s|$ at low V
    - At low V, is smallest when $|T_a - T_s|$ and $\widehat{N} \bullet \widehat{Z}$ have the same sign
    - At low V, is largest when $|T_a - T_s|$ and $\widehat{N} \bullet \widehat{Z}$ have opposite signs
  - Effective air temperature (T_{rec})
    - Decreases slowly with altitude
    - Increases moderately with $R_{type}\ \left|\widehat{N} \times \widehat{Z}\right|$ at high V
    - Increases moderately with V^2 at high V

6.5. Material Classification of CDB Terrains

6.5.1 Configuring CDB Raster Material

If your CDB database contains CDB Raster Material dataset (005_RMTexture and 006_RMDescriptor), you can configure your earth file to add support for material classification on demand (MCOD). This example shows how to materially classify terrain using the Yemen Sample CDB that was provided by Presagis/CAE.

You would add this section to your earth file:

```
<image name="cdb_rm" driver="cdbraster"
      shared="true"
      visible="false"
      min_filter     = "LINEAR_MIPMAP_LINEAR"
      mag_filter     = "LINEAR"
      max_range      = "100000000"
  vrfsim:enabled="false"
      cdb_dataset="5" cdb_component_selector1="1"
            cdb_component_selector2="1"
            enable_cache=myEnableCache>
        <cdb_mcod>true</cdb_mcod>
        <shared_sampler>jrmuMCODTextureSampler</shared_sampler>
        <shared_matrix>jrmuMCODTextureMatrix</shared_matrix>
        <cdb_root>E:\CDB\Yemen\CDB_Yemen_3_2</cdb_root>
</image>
<external>
  <msfile>../../ModelData/sensorFX/cdb.terrain.default.ms</msfile>
</external>
```

6.5.2 Configuring CDB to Sensorized CDB 3D Models (Geotypical or Geospecific)

If your CDB contains models that have textures with Raster Materials, you can configure your earth to also have MCOD for these datasets.

Geotypical Trees Dataset Example

```
<model driver="cdb"
      name="GT Trees"
      cdb_dataset="101"
      cdb_component_selector1="2"
      cdb_component_selector2="1" >
  <cdb_root>E:\CDB\myterrain\myCDBterrain</cdb_root>
        <msfile>../../ModelData/sensorFX
            /cdb.model_vegetation.ms</msfile>
        <mscolorfile>../../ModelData/sensorFX
            /cdb.model_vegetation.ms.mapping.csv</mscolorfile>
</model>
```

Geotypical Man-Made Dataset Example

```
<model driver="cdb"
       name="GT Man Made"
       cdb_dataset="101"
       vrfsim:enabled="true"
       cdb_component_selector1="1"
       cdb_component_selector2="1" >
  <cdb_root>E:\CDB\myterrain\myCDBterrain</cdb_root>
        <msfile>../../ModelData/sensorFX/cdb.model_manmade.ms</msfile>
        <mscolorfile>../../ModelData/sensorFX
            /cdb.model_manmade.ms.mapping.csv</mscolorfile>
</model>
```

Geospecific Man-Made Dataset Example Using Geotypical Texture

```
<model driver="cdb"
       name="GS Man Made"
       cdb_dataset="100"
       vrfsim:enabled="true"
       cdb_component_selector1="1"
       cdb_component_selector2="1" >
  <cdb_root>E:\CDB\myterrain\myCDBterrain</cdb_root>
        <msfile>../../ModelData/sensorFX/cdb.model_manmade.ms</msfile>
        <mscolorfile>../../ModelData/sensorFX
            /cdb.model_manmade.ms.mapping.csv</mscolorfile>
</model>
```

At this time, MAK ONE applications do not support sensorization with Geospecific Models referencing Geospecific textures.

6.5.3 CDB Base Material Mapping to Sensor Material Classification

MAK ONE applications have the file *./appData/cdb/CDBMaterials_ToProceduralClass.xml* that maps a CDB base material to a string used by the *.MS* file to give the CDB material the physical response in SensorFX.

For example, look at the following entry. When performing MCOD on the terrain, when SensorFX crosses a CDB composite material containing the CDB base material BM_ ASPHALT, it maps it to the values defined in GroundMCOD by MM_ROAD (defined in the MS file in the section below). If the same CDB material is used by a 3D model, it maps to the value in ModelMCOD, in this case MM_ASPHALT-ROOF (defined in the *.MS file* as shown below).

```
<Base_Material>
  <Name>BM_ASPHALT</Name>
  <Description>Asphalt</Description>
  <ClassValue>3</ClassValue>
  <GroundMCOD>MM_ROAD</GroundMCOD>
  <ModelMCOD>MM_ASPHALT-ROOF</ModelMCOD>
</Base_Material>
```

The *.ms* Files for CDB

MAK ONE data ships with three sets of SensorFX files for CDB:

- 3D models:
  - *./SharedData/19/latest/ModelData/sensorFX/cdb.model_manmade.ms*
  - *./SharedData/19/latest/ModelData/sensorFX/cdb.model_manmade.ms.mapping.csv*
- 3D vegetation models:
  - *./SharedData/19/latest/ModelData/sensorFX/cdb.model_vegetation.ms*
  - *./SharedData/19/latest/ModelData/sensorFX/cdb.model_vegetation.ms.mapping.csv*
- Terrain classification:
  - *./SharedData/19/latest/ModelData/sensorFX/cdb.terrain.default.ms*
  - *./SharedData/19/latest/ModelData/sensorFX/cdb.terrain.default.ms.mapping.csv*

7. Configuring Feature Data in VR-Forces and VR-Engage

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

7.1. Introduction 230
7.2. Named Queries 230
7.3. Feature Configuration Attributes 230
7.4. Attribute Transformations 231
7.4.1 Transformations as Aliases 231
7.4.2 Transformations as Derived Attributes 231
7.4.3 Transformations as New Attributes 231
7.4.4 Writing Multiple Transforms to Account for Possible Failures 232
7.5. Source-Specific Configuration 232
7.5.1 Loading Source-Specific Configuration Files 232
7.5.2 Redefining Queries 232
7.6. Extracting Feature Data from GDB to Shapefiles 233
7.7. Converting S-57 Feature Data to Shapefiles 233
7.8. Configuring Aggregate-Level Movement Restrictions 234
7.9. Dynamic Features 236

7.1. Introduction

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

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

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

7.2. Named Queries

Named queries are used throughout the VR-Forces and VR-Engage code and configuration to determine what features are. For example, this 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 referenced either in a simulation model set configuration or in the VR-Forces code. Do not change the names of these queries.

7.3. Feature Configuration Attributes

When VR-Forces or VR-Engage loads a feature in the simulation engine, it adds attributes that indicate where the feature came from. The applications can 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.
- 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 the applications can load directly.

7.4. Attribute Transformations

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

7.4.1 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 another name. You can use attribute transformations to create an "alias" for the different names, ensuring a single attribute name is used in code and configuration.

For example, this code defines the attribute MAK_WIDTH:

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

For each feature matching the named query ALL, this adds 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.

7.4.2 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, this 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.

7.4.3 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.

7.4.4 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 these 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.

7.5. 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.

7.5.1 Loading Source-Specific Configuration Files

Whenever VR-Forces or VR-Engage loads feature source, it 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 *./appData/settings/featureconfig.txt* for the details of source-specific configuration file loading by source type.

7.5.2 Redefining Queries

Source-specific configuration files can redefine queries in the main configuration file, helping the main configuration file to be as simple as possible. In this 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/19/latest/Terrain/Test/Features/RoadL.shp*. Any feature with the attribute "HIGHWAY" set to 1 is a main road.
- *./SharedData/19/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 definition:

  ```
  MAINROAD: HIGHWAY = 1
  ```

- *MainRoads.mfc* contains the 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.

7.6. Extracting Feature Data from GDB to Shapefiles

When you load feature data from a shapefile, it is used more efficiently than when you load it from a GDB file. (For details about loading feature data into a feature layer, see "2.4. Loading Features (Points, Lines, and Areas)" on page 81.) 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:

```
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.

7.7. Converting S-57 Feature Data to Shapefiles

S-57 feature data does not have a spatial index, causing it to load very slowly. By contrast, shapefiles can have a spatial index, which means 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 https://gdal.org/drivers/vector/s57.html and https://gdal.org/programs/ogr2ogr.html.)

S-57 files have many layers of data. A shapefile contains only one layer of data. To convert your S-57 data, you must create a shapefile for each layer of data that you want to convert. Since VR-Forces and VR-Engage do 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 variable:

```
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 or VR-Engage root. If you run ogr2ogr from *./appData/gdal_data*, you do need 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 and VR-Engage need. If you want to convert other layers, you must research the proper environment variable settings.

The command syntax is:

```
ogr2ogr -skipfailures [-nlt type] output_file
  input_file layer
```

where:

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

This example commands convert the layers that VR-Forces and VR-Engage need:

```
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.

7.8. 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:

```
  --
  -- Mobility areas
  --
  MAK_INFANTRY_UNRESTRICTED_TERRAIN:  MAK_ROAD
  MAK_INFANTRY_RESTRICTED_L1_TERRAIN: FOREST OR MOUNTAIN OR URBAN OR
DESERT
  MAK_INFANTRY_RESTRICTED_L2_TERRAIN: ALPINE
  MAK_INFANTRY_IMPASSABLE_TERRAIN:    MAK_WATERWAY

  MAK_MECH_UNRESTRICTED_TERRAIN:  MAK_ROAD
  MAK_MECH_RESTRICTED_L1_TERRAIN: HILLS OR CULTIVATED OR DESERT
  MAK_MECH_RESTRICTED_L2_TERRAIN: FOREST OR URBAN OR MOUNTAIN
  MAK_MECH_IMPASSABLE_TERRAIN:    MAK_WATERWAY OR ALPINE

  MAK_MOTOR_UNRESTRICTED_TERRAIN:  MAK_ROAD
  MAK_MOTOR_RESTRICTED_L1_TERRAIN: HILLS OR CULTIVATED OR DESERT
  MAK_MOTOR_RESTRICTED_L2_TERRAIN: FOREST OR URBAN OR MOUNTAIN
  MAK_MOTOR_IMPASSABLE_TERRAIN:    MAK_WATERWAY OR ALPINE

  MAK_TANK_UNRESTRICTED_TERRAIN:  MAK_ROAD
  MAK_TANK_RESTRICTED_L1_TERRAIN: HILLS OR CULTIVATED OR DESERT
  MAK_TANK_RESTRICTED_L2_TERRAIN: FOREST OR URBAN OR MOUNTAIN
  MAK_TANK_IMPASSABLE_TERRAIN:    MAK_WATERWAY OR ALPINE

  MAK_SHIP_IMPASSABLE_TERRAIN:    LAND
  MAK_SHIP_RESTRICTED_L1_TERRAIN: COAST OR RIVER OR LAKE
  MAK_SHIP_RESTRICTED_L2_TERRAIN: None
  MAK_SHIP_UNRESTRICTED_TERRAIN:  OCEAN
```

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

```
  (terrain-mobility
    (entry
      (query-name "MAK_MECH_UNRESTRICTED_TERRAIN")
      (speed-factor 1)
      (priority 100)
    )
    (entry
      (query-name "MAK_MECH_IMPASSABLE_TERRAIN")
      (speed-factor 0)
      (priority 200)
    )
    (entry
      (query-name "MAK_MECH_RESTRICTED_L2_TERRAIN")
      (speed-factor 0.25)
      (priority 300)
```

```
  )
  (entry
    (query-name "MAK_MECH_RESTRICTED_L1_TERRAIN")
    (speed factor 0.65)
    (priority 400)
  )
)
```

7.9. 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 other 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)—minefields, fortified lines, bridges, and so on—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_LINE:              {"1:17:0:0:0:-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_Abatis:                    {"1:17:1:0:62:3:0:0"}
MAK_VRF_Beam_Post:                 {"1:16:1:0:62:4:0:0"}
MAK_VRF_Falling_Blocks:            {"1:16:1:0:62:6:0:0"}
MAK_VRF_Road_Crater:               {"1:16:1:0:62:7:0:0"}

MAK_VRF_Engineering_Object_Lines:  {"1:17:1:-1:62:-1:-1:-1"}
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"}
...
```

8. Terrain Performance and Configuration

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

8.1. Configuring File Caching 238
8.1.1 Specifying the Location to Cache Terrain Server Data 239
8.1.2 Caching earth File Terrain Data 239
8.1.3 Enabling Texture Compression 240
8.1.4 Clearing the File Cache 241
8.2. Configuring Paged and Streaming Terrains 241
8.3. Debugging earth Files 241
8.3.1 Viewing a Terrain in Debug Mode 242
8.4. Using Shader-based Effect Maps 243
8.4.1 Composite Effects Maps 245
8.4.2 Debugging Shaders 246
8.4.3 Reloading Shaders 247
8.5. Indirect Rendering of Terrain 248
8.5.1 Scaling the LOD Distance for Indirect Rendering 248
8.5.2 Setting the LOD Transition Percentage 249
8.5.3 Disabling Indirect Rendering for Terrain 249
8.6. Specifying the Maximum Number of Paged LODs 250
8.7. Indirect Rendering for osgEarth Layers 250
8.8. Displaying Terrain Intersection Lines 250
8.9. Displaying DDS Textures Correctly 251
8.9.1 Flipping DDS Textures Globally 251
8.10. Optimizing GeoTIFF Files 252
8.11. Converting OpenFlight File Texture Data 252
8.11.1 Legacy Approach to OpenFlight File Texture Data 253
8.12. Scene Graph Nodes 254
8.12.1 Drawing Bounding Volumes for Scene Graph Nodes 254
8.12.2 Visualizing Scene Graph Node Bounds and LOD Range and Centers 255

8.13. Masking the Ocean 257
8.14. Debugging the VR-Forces Simulation Terrain 257
8.14.1 Starting the Debug Simulation Terrain Tool 258
8.14.2 Refreshing the Debug Display 258
8.14.3 Viewing Terrain Skin, Normals, and Collision Objects 258
8.14.4 Viewing Simulation Terrain Features 261
8.14.5 Displaying the NavMesh 264
8.14.6 Displaying the Query Area 265
8.14.7 Displaying Elevation Tiles 266
8.14.8 Displaying Recent Intersections 267
8.14.9 Displaying Page-in and Page-out Nodes 268
8.14.10 Enabling Auto Refresh 269
8.14.11 Simulation Terrain Debugging from the Simulation Engine Console 269

8.1. Configuring File Caching

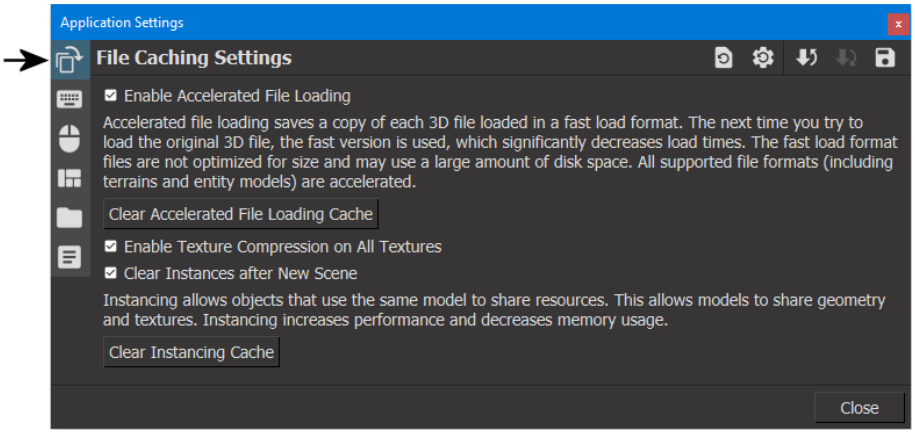
You can enable or disable file caching. When file caching is enabled, MAK ONE applications store information about the terrain and models in formats that enable the application to load them quickly. This improves loading time the next time the application loads the cached objects. When caching is disabled, 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 a MEDF version of the file. If it does not exist, it loads the file in the original format.
- Updating GPU drivers deletes any cached data on a machine. When you update the GPU drivers on a machine where you have MAK ONE cached data, you must regenerate the cache.

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

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 38).

Figure 38. File Caching Settings page



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

8.1.1 Specifying the Location to Cache Terrain Server Data

In MAK ONE applications, when file caching is enabled, the application caches data from terrain servers in the default locations. For the application, 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 GUI and sim engine 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 GUI uses OSGEARTH_CACHE_PATH and the sim engine uses VRFSIM_OSGEARTH_CACHE_PATH.
- If you only set VRFSIM_OSGEARTH_CACHE_PATH, the sim engine uses that value and the GUI uses the default cache location or the location specified in the earth file.

8.1.2 Caching earth File Terrain Data

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

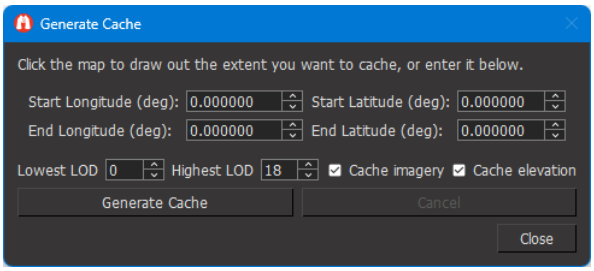
> It can take a very long time to generate the cache for large areas.

➤ **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 tab.

4. Click the Generate Cache button. The Generate Cache dialog box opens (Figure 39).

Figure 39. osgEarth Cache dialog box



5. Specify the start and end latitudes and longitudes of a rectangle that define the area you want to cache. You can do this by:
   - Entering coordinates for each of the four values.
   - Using the spin box up and down arrows.
   - Drawing a rectangle on the terrain.

     To draw a rectangle, click the left mouse button and release it to specify the upper left corner of the rectangle. Move the mouse to define the cache area. Left-click again to specify the lower right corner.
6. Optionally, enter the lowest LOD level and the highest LOD level.
7. Use the check boxes to specify whether you want to cache the image data, elevation data, or both.
8. Click Generate Cache. A message displays when the cache is complete.
9. Optionally, repeat steps 5 through 8 to cache another area.
10. When you are finished, click Close.

ⓘ To cancel the cache generation after starting it, click the Cancel button.

8.1.3 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 when you add 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 38 on the previous page).
3. Select or clear the Enable Texture Compression On All Textures check box.

8.1.4 Clearing the File Cache

You can delete 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. In that case, 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 38 on page 239).
3. Click Clear Accelerated File Loading Cache.

8.2. Configuring Paged and Streaming Terrains

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

- 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 GUI.)
- 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.

8.3. 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, you can 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 177.

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:

- Visible. Enable or disable display of the layer.
- Opacity. Set the opacity of the layer. The range of values is 0.0 through 1.0.
- Minimum Visible Range. Specify the minimum camera range at which this layer is visible.
- Maximum Visible Range. Specify 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 these 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. Specifies the percentage of land that this layer's instances cover. Lower values result in a patchy appearance.
- Maximum Distance. Specifies the maximum viewing distance from the camera.
- Brightness. Defines the level of brightness to apply. Default 1.0. (Some terrains provided with MAK ONE applications override the default.)
- Contrast. Sets the level of contrast to apply. Default: 0.5.

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

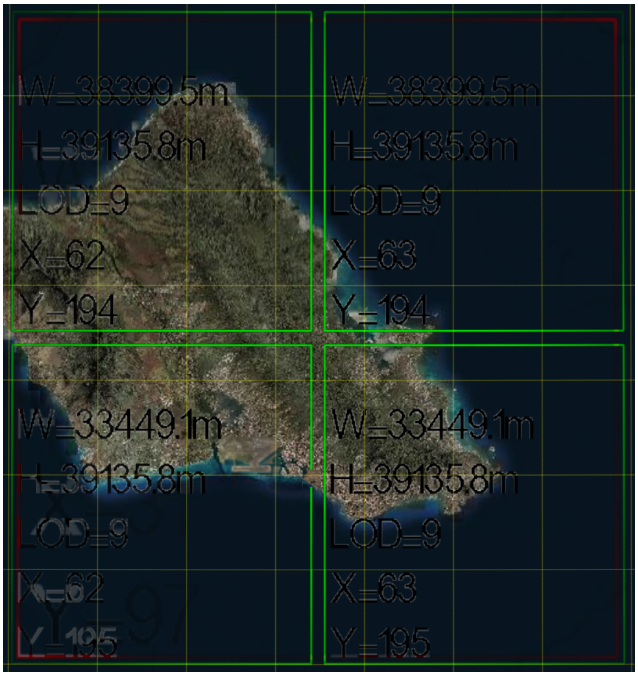
The properties described above are displayed in the Runtime Options window on the Terrain Editor. For details about viewing and changing properties, see "4.3.2 Editing the Properties of a Layer" on page 178.

> 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, please see "Filtering Earth File Layers" in the *VR-Forces User Guide*.

8.3.1 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 40. 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 and you see the osgEarth tile names. This also allows you to know what LOD you are seeing. Use this information to help debug a problem you are experiencing at a particular location in your terrain.

➤ **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.

➤ **To exit debug mode:**

1. Select the name of the terrain in the Terrain Content window. The Properties window changes to show the Debug Mode check box.
2. Clear the Debug Mode check box.

8.4. 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 (for example, a window and perhaps its 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. *Test-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 CameraFX'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 they use to process the files properly without any additional configuration. To specify the type of map file, use these 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 for Linux support.

To see the effect of effect maps, you must enable lighting. For details, please see the chapter:

- *Lighting Effects* in the *VR-Vantage User Guide*
- *Lighting Effects* in the *VR-Forces User Guide*

8.4.1 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 must use the Composite Effects Maps naming convention.

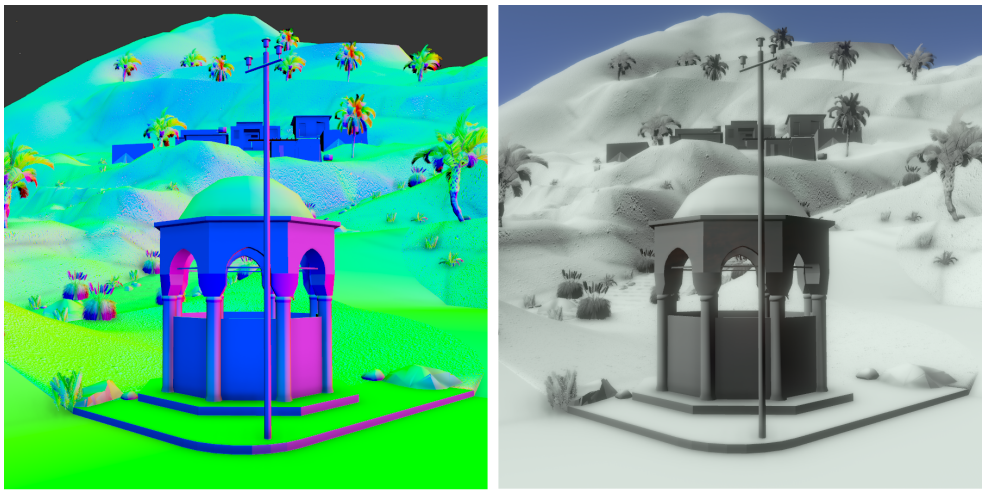
- NML: 3 Channel Normal map
- SPR: 1 Channel Specular map
- EMS: 1 Channel Emissive map
- LMM: 3 Channel Lightmap (supported for CDB)

- 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

8.4.2 Debugging Shaders

MAK ONE applications provide some shader debugging capabilities within the 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 41 shows a view of VR-Village with two different shader debug settings.

Figure 41. Shader component visualization

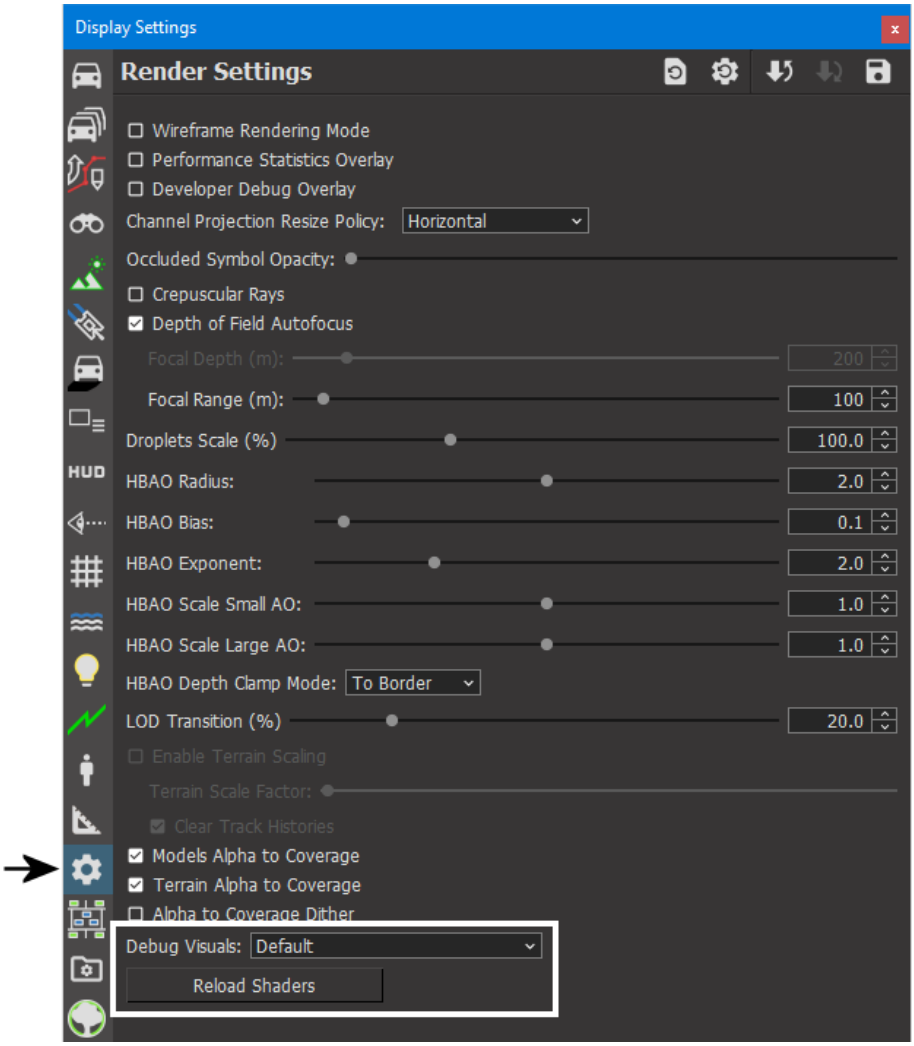


Bumpmap Normals | Diffuse Light

➤ **To debug shaders:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 42).

Figure 42. 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.)

When working with material classifications for MAK ONE applications with SensorFX, you can access more specific material system debugging in the Developer Debug Overlay, which you access by pressing **F4**. Use the options under SensorFX Material System Debug Settings.

8.4.3 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 42 on the previous page).
3. Click Reload Shaders.

8.5. 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:

- "2.6.6 MAK ONE Custom Feature Processing" on page 99
- "4.4.1 Adding Elevation Data to a Terrain" on page 180
- "3.1. Connecting to Terrain Servers" on page 154
- "8.5.1 Scaling the LOD Distance for Indirect Rendering" below
- "15.18. Disabling Indirect Rendering for an OpenFlight Model" on page 381

Indirect rendering does not affect the rendering of:

- Extracted external references
- SpeedTrees
- Triton dynamic ocean
- SilverLining sky
- osgEarth terrain skin
- Tactical graphics
- TerraPage terrains

8.5.1 Scaling the LOD Distance for Indirect Rendering

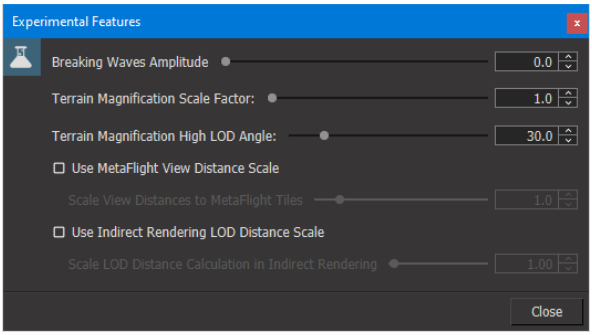
When you use indirect rendering, you can dynamically adjust the distance at which LODs are paged in. If a terrain contains high LODs for objects at distances from the eyepoint at which the details are not visible, you can pay the performance price for the LODs without the visual benefit of high resolution. By scaling the LOD distance, you can cause MAK ONE applications 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 43).

Figure 43. 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.

8.5.2 Setting the LOD Transition Percentage

When a MAK ONE application transitions between LODs, it blends them to ensure that the change in resolution is not visually jarring. You can set the LOD transition percentage to see the most pleasing effects for your terrain. A setting of 0 (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 42 on page 247).
3. Adjust the Indirect LOD transition percentage slider.

8.5.3 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 an ImGui menu, the Developer Debug Overlay.

➤ **To disable indirect rendering:**

1. Press **F4** to open the Developer Debug Overlay (Figure 1 on page 52).
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.

8.6. 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 display in the Properties window.
4. Edit the value for the Max Paged LOD property.
5. Save the terrain.

8.7. 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.

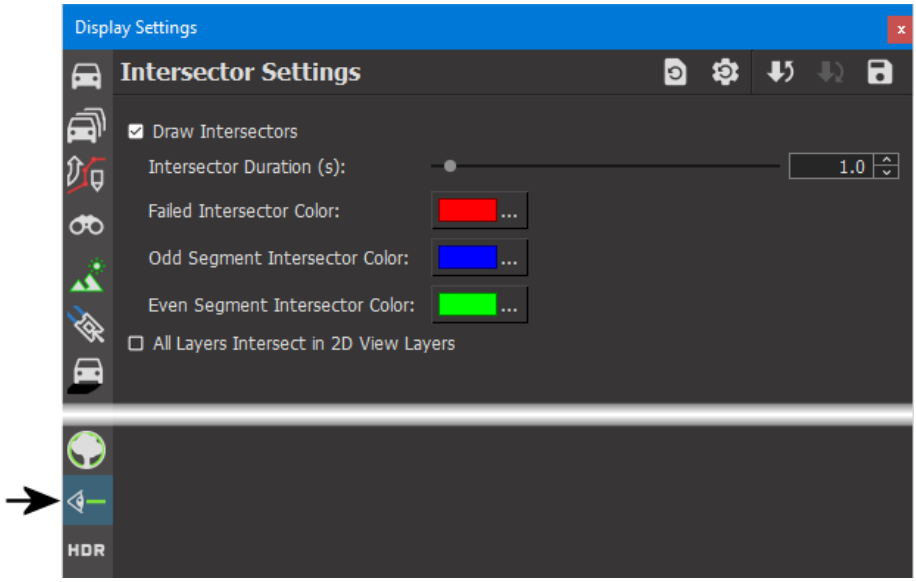
8.8. Displaying Terrain Intersection Lines

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 44).

Figure 44. 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.

8.9. 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:

- When you add a terrain patch. See "4.4.1 Adding Elevation Data to a Terrain" on page 180.
- On a per-model basis. See "15.15. Flipping DDS Textures for a Model" on page 376.
- As a global setting. See "8.9.1 Flipping DDS Textures Globally" below.

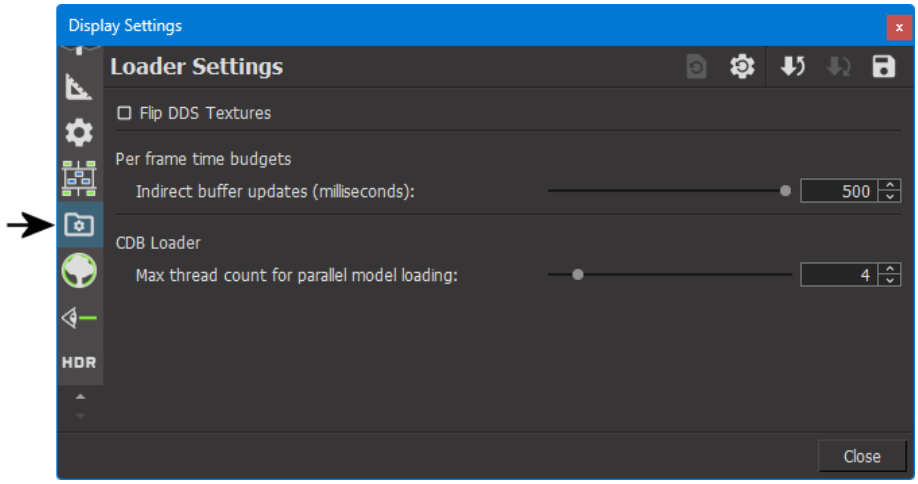
8.9.1 Flipping DDS Textures Globally

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 whether textures should 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 pages 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 45).

Figure 45. Loader Settings page



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

8.10. Optimizing GeoTIFF Files

You can optimize GeoTIFF files to improve their performance. There are several methods to consider.

- Build internal tiles. This is always good for GeoTIFFs and often improves performance. You can use the gdal_translate program. For more information, go to https://gdal.org/programs/gdal_translate.html.
- Reproject the imagery. While osgEarth reprojects your data, doing it yourself offline can improve performance. Optimal projections are WGS84 (EPSG:4326) or Spherical Mercator (EPSG:3857). There are several tools you can use, one of which is gdal_warp. For information on that program, go to https://gdal.org/programs/gdalwarp.html. The Global Mapper or ArcGIS tools are also available.
- Build overviews. Overviews are image pyramids at low resolution that can improve the performance of large data sources. You can use the gdaladdo program. For more information, go to https://gdal.org/programs/gdaladdo.html.

The gdal info program provides information on your GeoTIFF file, such as the block size. For more information, go to https://gdal.org/programs/gdal_info.html.

8.11. Converting OpenFlight File Texture Data

For OpenFlight, if the polygons contain an SMC (surface material code) or FID (DFAD Feature ID), the makOsg FLT loader puts the SMC or FID in a comment node when the model is converted to osg (medf).

The priority is first to check the SMC, then the FID, and finally the texture/soiltype mapping described in "8.11.1 Legacy Approach to OpenFlight File Texture Data" on the facing page.

When VR-Forces loads the terrain and parses all of the geometry, if it finds a comment then it uses the *./appData/settings/vrfSim/smc_fid_soiltype_map.csv* file to map to the VR-Forces soiltype.

For FBX models, the concept of SMC/FID does not exist. To do the same thing, the database modeler must add the comments in the FBX nodes.

The comment uses the format `@dis material_smc` *xx* or `@dis material_fid` *xx*

8.11.1 Legacy Approach to 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 soiltype information from the names of the texture files that are usually associated with OpenFlight files.

The VR-Forces sim engine soiltype can come from multiple sources:

- Landcover data, where the landcover values are mapped to the VR-Forces soiltype (*./appData/settings/vrfSim/landCoverDataSurfChar.map* and optionally *myterrain.landCoverDataSurfChar.map*).
- Terrain inset (OpenFlight or FBX) where textures are mapped to a soiltype, for example, in a *./appData/settings/vrfSim/myterrain.SurfChar.map* file.
- Feature datasets mapped to a feature type and then to a soiltype in *./appData/settings/featureconfig.txt*.

This applies to FBX terrains as well as OpenFlight.

By default, VR-Forces tries to determine the soiltypes 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 soiltype information in the file you are importing, you can map the texture files to soiltypes in the *surfChar.map* file. This file is specific to each terrain. For an example, see *./SharedData/19/latest/TerrainConfiguration/AlaMoana.surfChar.map*.

Table 35 lists the MAK soiltypes to which you can map the OpenFlight texture files.

Table 35. MAK soiltype

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 soiltypes: 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 soiltype.

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 soiltype.

This 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 soiltype.

> The list of supported soiltypes is fixed. You cannot add new soiltypes.

8.12. Scene Graph Nodes

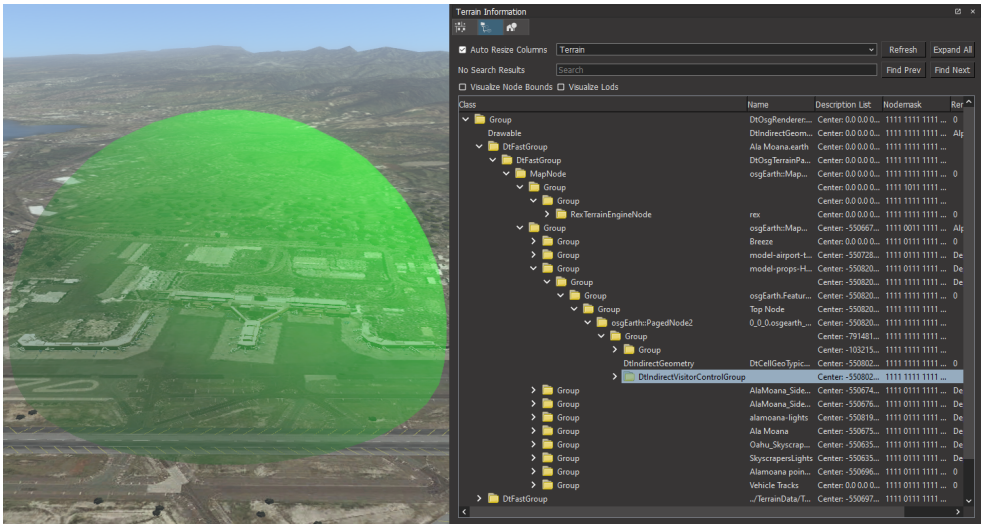
8.12.1 Drawing Bounding Volumes for Scene Graph Nodes

You can draw bounding volumes for the nodes in the scene graph. This allows a terrain artist or developer to visualize the size of the bounding volume for a portion of the terrain and thereby know when it will be culled from the scene. It is also useful for finding out whether 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 46 shows a bounding volume for the terminal area of the airport in Honolulu, Hawaii, USA, in the MAK Earth (online) terrain.

Figure 46. 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.

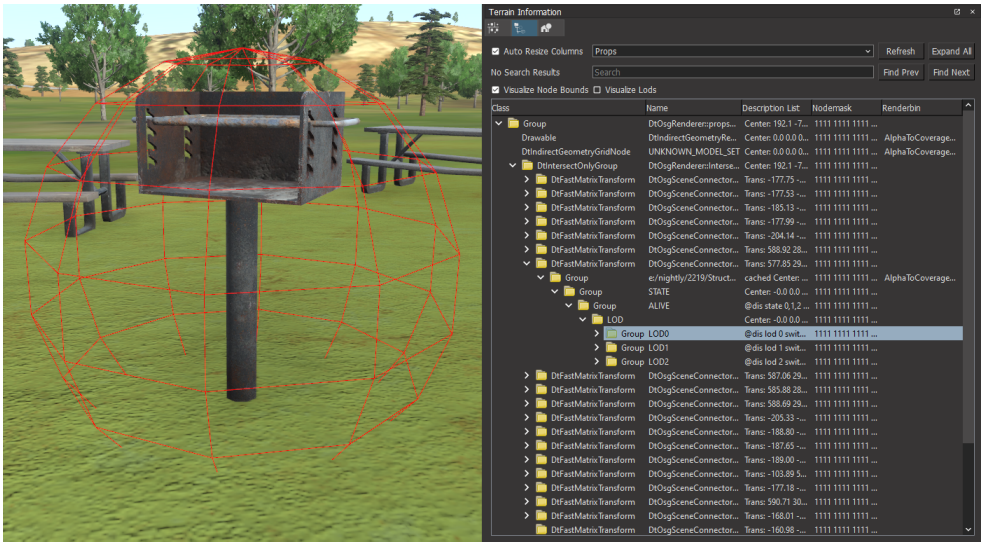
8.12.2 Visualizing Scene Graph Node Bounds and LOD Range and Centers

You can visualize scene graph node properties, specifically the node bounds and the LOD centers and ranges. This is a way that a terrain artist or developer can debug the terrain. For example, you can use it to inspect objects by ensuring the nodes in the scene graph are what you see visually, to debug culling and intersections, and ensure that LOD distances are set up properly. It is helpful when working with indirect.

Node Bounds

Figure 47 shows the node bounds for the park grill prop on Ground DB II.

Figure 47. Node bounds



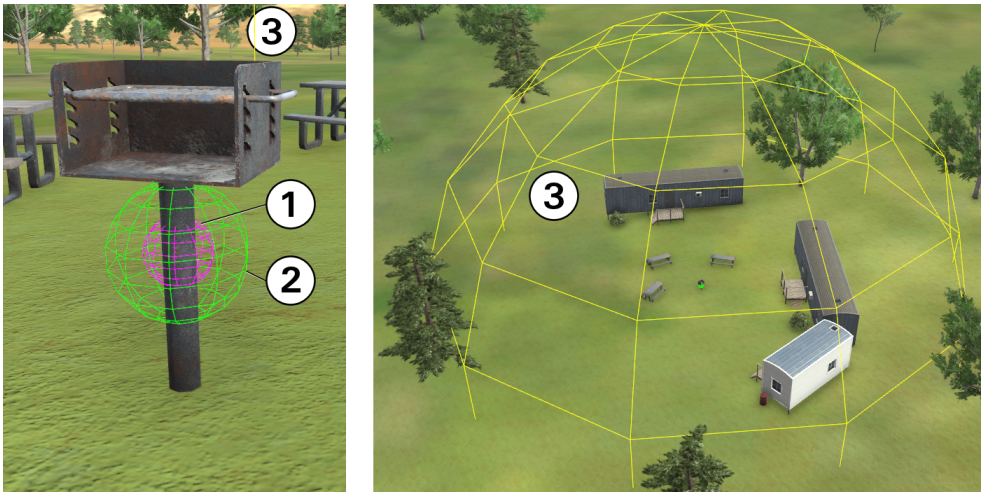
➤ **To display prop node bounds:**

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 the type of object you want to inspect.
4. Click Refresh.
5. Select the Visualize Node Bounds check box.
6. Expand the scene graph to display the node whose node bounds you want to display.
7. Move the observer until you see the node bounds.

Node LODs

Figure 48 shows the grill's LOD center (1), range minimum (2), and range maximum (3).

Figure 48. Node LODs



➤ **To display prop node LODs:**

1. Select the Scene Graph tab in the Terrain Information Panel.
2. In the drop-down list at the top of the tab, select the type of object you want to inspect.
3. Click Refresh.
4. Select the Visualize LODs check box.
5. Expand the scene graph to display the node whose node LODs you want to display.
6. Move the observer until you see the node LODs.

8.13. 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.
- See "15.14. Adding a Depth Water Mask to a Model" on page 375 to learn how to mask water from inside simulation objects.

8.14. Debugging the VR-Forces Simulation Terrain

VR-Forces has a tool that you can use to help debug problems in the simulation terrain, the Debug Simulation Terrain tool. The GUI and sim engine 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 these aspects of the terrain:

- The terrain skin
- Point, areal, and linear features

- The navigation mesh
- Terrain queries

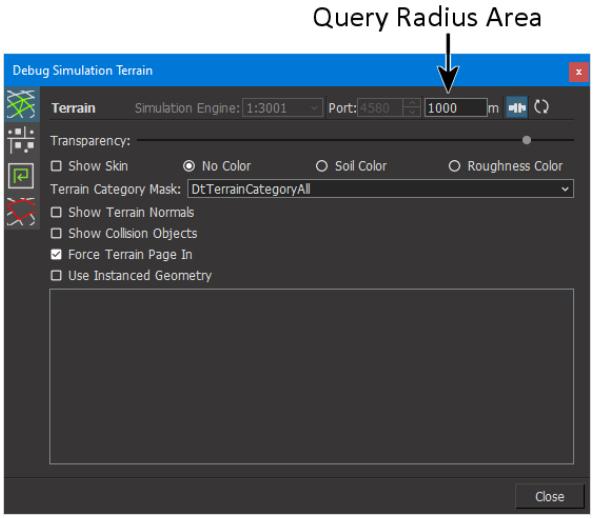
8.14.1 Starting the Debug Simulation Terrain Tool

Use this procedure 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 49).

Figure 49. Debug Simulation Terrain, Terrain page



2. If you are running more than one sim engine, in the Simulation Engine list, select the sim engine 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 sim engine.

8.14.2 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, see "8.14.10 Enabling Auto Refresh" on page 269.

➤ To refresh the display, click the Refresh button ( ).

8.14.3 Viewing Terrain Skin, Normals, and Collision Objects

Use the Terrain page to view the terrain skin, terrain normals, and collision objects. When you view the terrain skin you can also associate soiltype or roughness type with a color. The colors associated with soiltypes and roughness types are configured in *./appData/settings/vrfGui/debugDrawColors.csv*.

You have an option to force page-in of the terrain. By 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.

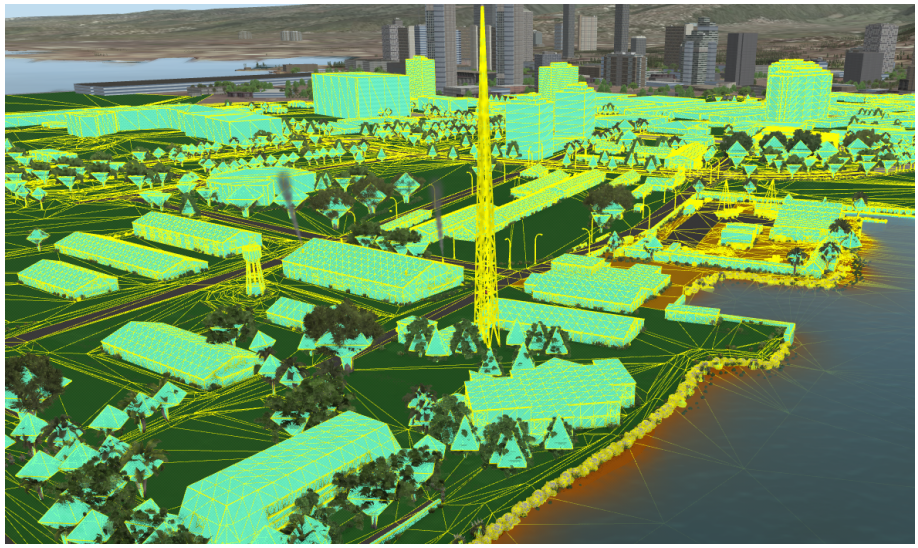
➤ **To show the terrain skin:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on the previous page).
2. Select the Terrain page.
3. Select the Show Skin check box. The terrain skin displays (Figure 50).

Figure 50. Showing the terrain skin

4. Optionally, select the Soil Color or Roughness Color option buttons. Figure 51 shows the terrain with soil color. (To eliminate the glow effect, disable high dynamic range (HDR).)

Figure 51. Terrain skin with soil color

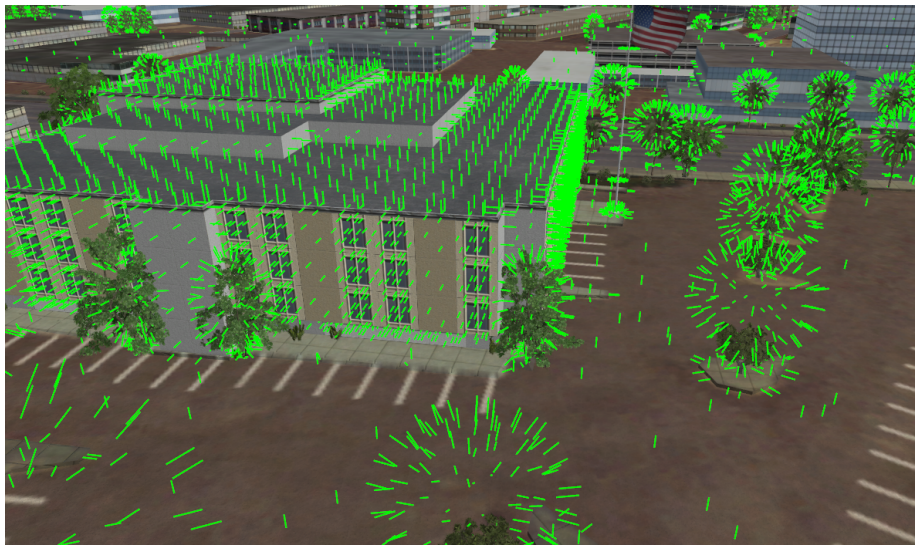


5. Optionally, select or clear the Force Terrain Page In check box.

➤ **To show terrain normals:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on page 258).
2. Select the Terrain page.
3. Select the Show Terrain Normals check box. The terrain normals display (Figure 52).

Figure 52. 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 49 on page 258).
2. Select the Terrain page.
3. Select the Show Collision Objects check box. The terrain normals display (Figure 53).

Figure 53. Collision objects

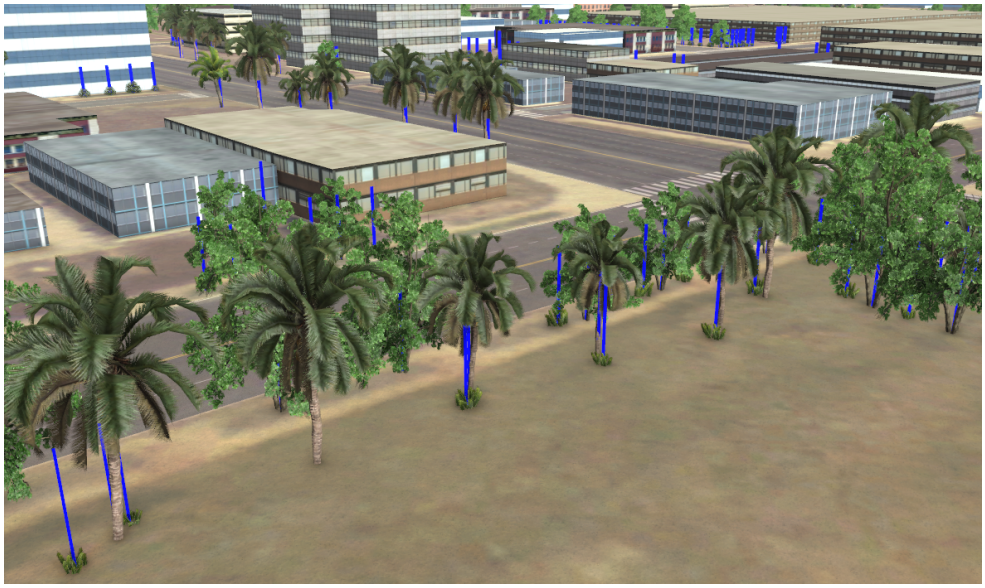
4. Optionally, select or clear the Force Terrain Page In check box.

8.14.4 Viewing Simulation Terrain Features

You can display point, areal, and linear features. Features display based on a set of preconfigured queries that you can select. You can also enter your own queries. For information about defining queries, see "7.2. Named Queries" on page 230.

Figure 54 shows how point features display.

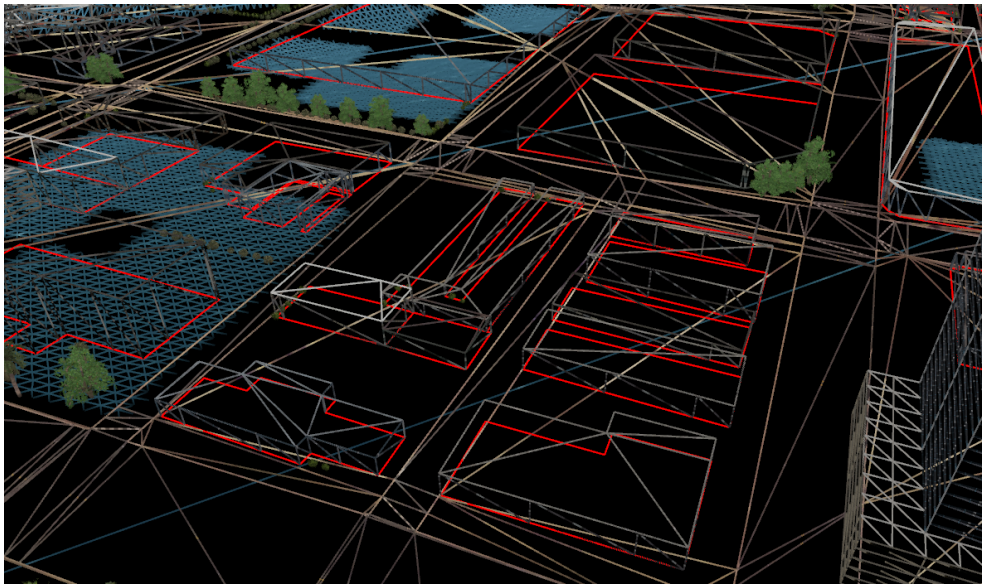
Figure 54. Point features



Areal features display extruded buildings. Changing to wireframe mode helps you see the areas.

Figure 55 shows how linear features display when wireframe is enabled.

Figure 55. Areal features



Each linear feature displays using a different color. There is no particular significance to the color choices.

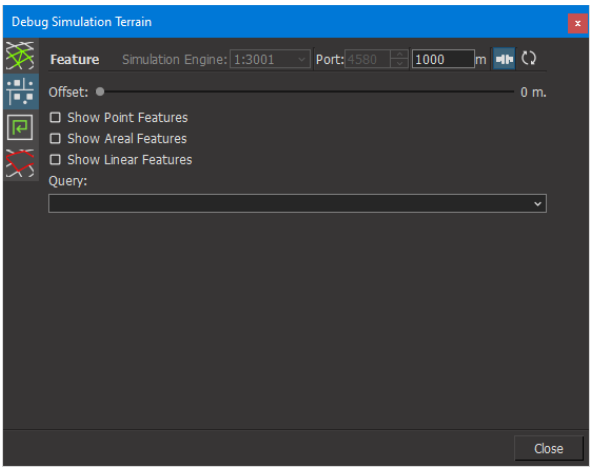
Figure 56 shows how linear features display.

Figure 56. Linear features

➤ **To display simulation terrain features:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on page 258).
2. Select the Feature page (Figure 57).

Figure 57. Debug Simulation Terrain, Feature page

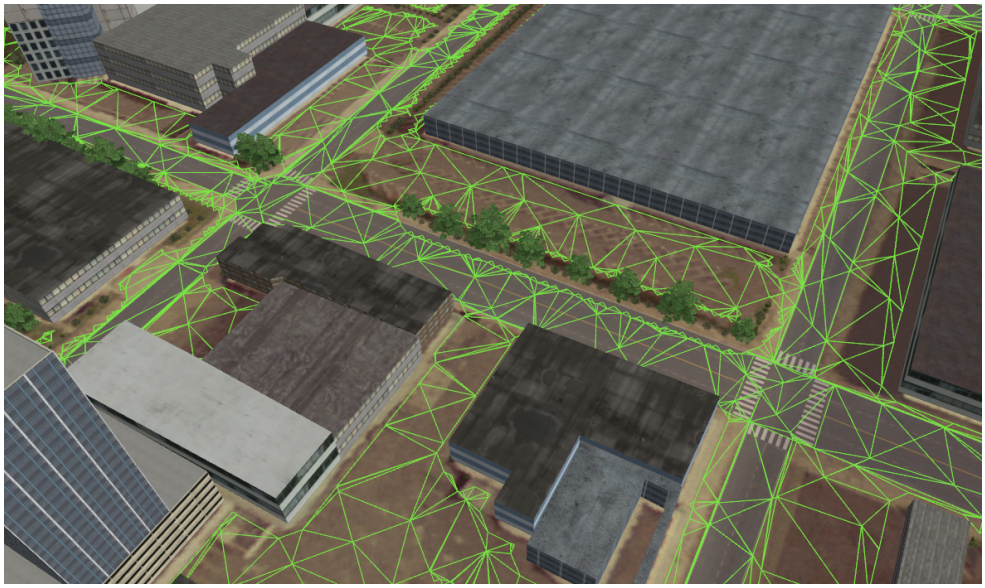


3. Select the check box for each feature type that you want to display.
4. In the list, select or enter a query.
5. Optionally, adjust the Offset slider. It raises the feature display above the terrain, which can improve visibility.

8.14.5 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 *Generating Navigation Data* chapter in the *VR-Forces User Guide*. Figure 58 displays a navigation mesh.

Figure 58. NavMesh

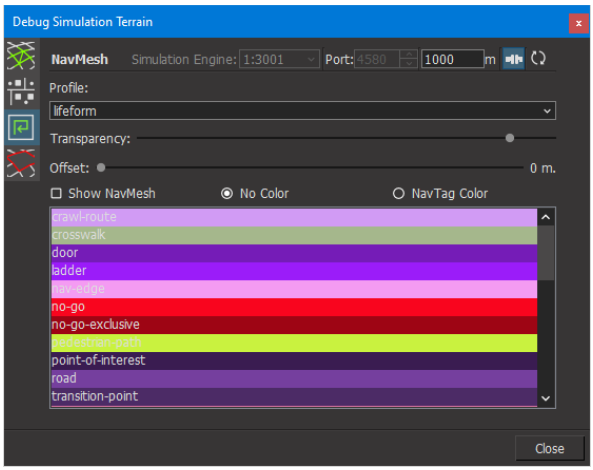


The NavMesh does not always align with the terrain.

➤ **To display the NavMesh:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on page 258).
2. Select the NavMesh page (Figure 59).

Figure 59. Debug Simulation Terrain, 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.

8.14.6 Displaying the Query Area

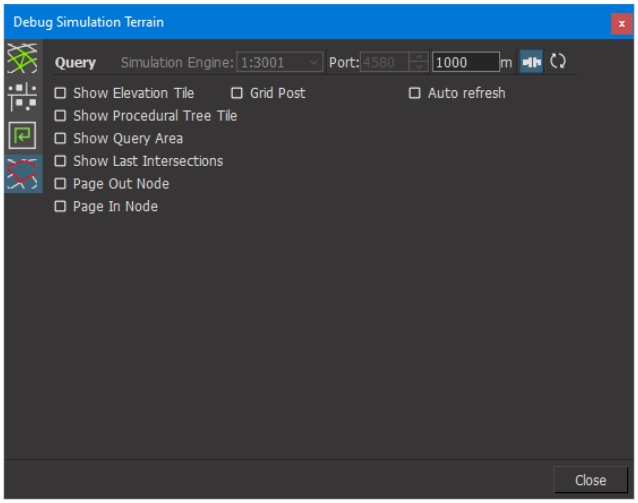
You can display a box that shows the query area radius (Figure 60).

Figure 60. Query area

➤ **To display the query area:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on page 258).
2. Select the Query page (Figure 61).
3. Select the Show Query check box. A red box appears on the terrain.

Figure 61. Debug Simulation Terrain, Query page

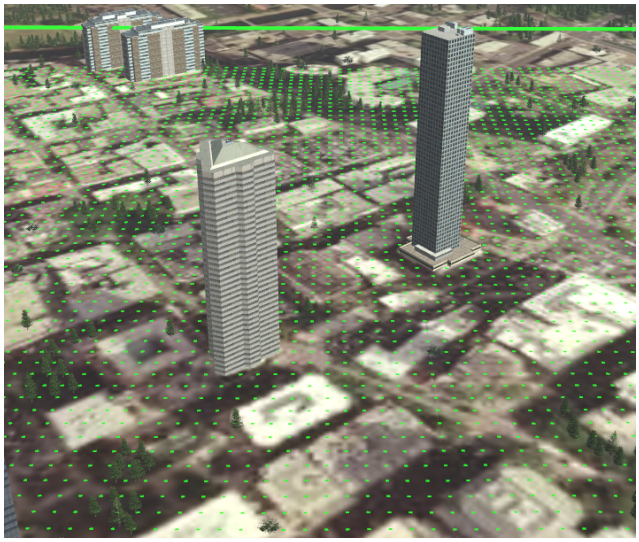


8.14.7 Displaying Elevation Tiles

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 62). You can also display the grid posts in those areas.

Figure 62. Elevation tiles

Figure 63. Elevation tile and Grid post



VR-Forces only shows elevation tiles that are within or overlap the boundary of the query radius area.

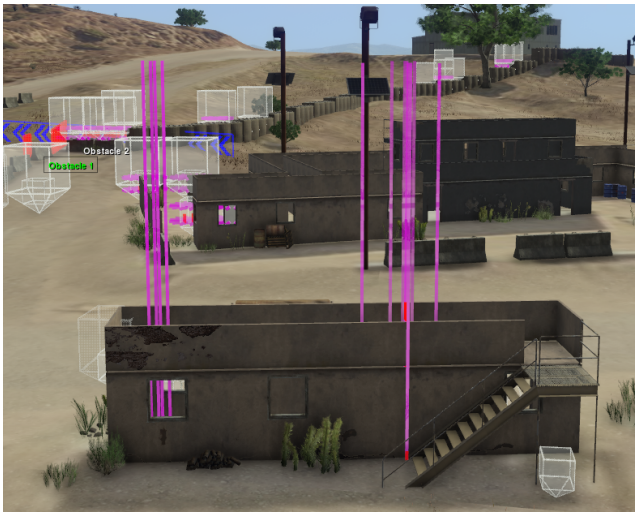
➤ **To display elevation tiles:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on page 258).
2. Select the Query page (Figure 61 on the previous page).
3. Select the Show Elevation Tile check box.
4. Optionally, select the Grid Post check box.

8.14.8 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 64. Intersections



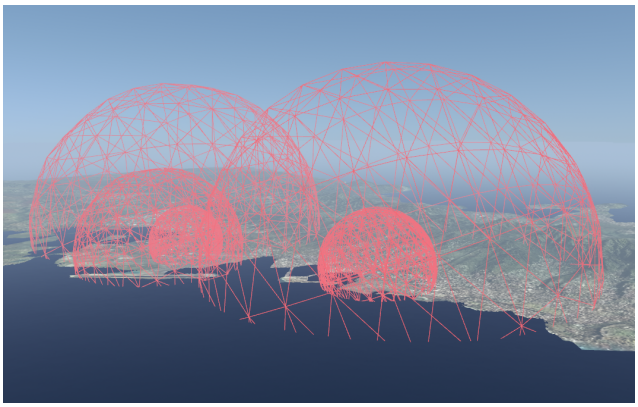
➤ **To display intersections:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on page 258).
2. Select the Query page (Figure 61 on page 266).
3. Select the Show Last Intersections check box.

8.14.9 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 must be paged in.

Figure 65. Page out nodes



➤ **To display page-in and page-out nodes:**

1. Choose **Settings > Debug Simulation Terrain**. The Debug Simulation Terrain Tool opens (Figure 49 on page 258).
2. Select the Query page (Figure 61 on page 266).

3. If you are in Stealth observer mode, you may need to zoom out to seen the full extents.
4. Select or clear the Page Out Node or Page In Node check boxes.

8.14.10 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 49 on page 258).
2. Select the Query page (Figure 61 on page 266).
3. Select the Auto Refresh check box.

8.14.11 Simulation Terrain Debugging from the Simulation Engine Console

You can display additional terrain paging information by entering commands on the sim engine console. You can also display information about feature sets if they are present. Table 36 lists the commands and their results.

Table 36. Terrain paging information commands

Command	Result
`features config`	Prints queries and transforms from *featureconfig.txt*. For details about feature configuration, see "Chapter 7. Configuring Feature Data in VR-Forces and VR-Engage" on page 229.
`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 GUI 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 36. 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 sim engine console.
2. Enter the command. You can enter commands even while the console is displaying status messages. It remembers what you entered.
3. Press **Enter**.

9. Terrain Tutorials

This section explains how to create a local terrain server. It also shows how to extract props from a terrain.

9.1. Creating an earth File to Load Terrain Data 272
- 9.1.1 Add a Map Element 272
- 9.1.2 Include Options and Elevation in the earth File 272
- 9.1.3 Add Imagery to the earth File 272
- 9.1.4 Add Feature Data to the earth File 273
- 9.1.5 Save the earth File 276
- 9.1.6 Load the earth File and Save it as an MTF File 276

9.2. Extracting Props from a Terrain 278
- 9.2.1 Create a New Terrain and Add a Terrain Patch 278
- 9.2.2 Extract the Props 279
- 9.2.3 Change the Model Definition for a Prop 281

9.1. 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 *Test-Little Pond (online).earth* that merges local Little Pond data with streaming terrain from VR-TheWorld. For this tutorial, we 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, replace, or remove elements as needed. (*./SharedData/19/latest/TerrainData/TerrainConfiguration/Example-ComposedTerrain.earth* was designed for this use case.). For this tutorial, however, we start from an empty file and build all the major elements.

9.1.1 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 this code:

```
<map name="Little Pond" type="geocentric" version="2">

</map>
```

9.1.2 Include Options and Elevation in the earth File

We add options, elevation, and imagery information to our earth file by making references to XML files that have that information. In our example, we use offline information.

Add Options to the earth File

Add a reference to a file to include common options. Add the line at the top of the file, just below the opening `map` element. We use the options for the REX terrain engine, which is the default for terrains in MAK ONE applications.

```
{% include options.terrain-rex-default.xml %}
```

Add Elevation Data to the earth File

We want this terrain to load the elevation data in *./SharedData/19/latest/TerrainData/TerrainConfiguration/elevation.worldwide.lowres.xml*. Elevation data is configured in the XML file that we reference using a `GDALElevation` element. The location of data in an earth file is provided in a `url` element.

Add this text after the line where we included the options:

```
{% include elevation.worldwide.lowres.xml %}
```

9.1.3 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 see only one of the images.

To use more than one image, we place an `image` element for each image file as layers inside a `CompositeImage` element, which uses a composite driver. When adding multiple images, be sure to list the images from bottommost to topmost. If you list them in the opposite order, for example, *LittlePond.tif* would be underneath *W72N42.tif* and you will not be able to see it.

Add this text after the line where we include the elevation information:

```
<CompositeImage name="Little Pond Imagery">
  <layers>
    <GDALImage name="Surrounding Area Imagery">
      <url>../../TerrainData/Terrain/LittlePond/Imagery/W72N42.tif
</url>
    </GDALImage>
    <GDALImage name="Little Pond Imagery" min_level=”14”>
      <url>
../../TerrainData/Terrain/LittlePond/Imagery/LittlePond.tif</url>
    </GDALImage>
  </layers>
</CompositeImage>
```

We use the GDAL driver to load the images.

> We use local imagery for our example. The *Example-ComposedTerrain.earth* file includes a reference to an XML file, in the same way we have included the options and elevation data in our example. That XML file references imagery information included in the MAK ONE data package. The *Test-Little Pond (online).earth* file also includes a reference to an XML file, but that XML file references imagery information online. See *Example-ComposedTerrain.earth* and *Test-Little Pond (online).earth* to see how they include the imagery.

9.1.4 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 `FeatureModel` elements as you need. A `FeatureModel` 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.

> As with the other data we added earlier in this tutorial, you can include references to XML files containing feature data. The *Example-ComposedTerrain.earth* file includes a variety of features.

Add a Building to the earth File

Add this 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 must disable feature_ indexing for the `FeatureModel` element and specify the shapefile in the `OGRFeatures`

element.

```
  <FeatureModel name="Little Pond Buildings">
    <feature_indexing enabled="false"/>
    <OGRFeatures name="buildings" build_spatial_index="true">
      <url>
../../TerrainData/Terrain/LittlePond/Features/BuildingsP.shp</url>
    </OGRFeatures>
    <layout>
      <tile_size_factor>3</tile_size_factor>
    </layout>
    <styles>
      <style type="text/css">
        HousesColonialGarrison {
        model:               "../../ModelData/Structures/HousesMediumC
lapboardBlue.medf";
        model-heading:       0.0 - feature.properties.rotation;
        model-scale:         1;
        model-placement:     vertex;
        altitude-clamping:   terrain;
        altitude-offset:     0;
        altitude-resolution: 5m;
        }
      </style>
      <selector class="HousesColonialGarrison">
        <query>
          <expr>TYPE = 'HousesColonialGarrison'</expr>
        </query>
      </selector>
    </styles>
    <clustering>false</clustering>
    <lighting>true</lighting>
  </FeatureModel>
```

The `styles` element allows you to specify a `style` for each building you want to display. This example shows only one building. Take a look at *Test-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 that can read DBF files (Figure 66). The TYPE column lists the different types of buildings in the shapefile. The `selector` element looks in the TYPE column.

Figure 66. *BuildingsP.shp*

	A	B	C	D	E
1	LENGTH,N,19,11	WIDTH,N,22,11	HEIGHT,N,22,11	ROTATION,N,22,0	TYPE,C,50
203	8.21	11.02	9	345	HousesColonialGarrison
204	8.21	11.02	9	35	HousesColonialGarrison
205	8.21	11.02	9	320	HousesColonialGarrison
206	8.21	11.02	9	230	HousesColonialGarrison
207	8.21	11.02	9	230	HousesColonialGarrison
208	8.21	11.02	9	250	HousesColonialGarrison
209	8.21	11.02	9	155	HousesColonialGarrison
210	8.21	11.02	9	325	HousesColonialGarrison
211	8.21	11.02	9	250	HousesColonialGarrison
212	8.21	11.02	9	140	HousesColonialGarrison
213	8.21	11.02	9	335	HousesColonialGarrison
214	8.21	11.02	9	65	HousesColonialGarrison
215	8.21	11.02	9	210	HousesColonialGarrison
216	8.21	11.02	9	125	HousesColonialGarrison
217	8.21	11.02	9	335	HousesColonialGarrison
218	8.21	11.02	9	30	HousesColonialGarrison
219	8.21	11.02	9	335	HousesColonialGarrison
220	8.21	11.02	9	0	HousesColonialGarrison
221	8.21	11.02	9	0	HousesColonialGarrison
222	8.21	11.02	9	320	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 specify the shapefile in the `OGRFeatures` 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 definition 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 this code to the earth file after the `FeatureModel` block for the buildings:

```
  <model name="Little Pond Vegetation" driver="feature_custom"
vrfsim:enabled="true" indirect_rendering="false">
    <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="Vegetation" build_spatial_index="true">
      <url>
../../TerrainData/Terrain/LittlePond/Features/VegetationP.shp</url>
    </OGRFeatures>
    <layout>
      <tile_size_factor>3</tile_size_factor>
    </layout>
    <styles>
      <style type="text/css">
        Vegetation {
        model:              feature.properties.veg + "ST";
        model-scale:        1;
        model-placement:    vertex;
        altitude-clamping:  terrain;
        altitude-offset:    0;
```

```
        }
      </style>
    </styles>
    <clustering>true</clustering>
    <lighting>true</lighting>
  </model>
```

9.1.5 Save the earth File

You should save earth files in the *./SharedData/19/latest/TerrainData/TerrainConfiguration* directory. Save the file with a name such as *myLittlePond.earth*.

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.

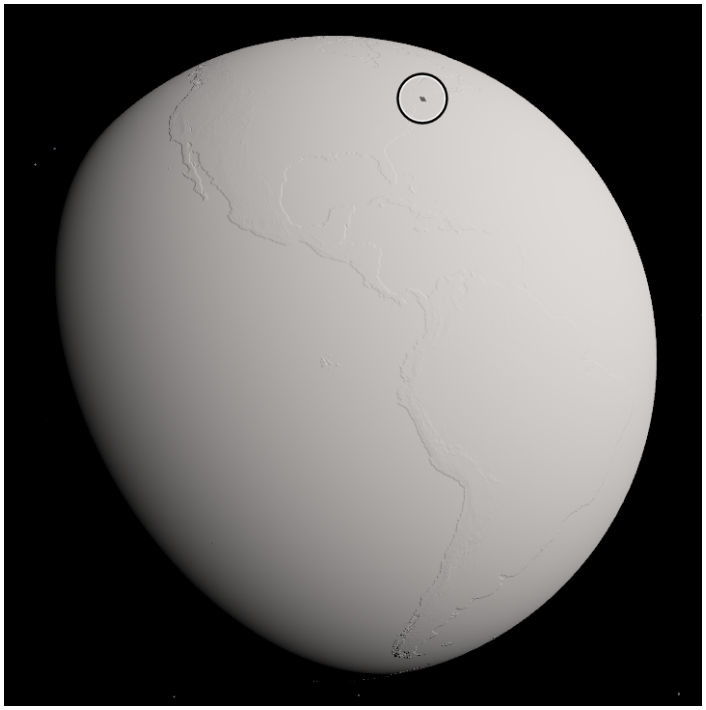
9.1.6 Load the earth File and Save it as an MTF File

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:**

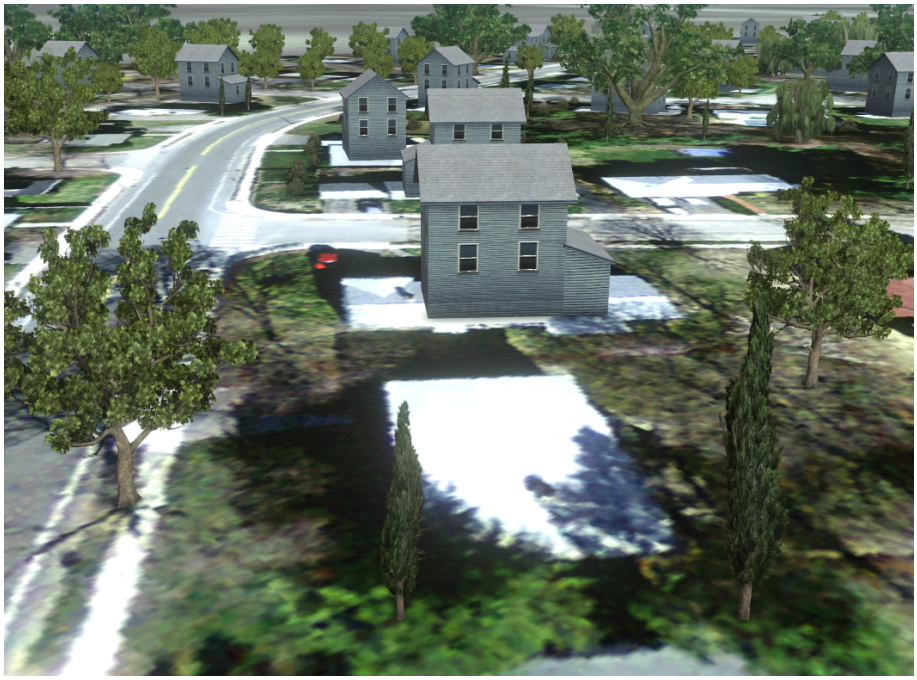
1. Choose **View > Terrain Editor Panel**. The Terrain Editor Panel opens.
2. Select the Add Data tab.
3. Click Terrain Patch. The Add Terrain Patch dialog box opens, as well as the Add Terrain Patch file browser window.
4. Navigate to *./SharedData/19/latest/TerrainData/TerrainConfiguration* and then select *myLittlePond.earth*.
5. Click Open. The file populates the Add Terrain patch dialog box.
6. Click OK, disregarding the warning message. The terrain loads. It is visible as a small patch on the Earth's sphere (Figure 67).

Figure 67. Little Pond terrain



7. Zoom in on the terrain (Figure 68).

Figure 68. Little Pond terrain



8. Choose **File > Save Terrain**. The Save Terrain dialog box opens.
9. Give the terrain a name and then click Save.

9.2. 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 ("9.2.1 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 ("9.2.2 Extract the Props" on the facing page).
3. Optionally, change the model definitions ("9.2.3 Change the Model Definition for a Prop" on page 281).

9.2.1 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.

   ⓘ In VR-Vantage and VR-Engage, you can also choose **File > Add Terrain Patch**and then go directly to step 5.

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/19/latest/TerrainData/Terrain/VR-Village/Geometry/OpenFlight* directory.
6. Select *VR-Village.medf*.
7. Click Open. The Add Terrain Patch dialog box displays the selected patch.
8. Clear the Use Indirect Rendering check box.
9. Click OK. The terrain loads.

9.2.2 Extract the Props

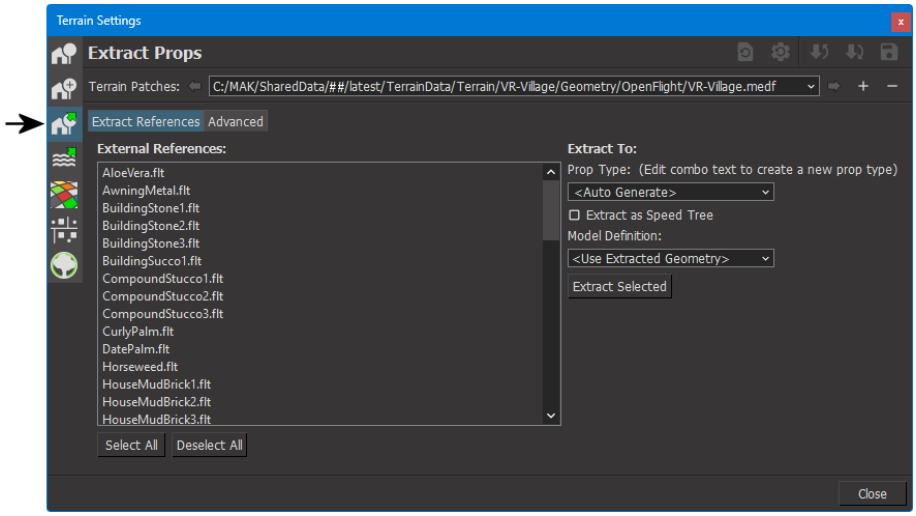
➤ **To extract the props:**

1. Move the observer so that you have a view of the village similar to Figure 69.

Figure 69. 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 70). 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 70. 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. Close or move the Terrain Settings dialog box.
8. Click the trees and buildings on the plateau. You can now select them because they are props, not part of the terrain (Figure 71). 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 71. Selected props

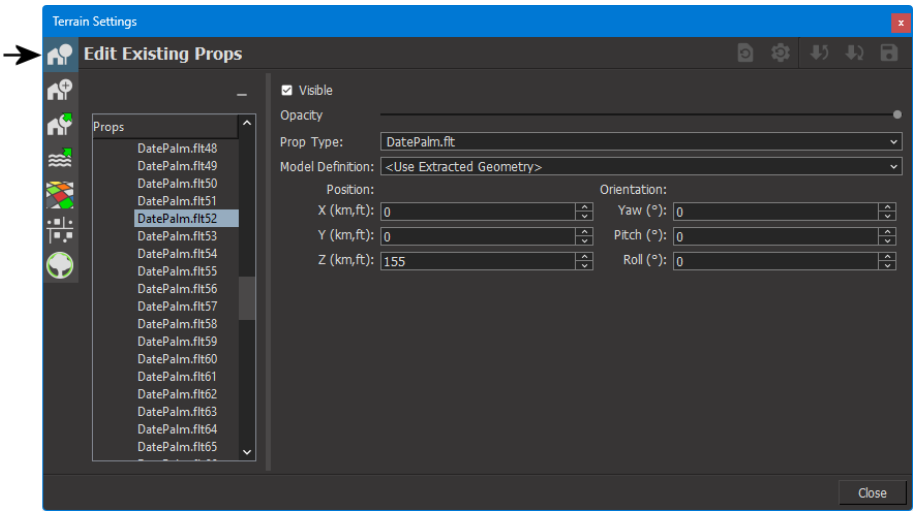
> You may see a message at the top of the Extract Props page that reads "Extraction is disabled due to terrain patch object being rendered with indirect. Please disable indirect for the terrain patch to enable prop extraction." If you do, follow the instructions in "8.5.3 Disabling Indirect Rendering for Terrain" on page 249.

9.2.3 Change the Model Definition for a Prop

➤ **To change the model definition for a prop:**

1. Select one of the trees.
2. If you closed the Terrain Settings dialog box, choose **Settings > Terrain** to reopen it.
3. Select the Edit Existing Props page. The selected prop is highlighted in the list of props (Figure 72).

Figure 72. Edit Existing Props page



4. In the Model Definition box, select a model definition from the list, such as TreesCurlyPalmLargeST. The tree changes to the new model definition (Figure 73).

Figure 73. Original and 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

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 sim engine or in a remote application. The chapters in this section explain how MAK ONE applications manage visual models.

Chapter 10. Introduction to Visual Models

Chapter 11. Schemas

Chapter 12. Model Definitions

Chapter 13. Element Definitions

Chapter 14. Mapping Models and Effects

Chapter 15. Wakes, Cockpits, 2D Icons, and Other Models

Chapter 16. Material Classification for SensorFX

Chapter 17. MAK Specification Notes

Chapter 18. Special Effects (Particle Systems)

10. Introduction to Visual Models

This chapter introduces basic concepts for how MAK ONE applications manage visual models.

10.1. Introduction to Object Modeling 286
10.1.1 Model Sets 287
10.2. Model Formats for 3D Objects, and Effects 288
10.3. OpenFlight and FBX Models 288
10.4. Managing Model and Element Definitions 289
10.4.1 Using the Simulation Object Editor 289
10.5. Visual Model Composability 290
10.6. Indirect Rendering for Entities 294

10.1. Introduction to Object Modeling

The objects that MAK ONE applications can display and 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 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, see "14.1. Introduction to Object Type Mapping" on page 332.)

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, and 2D Icon. MAK ONE applications choose the correct element definition to use based on the model set used by the observer. (For more information about model sets, see "10.1.1 Model Sets" on the facing page.)

Element definitions are created and edited in the Visual Model Editors dialog box. In addition to helping you manage element definitions, this dialog box enables you to create and manage model definitions and schemas. Model definitions specify values for some basic object parameters. Most importantly for most users of MAK ONE applications, 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.

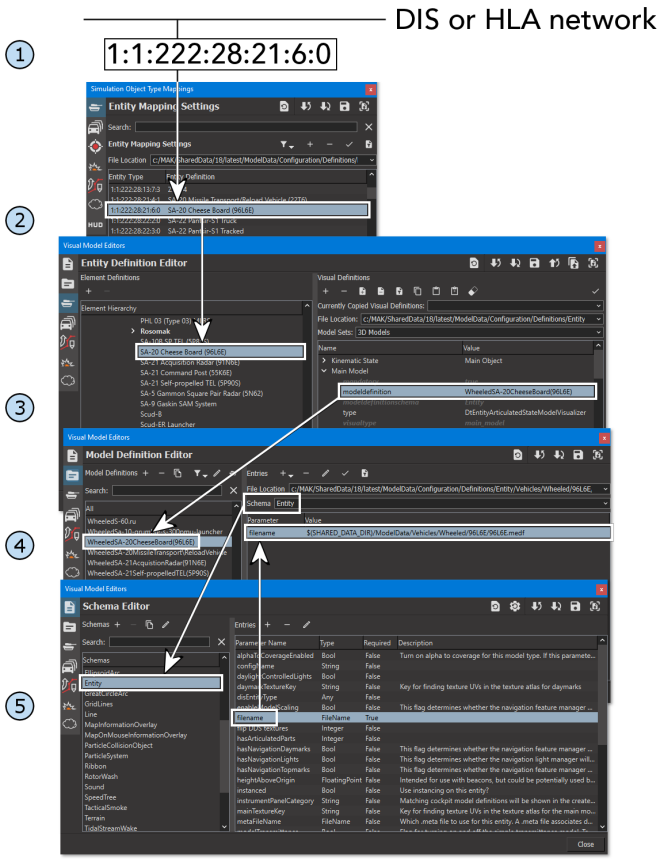
> The chapters in this manual that discuss how to configure visual models do so in the context of using the Visual Models Editor dialog box. We strongly recommend that—whenever possible—VR-Forces users use the Simulation Object Editor to add and configure visual models. The only time that you would need to use the Visual Models Editor dialog box is 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 User Guide* and the chapters in that section.

Schemas define the most basic scene elements, such as an entity, a SpeedTree tree, or a line. When you create a new model definition, you must make sure it uses the correct schema and that it includes all required parameters. Most people creating content for MAK ONE applications will never need to create or edit a schema.

Figure 74 illustrates these relationships using an SA-20 entity. A MAK ONE application receives the object type 1:1:222:28:21:6:0 over the network (callout 1). In the Entity Mapping Settings dialog box, this object type is mapped to the SA-20 Cheese Board

(96L6E) entity definition (2). (An entity definition is a specific type of element definition.) The SA-20 Cheese Board (96L6E) 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 WheeledSA-20CheeseBoard(96L6E). The WheeledSA-20CheeseBoard(96L6E) model definition has a **filename** parameter that specifies *96L6E.medf* as the model to use for entities that use this model definition (4). The WheeledSA-20CheeseBoard(96L6E) model definition has a **filename** parameter because it uses the Entity schema and the Entity schema requires a **filename** (5).

Figure 74. 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.

10.1.1 Model Sets

The model or icon 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. MAK ONE applications allow you to change model sets.)

Each 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 either:

- "Changing the Model Set" in the *VR-Vantage User Guide* or
- "Changing the Model Set" in the *VR-Forces User Guide*

10.2. 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 you can use to encrypt files in MEDF format. You cannot use it to decrypt encrypted files. For more information, see "15.19. Compressing Model Files (medfTool)" on page 381.

You can provide your own models if they are in OpenFlight or FBX format.

10.3. 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 commonly used by that industry, such as 3DS MAX and Maya.

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. The 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, see "17.1.1 The WRM Specification (DIS Notes)" on page 398.)

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, however, 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, see "17.1.2 The MAK FBX Specification" on page 399.

The Gouraud and Lit Gouraud facing drawing modes are legacy rendering modes that are not supported by MAK ONE applications. Use the Lit mode.

10.4. Managing Model and Element Definitions

When a MAK ONE 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.

These chapters in Section III 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.

10.4.1 Using the Simulation Object Editor

If you use VR-Forces and you simply need to add new object types that are similar to existing ones, you should use the Simulation Object Editor (SOE). When you create a simulation object in the SOE, it adds new model definitions and object type mappings automatically. You do 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 specific to the simulation model set (SMS). The visual definition files for the SMSs provided with MAK ONE applications are stored in subdirectories of *./SharedData/19/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*.

10.5. Visual Model Composability

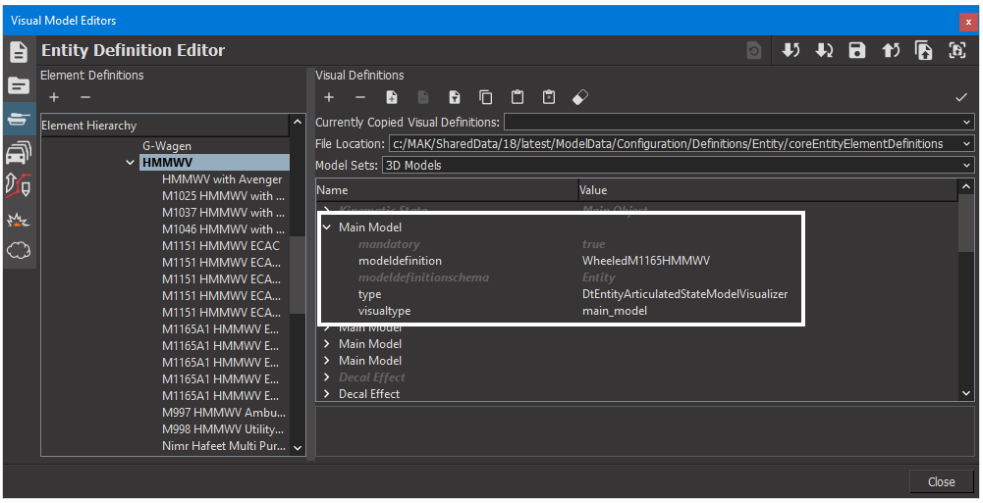
MAK ONE applications compose models for rendering performance, configuration optimization, and asset management efficiency.

- 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.

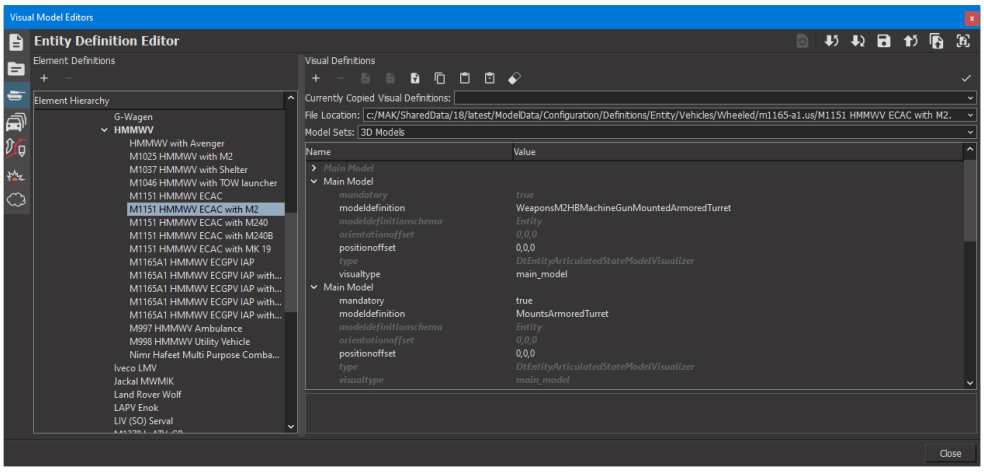
In the Visual Model Editors dialog box, each element definition has a Main Model visual definition, which specifies the model definition that it uses and, as a result, the visual model that displays (Figure 75).

Figure 75. 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 76).

Figure 76. Multiple Main Model definitions



In the HMMWV element definition, we have multiple entities that use the same base model, but different weapon configurations (Figure 77 and Figure 78).

Figure 77. Multiple weapon options

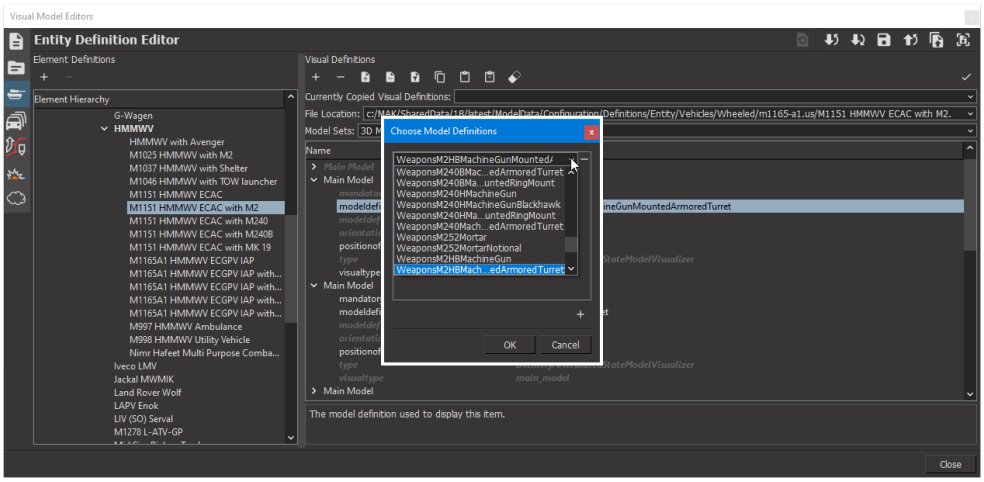
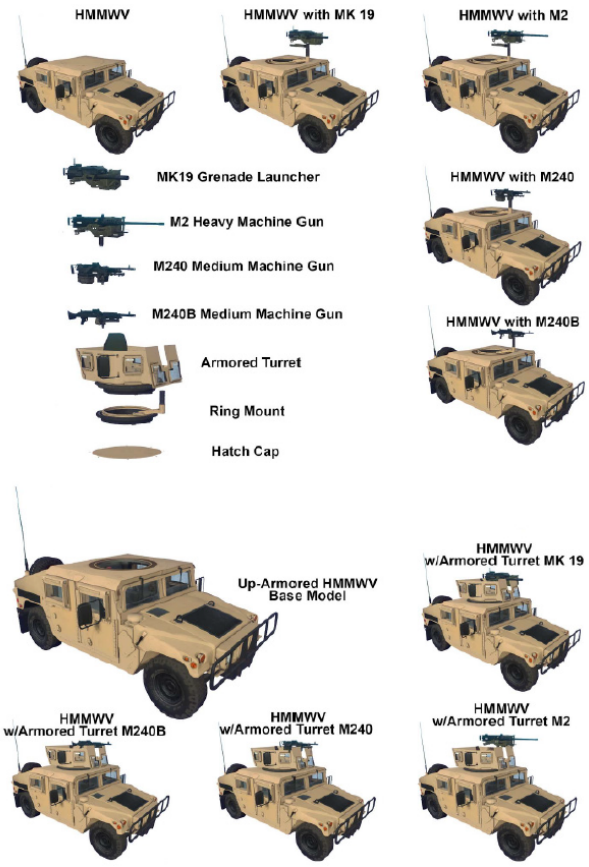


Figure 78. HMWWV models



The weapon configuration is also used on other entities. Figure 79 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 79. Weapon with offset

The M1 Variations are composed in the same fashion to create the M1A2 SEP and SEP V2 (Figure 80).

Figure 80. 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 81 and Figure 82).

Figure 81. HEMTT element definition

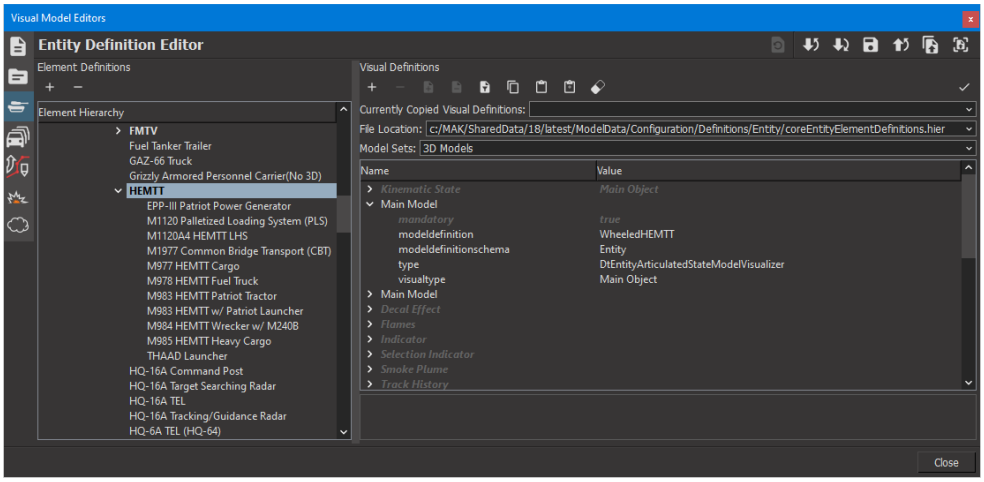
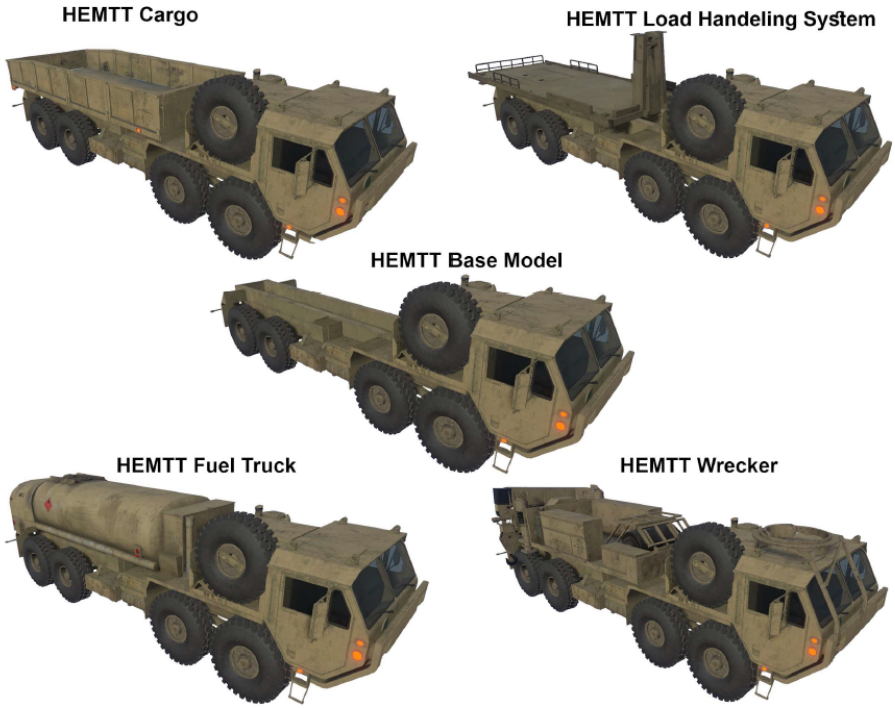


Figure 82. HEMTT models



10.6. 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, see "8.5. Indirect Rendering of Terrain" on page 248. 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.

11. Schemas

In these chapters in Section III, 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.

11.1. Creating and Editing Schemas 296
11.2. Creating Schemas 297
11.3. Saving Schemas 298
    11.3.1 Exporting Selected Schemas 298
11.4. Deleting Schemas 299
11.5. Copying Schemas 299
11.6. Adding a Parameter to a Schema 300
11.7. Editing a Schema Parameter 301
11.8. Deleting a Parameter from a Schema 301

11.1. 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 either "Managing VR-Vantage Settings" in the *VR-Vantage User Guide* or "Managing VR-Forces Settings" in the *VR-Forces User Guide*.
- Most people creating content for MAK ONE applications will never need to create or edit a schema.

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
- Channel3dViewCompass
- ChannelCompass
- CockpitDisplay
- ControlLine
- ControlPoint
- ControlPrism
- Cuboid
- Cylinder
- Decal
- DiGuyCharacter
- EllipsoidArc
- Entity
- GreatCircleArc
- GridLines
- Line
- MapInformationOverlay
- MapOnMouseInformationOverlay
- OverlayArea
- OverlayLine
- OverlaySpiderWeb

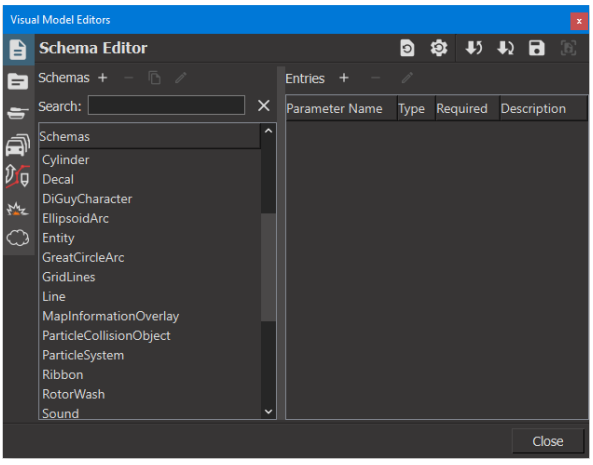
- ParticleCollisionObject
- ParticleSystem
- Ribbon
- RotorWash
- RunwaySelector
- Security
- Sound
- SpeedTree
- TacticalSmoke
- Terrain
- TidalStreamWake
- Tree
- Wake
- Watercraft
- WaterImpact
- WidgetTheme

11.2. Creating Schemas

➤ **To create a schema:**

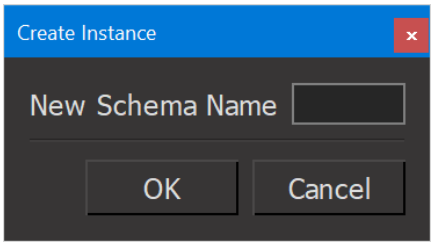
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 83).

Figure 83. Schema Editor page



3. Click the Add button (+) next to the Schemas label. The Create Instance dialog box opens (Figure 84).

Figure 84. Create Instance dialog box



4. Enter a name for the new schema.
5. Click OK.
6. Add parameters as described in "11.6. Adding a Parameter to a Schema" on page 300.

11.3. Saving Schemas

You can save all schemas or 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 83 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 enter the name of a new schema file.
7. Click Save.

11.3.1 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 83 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 enter the name of a new schema file.
7. Click Save.

11.4. 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 83 on page 297).
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.

11.5. 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:**

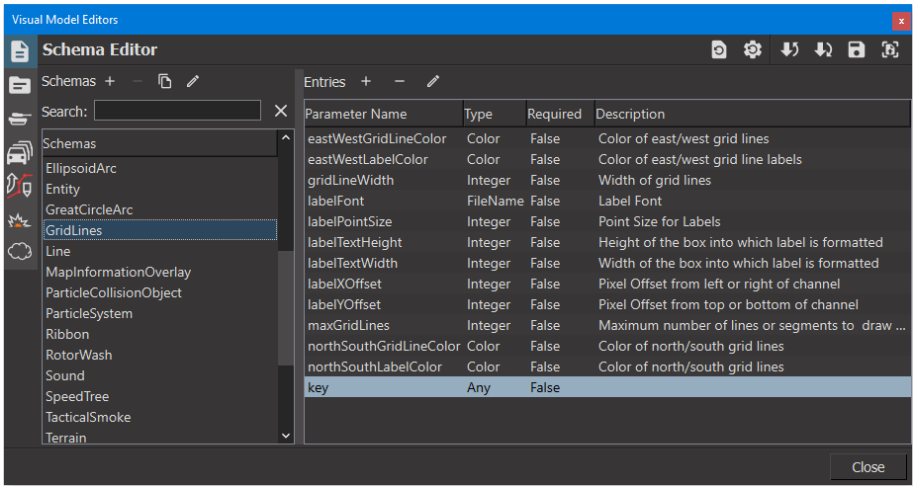
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Schema Editor page (Figure 83 on page 297).
3. Optionally, filter the list of schemas by typing a text string in the Search box. All schemas that include the string are listed.
4. Select the schema that you want to copy.
5. Click the Duplicate button (). The Duplicate Schema dialog box opens.
6. Enter 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 "11.7. Editing a Schema Parameter" on page 301.
10. Save the schemas.

11.6. 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 85).

Figure 85. New schema parameter



6. Double-click "key". The Rename Parameter dialog box opens.
7. Enter 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 should be 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. Enter the descriptive text.
    c. Click OK.

11.7. 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 83 on page 297).
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. Enter the new attribute value.

11.8. 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 83 on page 297).
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.

12. Model Definitions

In these chapters in Section III, we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit model definitions.

12.1. Creating and Editing Model Definitions 304
12.2. Workflow Issues for Model Definitions 305
12.3. Model Definition Files 306
12.4. Creating a Model Definition 307
12.5. Deleting a Model Definition 307
12.6. Copying a Model Definition 308
12.7. Filtering the List of Model Definitions 309
12.8. Editing a Model Definition 309
    12.8.1 Adding Parameters to a Model Definition 309
    12.8.2 Editing a Model Definition Parameter 310
    12.8.3 Deleting Parameters from a Model Definition 311
12.9. Saving Model Definitions 311
    12.9.1 Exporting Selected Model Definitions 311
12.10. Importing Model Definitions 312
    12.10.1 Importing Model Definitions from the Command Line 312

12.1. Creating and Editing Model Definitions

Many of the visual definitions in an element definition include 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, see "11.1. Creating and Editing Schemas" on page 296.)

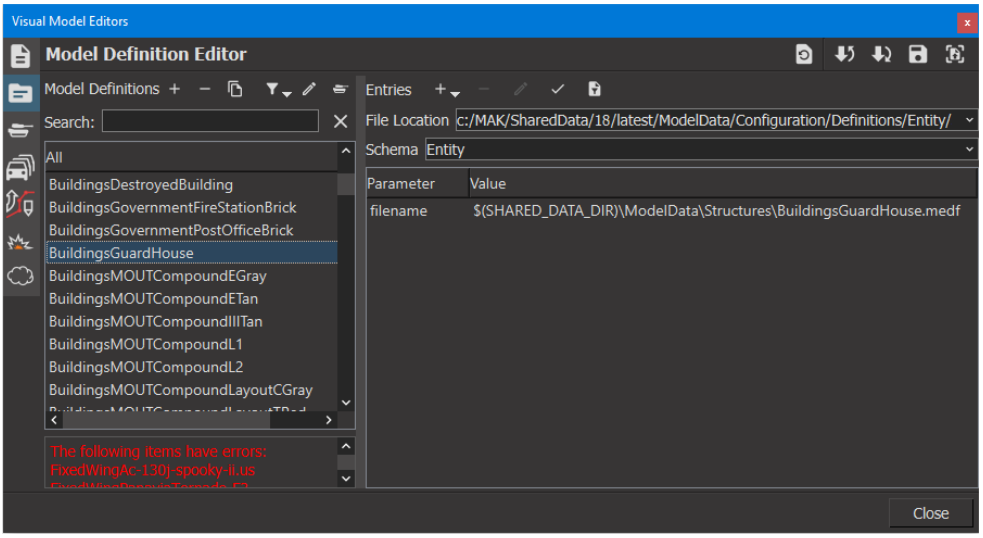
For most MAK ONE application users, the most important thing to know about model definitions is that for 3D models, the model definition specifies the 3D model, such as an OpenFlight file, that is used to display an entity. If you want to add new models to MAK ONE applications, you probably need to create a new model definition for each one.

> ⚠ If you are using VR-Forces, it 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 86).

Figure 86. Model Definition Editor page



The model definitions provided with MAK ONE applications are stored in *./SharedData/19/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 are loaded at startup. For details, see "12.10.1 Importing Model Definitions from the Command Line" on page 312.

- 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 User 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 User Guide*.
- Model definitions are sorted alphabetically. It's good practice to prefix model definitions with a name that indicates its category, such as Buildings or FixedWing.

12.2. Workflow 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 listed in "12.3. Model Definition Files" on the next page.

Before you add, delete, or edit a model definition, consider which of these workflows best meets your need:

- If you want to create new model definitions, the preferred workflow 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 by using the Load and Replace option to load the model definition file you want to edit. (For details, see "12.10. Importing Model Definitions" on page 312.)
  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 "12.10. Importing Model Definitions" on page 312.)
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 workflow for editing an original model definition file.

For more information about how to export model definitions, see "12.9.1 Exporting Selected Model Definitions" on page 311.

> ⚠ Do not use the Export All option if you are editing the combined model definition files that load by default unless you are very certain about how you want to use the resulting file. If you export to a file in the *./SharedData/19/latest/ModelData/Configuration/Definitions/Entity* directory, it will conflict with the other files in that directory.

12.3. Model Definition Files

A default MAK ONE application installation has multiple model definition files. This accommodates 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/19/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/19/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.

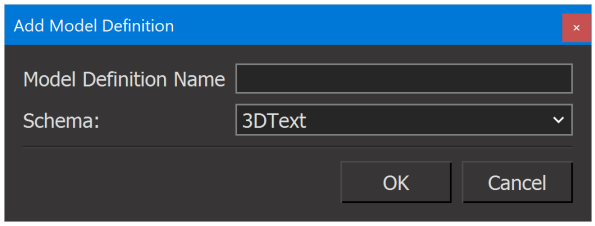
12.4. Creating a Model Definition

> ⚠ 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 86 on page 304).
3. Optionally, filter the model definitions in the list. (For details, see "12.7. Filtering the List of Model Definitions" on page 309.)
4. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens (Figure 87).

Figure 87. Add Model Definition dialog box



5. Enter 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 "12.8.1 Adding Parameters to a Model Definition" on page 309.
8. Save the model definition or export it.

12.5. Deleting a Model Definition

> ⓘ If a model definition is in use, 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 86 on page 304).

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 "12.7. 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.

12.6. Copying a Model Definition

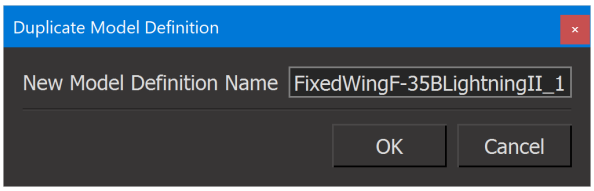
If you want a new model that is a variant of an existing one, it can be easier to copy the existing model and edit it than to create a new model and add all of the parameters.

> 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 86 on page 304).
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 "12.7. 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 88).

Figure 88. Copy model definition



6. Enter a name for the copied version of the model.
7. Click OK. The new model definition is added to the list.
8. Edit the parameters as described in "12.8. Editing a Model Definition" on the facing page.
9. Export or save the model definition.

12.7. 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.

12.8. 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.

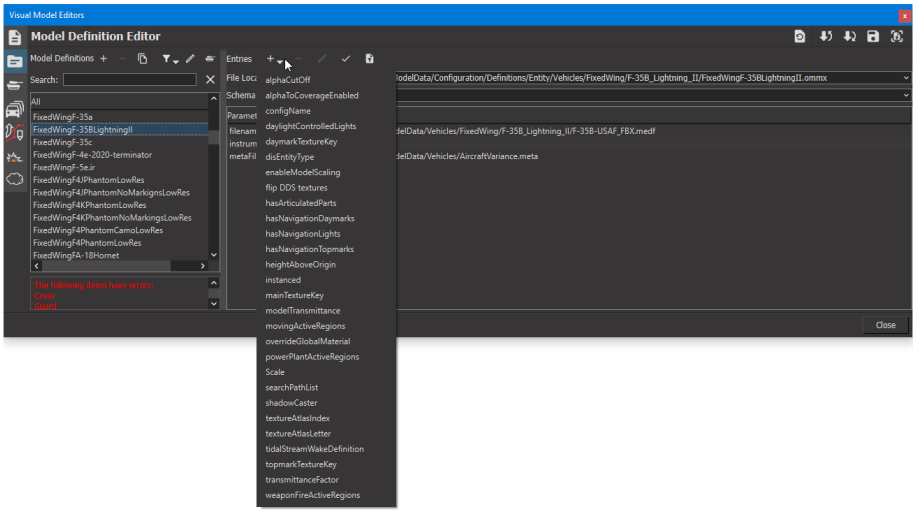
12.8.1 Adding Parameters to a Model Definition

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:**

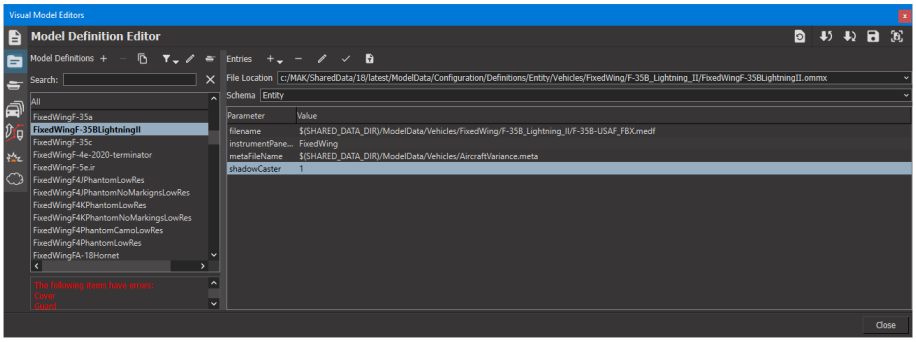
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 86 on page 304).
3. Optionally, filter the model definitions list. (For details about filtering, see "12.7. Filtering the List of Model Definitions" above.)
4. Select the model definition for which you want to add a parameter.
5. Click the Add Parameter button (+▼). A list of parameters displays (Figure 89).

Figure 89. 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 90).

Figure 90. Editable parameter



8. Select or enter a value for the parameter.
9. Click away from the field.
10. When you have finished editing the model definition, save or export it.

12.8.2 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 86 on page 304).
3. Optionally, filter the model definitions list. (For details about filtering, see "12.7. Filtering the List of Model Definitions" on the previous page.)
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 90 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.

12.8.3 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 86 on page 304).
3. Optionally, filter the model definitions list. (For details about filtering, see "12.7. Filtering the List of Model Definitions" on page 309.)
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.

12.9. 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 a MAK 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 86 on page 304).
3. Click the Save button (💾). A confirmation prompt displays.
4. Click Yes.

12.9.1 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 86 on page 304).
3. Optionally, filter the model definitions in the list. (For details, see "12.7. Filtering the List of Model Definitions" on page 309.)

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 enter a name for the model definition file.
7. Click Save.

12.10. 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 either "Managing VR-Vantage Settings" in the *VR-Vantage User Guide* or "Managing VR-Forces Settings" in the *VR-Forces User 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 86 on page 304).
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 86 on page 304).
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.

12.10.1 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 merges only 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/19/latest/ModelData/Configuration/Definitions* directory. To do this, start the application with the `--noAppDataModelDefs` command-line option.

13. Element Definitions

In these chapters in Section III, we describe the interplay of schemas, model definitions, simulation object definitions, and model mapping. This chapter explains how to create and edit element definitions.

13.1. Creating and Editing Element Definitions 314
13.2. How Entity, Unit, Spot Report, and Tactical Graphics Element Definitions Are Organized 314
13.2.1 Leaf File Descriptions 315
13.3. Workflow Issues for Creating and Editing Element Definitions 316
13.4. Creating an Element Definition 317
13.5. Filtering the List of Element Definitions 318
13.6. Editing an Element Definition 318
13.6.1 Adding a Visual Definition 319
13.6.2 Editing a Visual Definition 319
13.6.3 Adding an Attribute to a Visual Definition 321
13.6.4 Deleting an Attribute from a Visual Definition 322
13.6.5 Copying and Pasting Visual Definitions 323
13.7. Jumping to a Model Definition 323
13.8. Changing an Element Definition's File Location 324
13.9. Deleting an Element Definition 324
13.10. Saving Element Definitions 325
13.10.1 Saving Entity, Unit, Spot Report, or Tactical Graphics Element Definitions 326
13.10.2 Saving Interaction and Smoke Element Definitions to .elem Files 326
13.11. Importing Element Definitions 327
13.11.1 Importing a Hierarchy File 328
13.11.2 Importing Leaf Files 328
13.11.3 Importing Element Definitions from the Command Line 328

13.1. 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. Refer to the VR-Forces user documentation for details.

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.

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 User Guide*.

13.2. 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*)—depending on the element definition's intended use. 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 hierarchy element definitions, you must save the changes to the hierarchy file associated with that element definition. 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.

13.2.1 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, the MAK ONE computer generated forces (CGF) application. By default, a MAK ONE application loads all of the element definitions, allowing it to render whatever simulation objects VR-Forces sends. The element definitions are organized as follows:

- *./SharedData/19/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/19/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/19/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/19/latest/ModelData/Configuration/Definitions/Unit*.
- *./data/Configuration/Definitions/Entity* to *./SharedData/19/latest/ModelData/Configuration/Definitions/Entity*.
- *./data/Configuration/Definitions/TacticalGraphics* to *./SharedData/19/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.

13.3. Workflow 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 for the results you want and avoid problems.

- If you have created new element definitions, the preferred workflow 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.

13.4. Creating an Element Definition

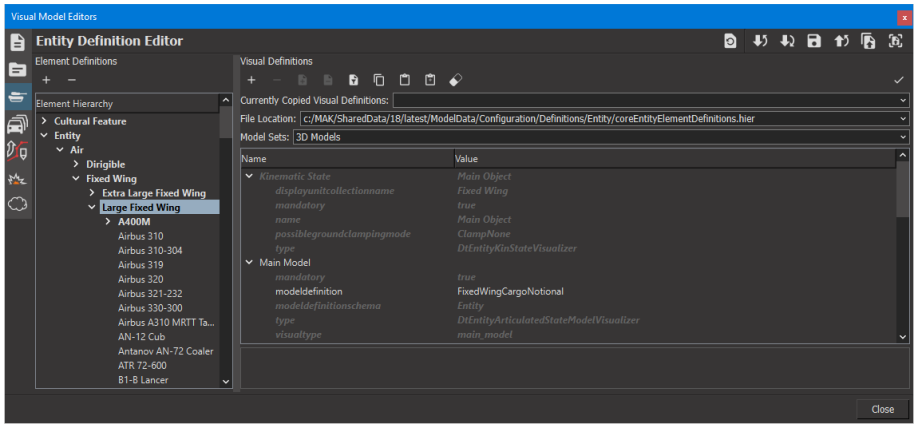
You can create new element definitions for entity, unit, tactical graphics, smoke, spot reports, and interactions. You can export them to *.hier* or *.leaf* files. Use a new or existing *.hier* file when creating an element definition that will have child element definitions. Use a *.leaf* file when creating a unique element 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.
- You cannot edit an element definition name after you create the definition, therefore you should be sure to avoid typing errors.

➤ **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 91).

Figure 91. Entity Definition Editor page



4. Click the Add button (+) under Element Definitions. The Create Element Definition dialog box opens.
5. Enter a name for the element definition. (Once you create the element definition, you cannot edit the name. Be sure that the name is what you want.)

6. Click OK. The Get Filename for Element Definition dialog box opens.
7. You can add to an existing definition or save as a new one:
   - To save as a new definition, enter a name for the new definition and choose whether to save it as a *hier* or *leaf* file.
   - To associate the new element with an existing file, select the file.
8. Click Save.
   - If you are saving to a new file, you return to the Visual Model Editors dialog box.
   - If you are associating with an existing file, the application prompts you to confirm that you want to "replace the file". You are not actually overwriting the file. You are simply adding your definition to the existing file.
9. Optionally, edit the visual definitions inherited by the element definition. For details, see "13.6.2 Editing a Visual Definition" on the facing page.
10. Optionally, add a visual definition to the element. For details, see "13.6.1 Adding a Visual Definition" on the facing page.
11. Save or export the element definition.
    - Choose Save All to save all of the current changes to the element definitions.
    - Use Export to save only the new element definition.

13.5. 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 element definitions by source file:**

1. In the File Location list, select the source file that you want to filter by. (To show all element definitions, select All.)
2. Click the Filter button ( ) below the Visual Definitions label.

13.6. 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.

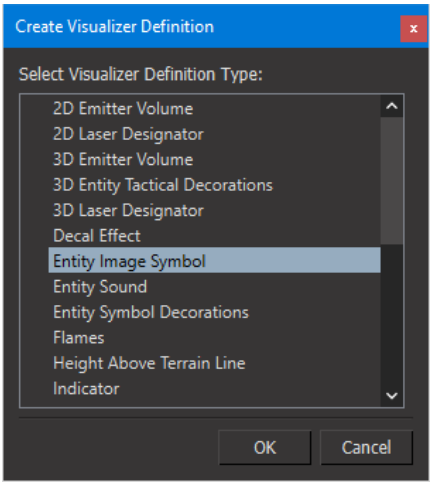
> ⚠ MAK ONE applications do not save changes to element definitions automatically. If you want to keep changes, you must export them to a file.

13.6.1 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 91 on page 317).
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 92. 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, see "13.6.2 Editing a Visual Definition" below.
9. When you have finished editing the element definition, save or export it.

13.6.2 Editing a Visual Definition

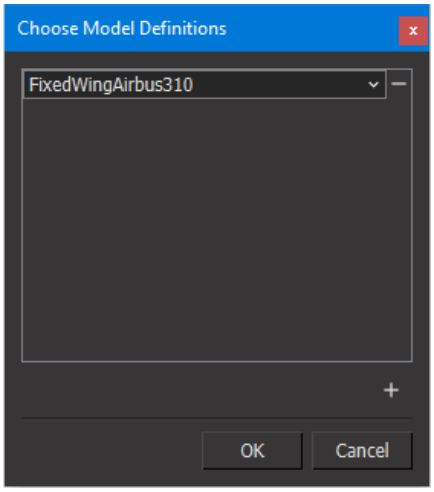
Visual definitions and their attributes are displayed either as italicized text or plain text (Figure 93). Italicized text indicates that the visual definition or attribute is inherited from a parent element. Visual definitions and attributes that are not italicized are particular to that element. (These entries could be inherited by child elements.)

Figure 93. 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 91 on page 317).
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 displays, a dialog box opens (Figure 94), or the field becomes editable.

Figure 94. Choose Model Definitions dialog box



5. For a list, select the new value. For an editable field, enter 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 finish editing the element definition, save or export it.

Mapping to Multiple Model Definitions or Schema

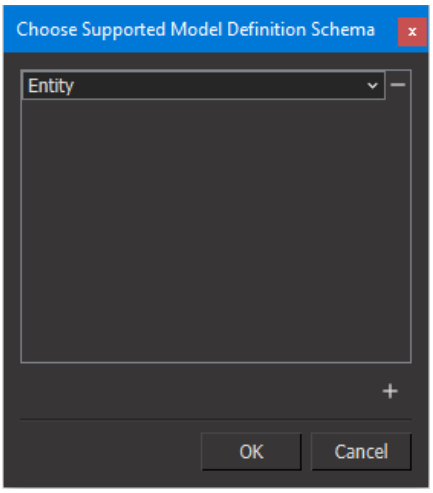
There may be cases where you want to map a visual definition to multiple model definitions or multiple schema. If you do this, the application randomly assigns the model definition or schema to instances of the object.

➤ **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 91 on page 317).
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 95).

Figure 95. 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.

13.6.3 Adding an Attribute to a Visual Definition

Each visual definition has a set of attributes. Some of them may be optional and, as a result, they are not specified as part of the factory definitions. You can add optional attributes to a visual definition.

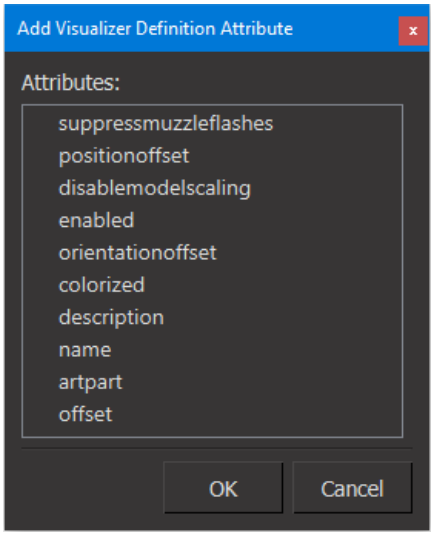
> ⚠ If you add an attribute to an element definition that is part of the element definition hierarchy, for example Entity > Land > Tracked, you must export all of the element definitions that inherit attributes from that definition. As a practical matter, this usually means exporting all of the definitions 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 91 on page 317).
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 96).

Figure 96. 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.

13.6.4 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 91 on page 317).
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.

13.6.5 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 91 on page 317).
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 91 on page 317).
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 91 on page 317).
3. Click the Clear all Visualizer Definitions button ( ).

13.7. 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 91 on page 317).
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.

13.8. 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 Entity Definition Editor page (Figure 91 on page 317).
3. Optionally, filter the model definitions list. (For details about filtering, see "12.7. Filtering the List of Model Definitions" on page 309.)
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.

13.9. Deleting an Element Definition

Deleting element definitions does not delete files. The node is simply deleted and, when you save, it becomes "empty". Subsequent loads ignore it.

The files for any nodes under a deleted hierarchy node remain unchanged, but they are not used on subsequent loads because the parent node is not found. If you want to remove the files, you must do so manually by deleting the files outside of the MAK ONE application.

⚠ MAK ONE applications do 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 referenced in an entity type mapping.

➤ **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 91 on page 317).
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 button ( ).

13.10. 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*). You can save individual interactions, however, which means it's possible to have more than one *.elem* file in the definitions directory for element types. In that case, the 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 are not saved automatically. You must explicitly save the changes. Options for saving changes are:

- Save All. Saves all hierarchy and leaf files that you modified to the file locations specified in the File Location list for each element definition.
- Save All Hierarchies. Saves all hierarchy files that you modified to the file locations specified in the File Location list for each element definition.
- Save All Leaves. Saves all leaf files to the file locations specified in the File Location list for each element definition.
- Export Selected Node (.hier/.leaf). Saves the selected element definition to a user-specified file.

13.10.1 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 save all 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 hierarchy files:**

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 Hierarchies 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 Save All Leaves button (). A confirmation prompt displays.
4. Click Yes.

➤ **To export a selected definition:**

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 definition you want to export.
4. Click the Export Selected Node button (). The Export Element Definitions dialog box opens.
5. Select the *.hier* or *.leaf* file you want to export to or enter the name of a new file.
6. Click Save.

13.10.2 Saving Interaction and Smoke Element Definitions to *.elem* Files

When you save interaction element definitions or tactical smoke element definitions, they are 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.

13.11. Importing Element Definitions

You can import hierarchy element definitions and replace the loaded element definitions. You can import element definitions and merge them with the loaded element definitions. (Replace works only for hierarchy files, not leaf files.)

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 previously loaded hierarchy element definitions that match (by name) element definitions in the imported file.

All visual definition inheritance by descendant element definitions is predicated on you creating them as part of a specific hierarchy which uses UUID strings to maintain that relationship. Therefore, the visualizer definitions of imported elements that were created with a different hierarchy will not inherit attributes from the existing hierarchy nodes.

If you add a new leaf node and maybe a hierarchy node and then export them to a file, another user can then import the file and use those element definitions assuming both you are both using the same MAK ONE data and hierarchies. If the other user deleted the hierarchy shipped with data package, however, and then built their own hierarchy file that uses some of the same key hierarchy names (such as Entity, Vehicle, Fixed Wing, and so on), that would not work. When they imported the element definitions you created, the element hierarchy would work (things parented to Entity would still parent to Entity on import) but the visualizer definitions would not be inherited by nodes from the foreign element hierarchy.

13.11.1 Importing a Hierarchy File

If you are using *.elem* files, you can replace the current hierarchy with a different one by importing a hierarchy file in the legacy *.elem* format.

When you are using *.hier* or *.leaf* files, because there can be multiple *.hier* files, the Import and Replace function allows you to load a *.hier* file. If it contains new hierarchy element definitions, they are added. If it contains existing element definitions (by name), they are replaced. Replaced element definitions remain associated with the files they were originally read from and saved with them. New element definitions remain associated with the file they were imported from. (Use the method described in "13.8. Changing an Element Definition's File Location" on page 324 to associate the new ones with other existing hierarchy files or export them to a new location where they will be found.)

➤ **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.

13.11.2 Importing Leaf Files

You can import and merge element definitions from *.leaf* files or legacy element definition (*.elem*) files. When you choose to import and merge a *.leaf* file it adds new element definitions (by name) and ignores ones that already exist.

> 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.

13.11.3 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 these command-line options:

- `--entityDefsDir` *`directory`* merges in all *.hier* and *.leaf* files in the specified directory. The directory must be *./SharedData/19/latest/ModelData/Configuration/Definintions/Entity*, *./SharedData/19/latest/ModelData/Configuration/Definintions/Unit*, *./SharedData/19/latest/ModelData/Configuration/Definintions/TacticalGraphics*, or a subordinate to one of them.
- `--mergeHierarchyEntityDef` *`filename`* imports the specified entity *.hier* file.
- `--mergeLeafEntityDef` *`filename`* merges in the specified entity *.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/19/latest/ModelData/Configuration/Definintions*. You can also specify that the element definitions in *./SharedData/19/latest/ModelData/Configuration/Definintions* not be loaded. In that case, only those files that you specify on the command line load. If none are specified, no element definitions load.

To specify that the element definitions in *./SharedData/19/latest/ModelData/Configuration/Definintions* not be loaded, start MAK ONE applications with the `--noAppDataEntityDefs`, `--noAppDataUnitDefs`, or `--noAppDataTacticalGraphicsDefs` command-line options.

14. Mapping Models and Effects

This chapter explains how to map element definitions to DIS and HLA object types.

14.1. Introduction to Object Type Mapping 332
14.2. Workflow Issues for Object Type Mappings 332
14.3. Adding an Object Type Mapping 333
14.4. Editing an Object Type Mapping 334
14.5. Filtering the Element Definition List 335
14.6. Changing an Object Type Mapping's File Location 336
14.7. Jumping to an Element Definition 336
14.8. Deleting an Object Type Mapping 337
14.9. Saving Object Type Mappings 337
    14.9.1 Exporting Selected Object Type Mappings 337
14.10. Importing Object Type Mappings 338
14.11. Starting Without Default Object Type Mappings 338

14.1. Introduction to Object Type Mapping

To visualize a scene element (entity, unit, detonation, and so on), MAK ONE applications must map their 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.

14.2. Workflow Issues for Object Type Mappings

A default MAK ONE application 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/19/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 these 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 these workflows best meets your need:

- If you have created new model mappings, the preferred workflow 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.

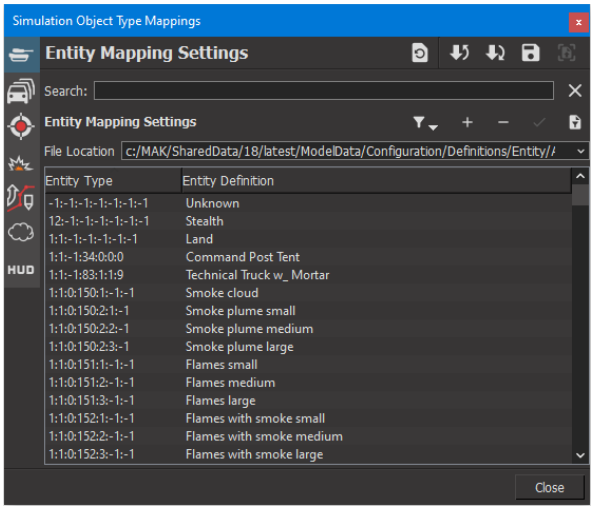
14.3. Adding an Object Type Mapping

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:**

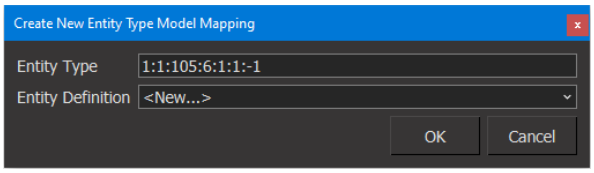
1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the Entity Mapping Settings page (Figure 97). The page shows a list of object type enumerations in the Entity Type column. The entity definition mapped to the object type is listed in the Entity Definition column.

Figure 97. Entity Mapping Settings page



3. Click the Add button (+). The Create New Entity Type Model Mapping dialog box opens (Figure 98). It lists a default object type enumeration.

Figure 98. Create New Entity Type Model Mapping dialog box



4. In the Entity Type text box, enter 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.

14.4. Editing an Object Type Mapping

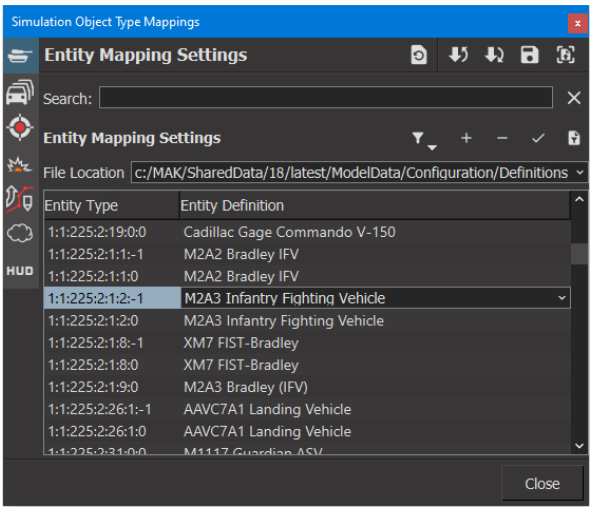
This procedure describes how to edit an object type mapping for a simulation object. The procedure is the same for any page on the Simulation Object Type Mappings dialog box.

➤ **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 97 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, see "12.7. Filtering the List of Model Definitions" on page 309.)
   b. In the Entity Definition column, click the element definition to make it an editable list (Figure 99).
   c. Select the new element definition from the list.
   d. Click someplace else in the window.

Figure 99. Editable entity definition



7. Save the model mapping to a new or existing file.

14.5. 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 99 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 only 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, enter 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 ().

14.6. 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.

14.7. 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.

14.8. 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 97 on page 334).
3. Use the Import and Replace procedure to load the object type mapping file that you want to delete a mapping from. (For details, see "14.10. 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.

14.9. Saving Object Type Mappings

New and changed simulation object type mappings are not 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 are saved.

14.9.1 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 the mapping was originally a part of to ensure that it overrides 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.

14.10. Importing Object Type Mappings

You can import simulation object type mapping files. An imported file replaces or merges with the previously loaded mappings.

- Replace. Any default mappings are replaced by those in your file, overwriting those matching mappings. If you have multiple models mapped to an object enumeration, all of the mappings are replaced by the new mappings.
- Merge. Mappings in your file are added to the default mappings and will exist alongside those.

> If you want more control over what files are merged or overwritten, use command-line options when you start the application. See the product's user guide or help to learn about the command-line options for your MAK ONE application.

➤ **To import simulation object type mappings:**

1. Choose **Settings > Simulation Object Type Mappings**. The Simulation Object Type Mappings dialog box opens.
2. Select the page for the type of simulation object whose mappings you want to import.
3. To replace the current set of mappings, click the Import and Replace button (). The Import Mapping Settings dialog box opens.

   To merge mappings, click the Import and Merge button (). The Merge Mapping Settings dialog box opens.
4. Select the settings file you want to import.
5. Click Open.

14.11. 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/19/latest/ModelData/Configuration/Definitions* or *./appData/settings/<application>*. To do this, start the MAK ONE application with the `--noAppDataEntityTypeMap`, `--noAppDataUnitTypeMap`, or `--noAppDataTacticalGraphicsTypeMap` command-line option.

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

The chapters in Section III provides a general introduction to model schemas, model definitions, and element definitions. This chapter describes how to configure several specific types of visual models.

15.1. Configuring 2D Icons 341
15.1.1 Editing Font-Based 2D Icons 341
15.1.2 Using Images for 2D Icons 342
15.1.3 Adding Images for Simulation Object Icons 347
15.2. Adding New Cockpit Display Models 347
15.2.1 Installing a Cockpit DLL 348
15.2.2 Creating a Model Definition for a Cockpit 349
15.3. Adding Object Models as Props 351
15.3.1 Adding a Category to the Props Palette 351
15.4. Clutter Entities 352
15.4.1 Configuring Aircraft Clutter Entities 352
15.4.2 Configuring Ship Clutter Entities 353
15.5. Configuring Light Points 355
15.5.1 Configuring Light Point Type Models 357
15.5.2 Configuring Light Point Group Models 357
15.5.3 Adding a New Light Point Type 358
15.6. Aerodrome and Vehicle Lighting 358
15.6.1 Best Practices for Lightpoints 358
15.6.2 Airfield Lighting 359
15.6.3 Vehicle Lighting Switches 360
15.7. Configuring a Light Source for a Specific Sensor Mode 361
15.8. Configuring Wakes in the Visual Model Editor 361
15.8.1 Configuring Tidal Stream Wakes 364
15.8.2 Displaying Propeller Wash 364
15.9. Configuring Persistent Tracks in Model Definitions 365
15.10. Configuring Floating Lifeforms 366

15.11. Adding Wind-based Controls to Models 367
15.12. Using Texture Atlases to Skin Models 368
15.12.1 Configuring Texture Mapping in the Model Definition 369
15.12.2 Metafiles for Texture Mapping 369
15.13. Adding Dynamically Changeable Markings to Models 371
15.13.1 Adding @dis annotations to the Model 372
15.13.2 The Metafile for Markings 373
15.13.3 The Texture Mapping File 375
15.14. Adding a Depth Water Mask to a Model 375
15.15. Flipping DDS Textures for a Model 376
15.16. Configuring SpeedTree Trees 377
15.16.1 Randomizing the Size of SpeedTrees 379
15.16.2 Displaying SpeedTrees in Planar Reflections 380
15.17. Best Practices for Creating Models 380
15.18. Disabling Indirect Rendering for an OpenFlight Model 381
15.19. Compressing Model Files (medfTool) 381

15.1. 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 User Guide*.

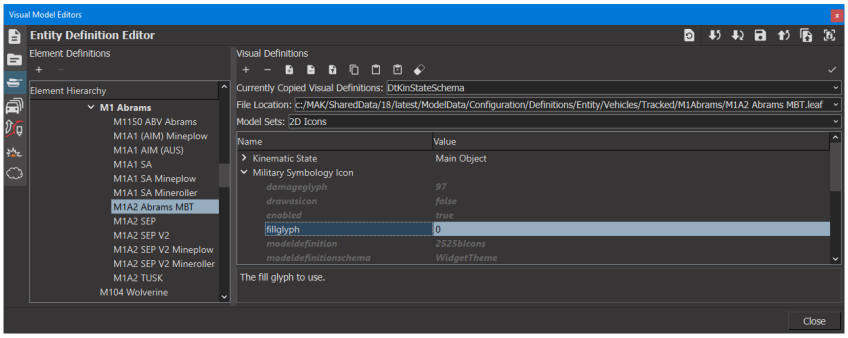
15.1.1 Editing Font-Based 2D Icons

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:**

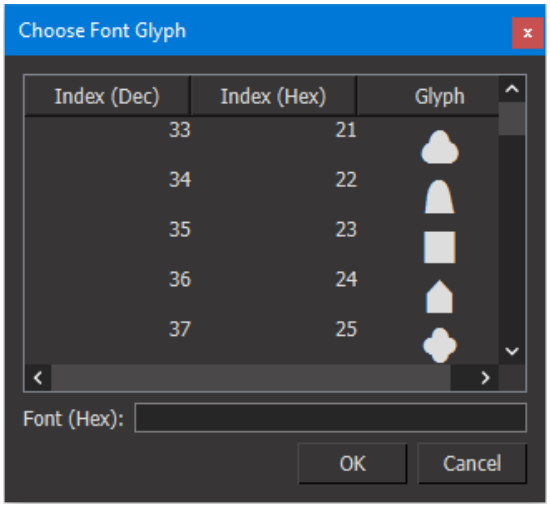
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 91 on page 317).
3. Select the simulation object whose icon you want to edit.
4. In the Model Sets list, select 2D Icons.
5. To change the fill glyph:
   a. Under the Military Symbology Icon visual definition, select the **fillglyph** attribute (Figure 100).

Figure 100. Military Symbol Icon visual definition



   b. Click the value in the Value column. The Choose Font Glyph dialog box opens (Figure 101).

Figure 101. 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.

15.1.2 Using Images for 2D Icons

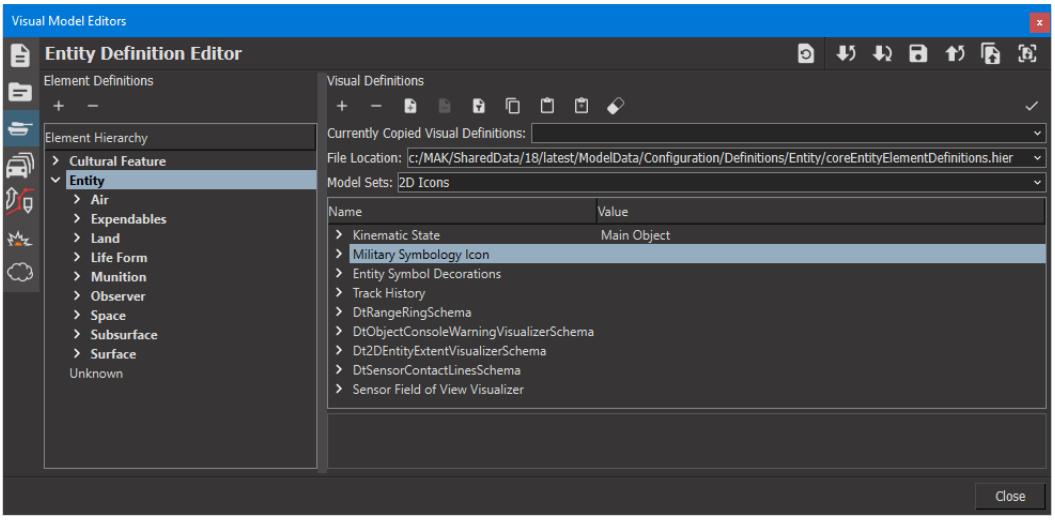
By default, MAK ONE applications use TrueType fonts for 2D icons. You can replace the font-based icons with images using the Entity Image Symbol visual definition.

> 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:**

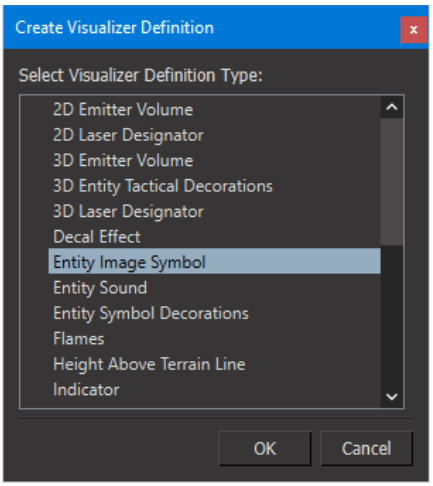
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page (Figure 91 on page 317). (If you want to change the icon for a unit, select the Unit Definition Editor page. To change the icon for a waypoint, select the Tactical Graphics 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 102). (Some tactical graphics do not have this visual definition.)

Figure 102. Entity element definition with Military Symbology Icon visual definition



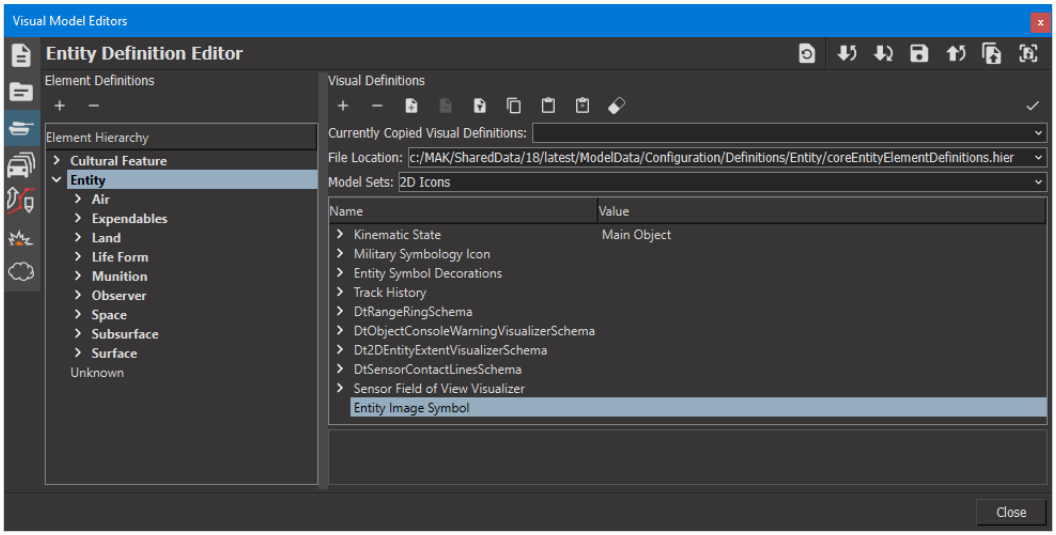
6. Click the Delete button (−). The visual definition is deleted.
7. Click the Add button (+). The Create Visualizer Definition window opens. It displays a list of visual definitions (Figure 103).

Figure 103. Create Visualizer Definition window



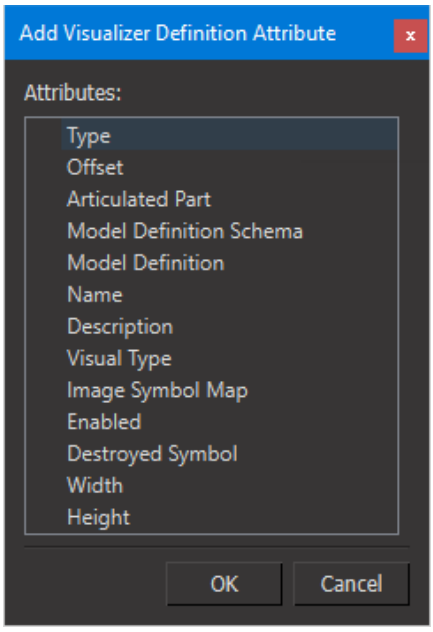
8. Select Entity Image Symbol. (If you are configuring images for units, select Aggregate Image Symbol; for tactical graphics, select Tactical Graphic Image Symbol.)
9. Click OK.
10. In the Visual Definitions list, select Entity Image Symbol (Figure 104).

Figure 104. Entity element definition with Entity Image Symbol visual definition



11. Click the Add Attribute button (). The Add Visualizer Definition Attribute window opens (Figure 105).

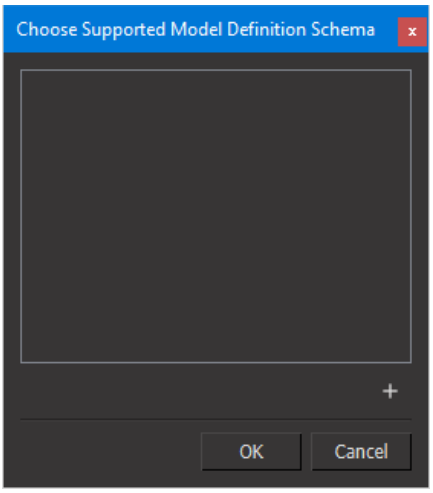
Figure 105. Add Visualizer Definition Attribute window



12. In the Attributes list, select **Type** and then click OK.
13. Select Entity Image Symbol Visualizer from the list. This assigns the DtEntityImageSymbolVisualizer value for the attribute.
14. Click the Add Attribute button.

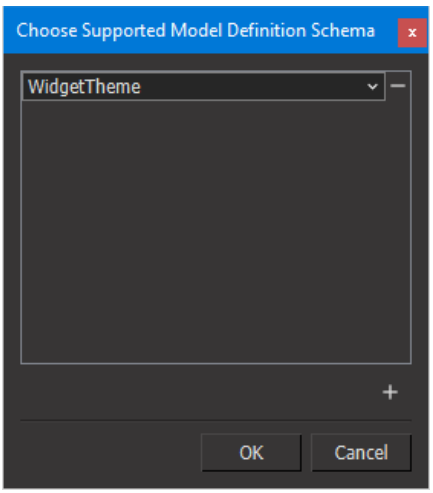
15. In the Attributes list, select **Model Definition Schema** and then click OK. The Choose Supported Model Definition Schema dialog box opens (Figure 106).

Figure 106. Choose Supported Model Definition Schema dialog box



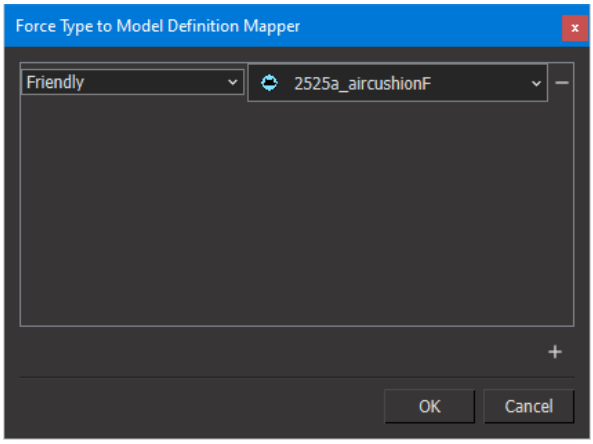
16. Click the Add button to add a list box to the dialog box.
17. In the list, select **WidgetTheme** (Figure 107).

Figure 107. Completed Choose Supported Model Definition Schema dialog box



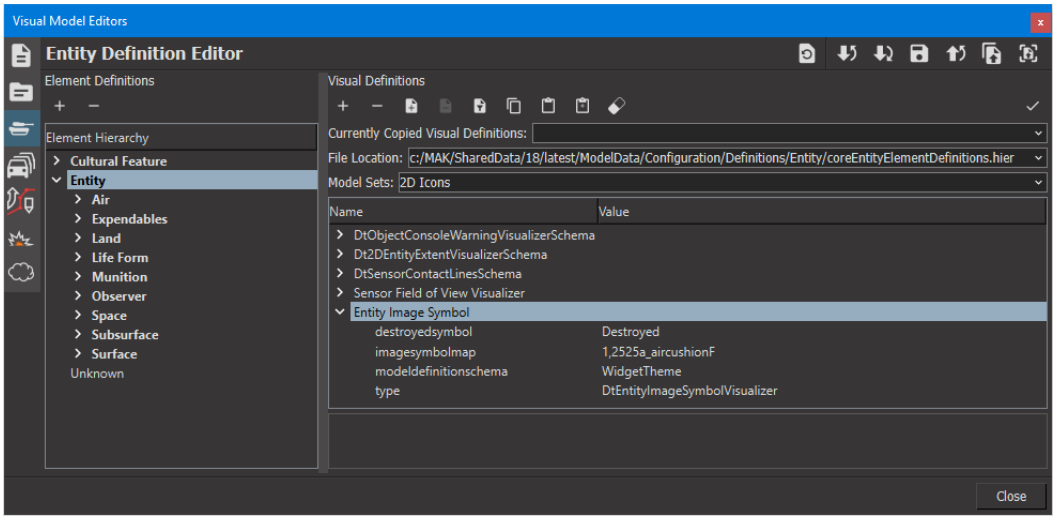
18. Click OK.
19. Click the Add Attribute button.
20. In the attribute list, select **Image Symbol Map** and then click OK. The Force Type to Model Definition Mapper dialog box opens.
21. Click the Add button. A list is added to the dialog box. It lists the forces for which you can map images and the images that you can apply to a particular force.
22. Select a force and an image from the lists (Figure 108).

Figure 108. 2D Symbol Image Mapper dialog box with force chosen



23. Add an entry for each force that you want to represent with an image.
24. Click OK.
25. Click the Add Attribute button.
26. In the attribute list, select **Destroyed Symbol** and then click OK. The **destroyedsymbol** attribute is added to the visual definition.
27. In the **destroyedsymbol** list, select Destroyed. The completed visual definition should look similar to Figure 109.

Figure 109. Entity Image Symbol visual definition



28. 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.

15.1.3 Adding Images for Simulation Object Icons

MAK ONE applications include a representative set of MIL-STD 2525a images. They are in *./SharedData/19/latest/images*. To use your own images, add a new model definition or change 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:**

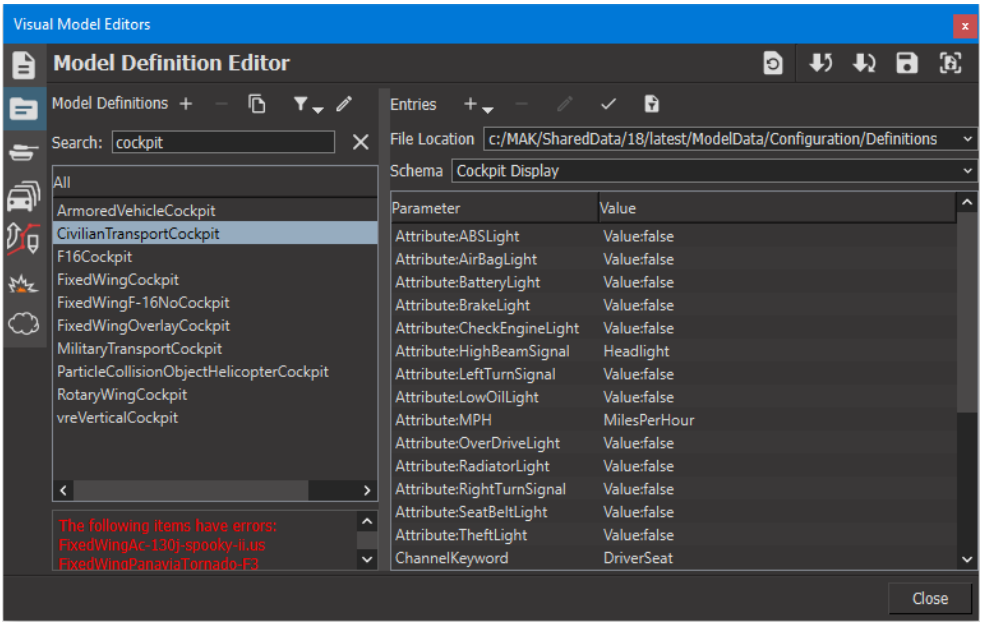
1. Copy the new images to *./SharedData/19/latest/images*.
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 86 on page 304).
4. In the Schema list, select WidgetTheme.
5. If you want to change the image used for 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 "12.1. Creating and Editing Model Definitions" on page 304. 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.

15.2. 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 110 shows the model definition for a cockpit.

Figure 110. 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 "14.3. Adding an Object Type Mapping" on page 333.

15.2.1 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/19/latest/Huds/platform* (where platform is the appropriate folder) and then create a model definition for it.

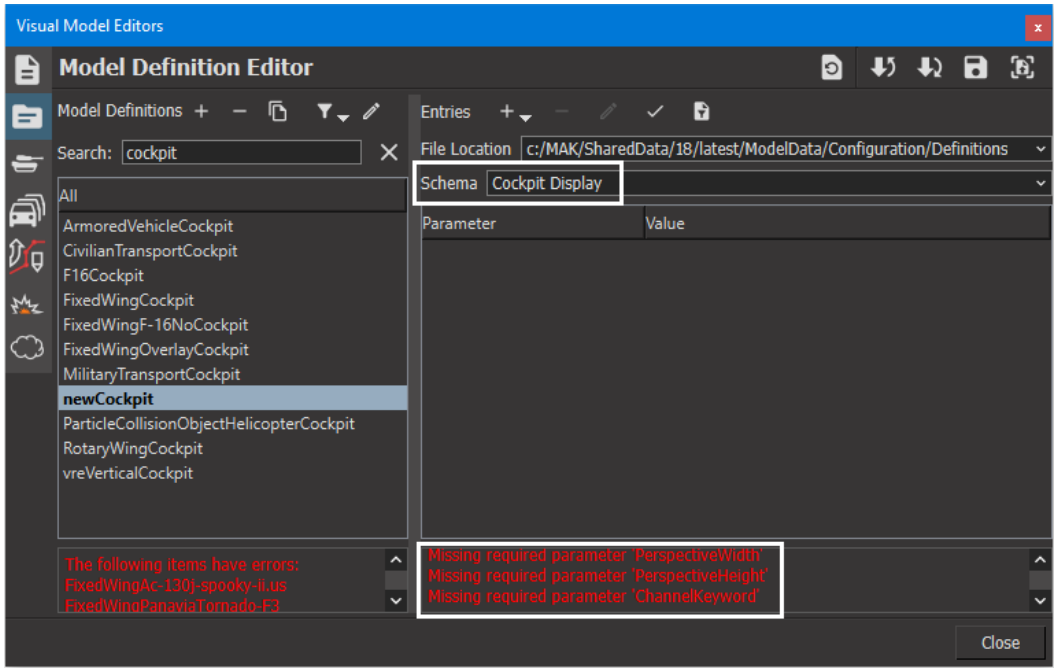
15.2.2 Creating a Model Definition for a Cockpit

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 "12.4. Creating a Model Definition" on page 307. Be sure to select Cockpit Display as the schema. A new model definition is added to the display (Figure 111). The message window lists the parameters that you must add. These are the parameters defined by the schema.

Figure 111. New cockpit model definition



2. Add the required parameters, as described in "12.8.1 Adding Parameters to a Model Definition" on page 309. Table 37 describes the required parameters.

Table 37. 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.

Table 37. Required parameters for cockpit models (continued)

Parameter	Description
ChannelKeyword	A keyword that tells a channel to display any object that has the keyword in its definition. Default: showCockpits. (This keyword means that any channel that has the showCockpits keyword will show cockpits.)
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:
    a. Add an Attribute parameter, as described in "12.8.1 Adding Parameters to a Model Definition" on page 309.
    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. Enter 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 38.
4. Map an object type to your definition, as described in "14.3. Adding an Object Type Mapping" on page 333.

Table 38. Updaters provided

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

Table 38. Updaters provided (continued)

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

15.3. Adding Object Models as Props

Props are terrain objects that you can select independently of the terrain. "9.2. Extracting Props from a Terrain" on page 278 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 "12.1. Creating and Editing Model Definitions" on page 304. 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.)

15.3.1 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.

15.4. Clutter Entities

In VR-Forces, users can create "clutter" in the form of surface watercraft or fixed-wing aircraft using a clutter area. These entities come from prerecorded data to provide realistic but low-impact background entities.

The watercraft and aircraft in the data are mapped to similar entities in VR-Forces. If you have models you would prefer to use, however, you can customize the files that VR-Forces uses to map the data to entity models and use your own model instead.

15.4.1 Configuring Aircraft Clutter Entities

The prerequisite is to have an aircraft entity already defined in the SMS that you intend to use for your VR-Forces scenario. You require the Object Type as it is configured in the Simulation Object Editor (SOE). In this example, we use an ultralight trike.

> The Aircraft Clutter Area in VR-Forces does not include rotary-wing aircraft, gyrocopters, or lighter-than-air aircraft.
>
> The area also defaults to excluding military aircraft, though you can enable them in the *./data/simulationModelSets/SMS-Nameaircraft-clutter-area.sysdef* file, where *SMS-Name* is the name of the included entity-level SMS. For example, if your SMS is an extension of *EntityLevel.sms*, then *SMS-Name* is EntityLevel.

The Mapping File

A *csv* file contains a list of aircraft types in the ADS-B data, mapping them to VR-Forces entities. The first value in each row is a type of aircraft, and the second value is the object type. The other values in each row assist in classifiying the object types.

The following procedure assumes that you will edit it using a spreadsheet program such as Microsoft Excel. You can also edit the values using a text editor.

Using Your Custom Model in a Clutter Aircraft Area

When specifying the entity type, you can use a wildcard (-1) value to include subvariants or multiple countries.

➤ **To add a custom model to the mapping file as a category option:**

1. Open *./appData/settings/vrfSim/ADS-B_Data_Mapping.csv*.
2. For each ICAO aircraft type in column A (AC Type) that you want to represent in an aircraft clutter area using your custom model, enter the object type in column B (SISO Enum) of the same row. This example maps all aircraft of the type TL30 to a specific ultralight trike, `1:2:225:81:1:0:1`.

A	B
AC Type	SIS Enum
TL30	1:2:225:81:1:0:1

3. Save the *ADS-B_Data_Mapping.csv* file.

   If an aircraft clutter area in a VR-Forces scenario includes an aircraft of the appropriate type, it displays in the area using your model.

15.4.2 Configuring Ship Clutter Entities

The prerequisite is to have a ship entity already defined in the SMS that you intend to use for your VR-Forces scenario. You require the Object Type as it is configured in the Simulation Object Editor (SOE). In this example, we use our custom model for a sailing ship.

The Mapping File

A *sysdef* mapping file maps information from the AIS data to models in VR-Forces. In the mapping file, there are two methods for mapping entities. One is to display the closest matching entity for a category of watercraft. The other is to map a specific vessel in the data to a specific model.

VR-Forces does not include models for specific vessels in the AIS data. Instead, it takes the AIS information and creates entities based on the category of ship. It selects a ship in the SMS of the closest type. You can add your own models to be another option in the categories.

The AIS data that VR-Forces uses to generate ship clutter contains the maritime mobile service identity (MMSI) and International Maritime Organization (IMO) identification numbers. You can use these to specify an entity to use for a particular vessel. If that vessel is included in a ship clutter at a particular time of day, VR-Forces uses that entity to represent it in the scenario. MMSI and IMO are expected to be unique, but there is no guarantee that one you specify is included in the AIS data.

> ⓘ In the world, MMSI data is more likely to be transmitted because it is associated with radios, therefore it has a higher precedence in the mapping file than IMO.

Using Your Custom Model in a Clutter Ship Area

As mentioned above, you can either add a model to use generically for watercraft of a certain category, or you can create an entry for a specific vessel. When specifying the entity type, you can use a wildcard (`-1`) value to include subvariants or multiple countries. When creating an entry for the vessel, you can enter either the MMSI or IMO information.

➤ **To add a custom model to the mapping file as a category option:**

1. Open *./data/simulationModelSets/`SMS-Name`/vrfSim/systems/other/ship-clutter-area.sysdef*, where `SMS-Name` is the name of the included entity-level SMS. For example, if your SMS is an extension of *EntityLevel.sms*, `SMS-Name` would be EntityLevel.

2. Locate the appropriate entry in the `vessel-type-code-mapping` block. For our example, we will add our sailing ship's entity type information to the `entry ;; "Engaged in" Sailing` block:

```
(vessel-type-code-mapping ;; priority 4
...
  (entry ;; "Engaged in" Sailing
    (type-code-leading-digits-list )
    (explicit-type-codes-list 36)
    (entity-types
      (entity-type 1 3 -1 61 7 -1 -1) ;; sailboat (one mast)
      (entity-type 1 3 -1 61 6 -1 -1) ;; sailing ship
          (two mast)
    )
    (use-hull-length True)
)
```

3. Save the *ship-clutter-area.sysdef* file.

   If a ship clutter area in a VR-Forces scenario includes a watercraft that matches the category, the model you specified can be used to represent .

➤ **To add a custom model for a specific vessel:**

1. Open *./data/simulationModelSets/SMS-Name/vrfSim/systems/other/ship-clutter-area.sysdef*, where *SMS-Name* is the name of the included entity-level SMS. (See step 1 in the above procedure for details.)
2. You can create an entry for the `mmsi-mapping`, the `imo-mapping`, or both. For example:

```
(mmsi-mapping ;; priority 1 OPTIONAL
  (entry
    (mmsi 368137770) ;; MAYFLOWER II
    (entity-type 1 3 0 61 6 0 1) ;; sailing ship (two mast)
  )
)
(imo-mapping ;; priority 2 OPTIONAL
  (entry
    (imo 8642016) ;; GOLDEN HINDE
    (entity-type 1 3 0 61 6 0 1) ;; sailing ship (two mast)
  )
..)
```

3. Save the *ship-clutter-area.sysdef* file.

   If a ship clutter area in a VR-Forces scenario includes a watercraft that has the MMSI or IMO that you specified, it displays in the area using your model.

Excluding Entities from Ship Clutter

You can exclude specific types of entities or specific watercraft from ship clutter areas. You can specify an entity type, or you can exclude specific MMSI or ISO codes. This enables you to permanently omit entities from your VR-Forces scenarios, rather than deleting them at runtime.

In the *./data/simulationModelSets/SMS-Name/vrfSim/systems/other/ship-clutter-area.sysdef*, there are blocks for excluding entities:

```
(excluded-entity-types )
(excluded-mmsi )
(excluded-imo )
```

You might want to exclude a type of entity if you do not need that type for your scenario. You might also see odd behavior from a particular entity and choose to omit them.

Examples

To exclude entities by type, add the entity object type to the list:

```
(excluded-entity-types
  (entity-type 1 3 0 61 4 0 0) ;; fishing boat
  (entity-type 1 3 0 61 7 0 0) ;; sailboat
)
```

To exclude entities by MMSI, add their IDs to the list:

```
(excluded-mmsi
  (mmsi 209798000)
  (mmsi 366721530)
  (mmsi 367467330)
)
```

15.5. 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 39 describes the parameters specified in *Lights_DefaultValues_3_2.csv*.

Table 39. 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.
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.

Table 39. Light point type settings (continued)

Parameter	Descriptions
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.
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.

Table 39. Light point type settings (continued)

Parameter	Descriptions
Daylight Controlled	Specifies whether 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 the SensorFX section of the VR-Vantage documentation for a description of how to configure light point bulb types in SensorFX.

15.5.1 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.

15.5.2 Configuring Light Point Group Models

You can group light points into light point groups, which allow you to 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.

Entity Light Source and Emissive Texture Support

You can control entity light sources and emissive textures using light groups.

To assign a light group to a light source or emissive geometry within an entity model, you must include the `@dis light_group` comment directly above the relevant light source or emissive geometry nodes. This comment specifies the light group to be used.

> Each light group must have an associated light point to be recognized. If a light group does not have an associated light point, the emissive geometry will be controlled by the default group, behaving as though there is no `@dis light_group` comment in the model.

15.5.3 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.

This 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.

15.6. Aerodrome and Vehicle Lighting

This section covers lighting for airports, runways, and vehicles.

15.6.1 Best Practices for Lightpoints

There are things to keep in mind when configuring lightpoints. For an OpenFlight file, you should configure them using CDB Lightpoint Palette Name. VR-Vantage applications map CDB Lightpoint Palette Names to the Hierarchy value in *./appData/importConfig/Lights_DefaultValues_3_2.csv* (column D in spreadsheet programs). In an FBX file, use a Standard Omni or Spot Light source to add the comment in the Object Properties dialog. The format is @dis_lightpoint followed by the CDB Lightpoint Palette Name, which is also the Hierarchy value in that CSV file.

Examples are:

```
  "@dis lightpoint \Light\Platform\Air\Aircraft_Helos\Navigation\Red_
Light"
  "@dis lightpoint LightPlatformAirAircraft_HelosNavigationGreen_Light"
```

As shown above, you can write it with or without backslashes.

For more information, see "15.5. Configuring Light Points" on page 355.

If you want to control the intensity for a light, not simply configure it for either on or off, you must specify a Light Group Name in a comment.

To control Emissive textures and Light Source intensities, there must be a Lightpoint with the same CDB Lightpoint Palette Name as the emissive texture geometry or dynamic light. For example:

```
"@dis lightpoint LightPlatformSurfaceShip_BoatMilitaryHorizon_
BarGreen_Light"
```

Example of Emissive Texture Geometry:

```
"@dis emissive LightPlatformSurfaceShip_BoatMilitaryHorizon_BarGreen_
Light"
```

Example of a Light Source:

```
"@dis dynamiclight  LightPlatformSurfaceShip_BoatMilitaryHorizon_
BarGreen_Light"
```

Table 53 on page 419 lists light groups for the surface domain.

15.6.2 Airfield Lighting

To control lights in an OpenFlight airport runway model, lightpoints must be organized into discrete light groups. These light groups are identified by a string in the Comment tab of a Lightpoint System node or Group node.

The string must consist of `@dis light-group` followed by the ICAO airport code, the runway identifier, and the light group type: `@dis light-group` *ICAO*:*Runway Identifier*:*Lightgroup*. For example:

```
@dis light-group KBOS:22L:APP
```

Table 40. Airfield light group types

Value	Description
APP	Approach lights
CNTR	Center lights
DTHR	Displaced threshold lights
Edge	Edge lights
END	End-of-runway lights
REIL	Runway end identifier lights
RGL	Runway guard lights
TDZ	Touchdown zone lights
THR	Threshold lights
VGSI	Visual guide slope indicator (includes PAPI and VASI lights)

See also "3.5.7 Adding Custom Airfield Lighting" on page 167.

15.6.3 Vehicle Lighting Switches

Entity appearance switches are configured on a Switch node in an OpenFlight file or a Dummy/Locator node in an FBX file.

There must be an attribute string in the Comment or Object Properties tab specifying what that switch controls. The string always begins with `@dis switch` and then an attribute such as `damage` or `head_lights`. It also must include child group nodes with a comment identifying the resulting state of the switch: `@dis state 0` or `@dis state 1`.

For more information and a complete list of all supported appearances and articulations, see "Chapter 17. MAK Specification Notes" on page 397.

Land Platforms

See Table 50 on page 412 for a list of all supported land domain appearances. The MAK extended attributes related to lighting are at the end of the table.

Example:

```
@dis switch left_indicator_lights_on
  @dis state 0
  @dis state 1
```

Aircraft Platforms

See Table 51 on page 414 for a list of all supported air domain appearances. The MAK extended attribute related to lighting is at the end of the table.

Example:

```
@dis switch hdu_lights_on
  @dis state 0
  @dis state 1
```

Maritime Vehicles

See Table 52 on page 417 for a list of all supported surface domain appearances. The MAK extended attributes related to lighting appearance are at the end of that table. You can also find the MAK extended attribute for the lighting-related articulation at the end of Table 63 on page 429.

Example of enabling or disabling lights:

```
@dis switch running_lights_on
  @dis state 0
  @dis state 1
```

Examples of specifying light groups for lighting intensity:

```
"@dis light_group flight_deck_lights"
"@dis light_group glide_path_indicator_lights"
"@dis light_group horizon_bar_lights"
```

Table 53 on page 419 lists the light groups you can use for surface vessel lighting intensity.

Example of surface vehicle lighting articulation:

```
"@dis articulated_part horizon_bar"
```

Table 63 on page 429 is a complete list of all supported articulations.

15.7. 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`.

15.8. Configuring Wakes in the Visual Model Editor

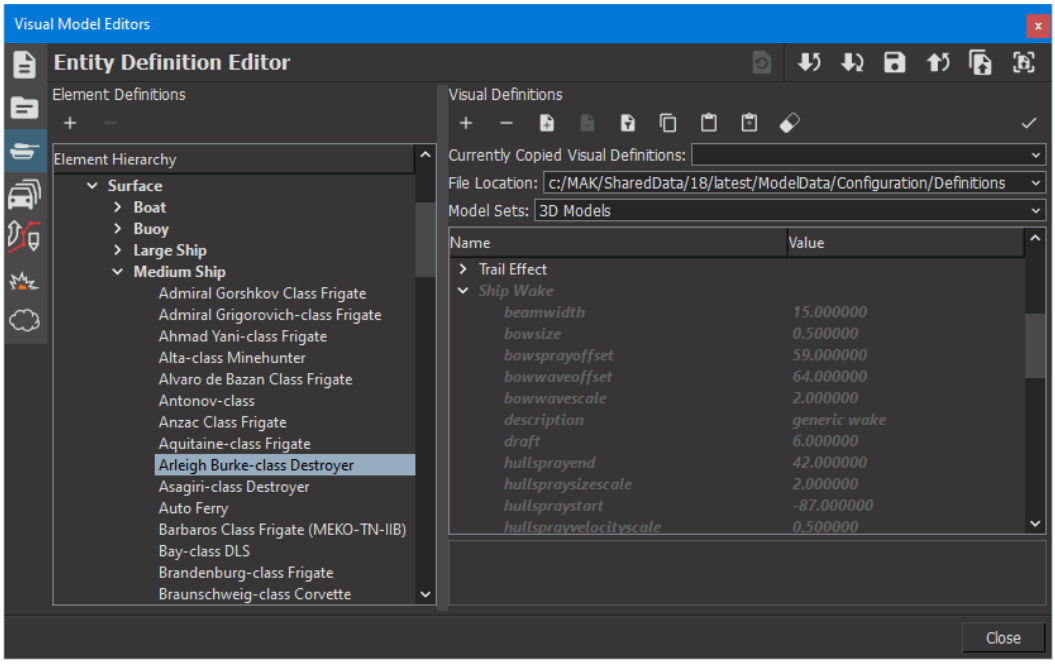
When dynamic ocean is enabled, MAK ONE applications simulate wakes on surface entities. Wakes are configured in the Ship Wake visual definition of an entity's element definition. If you add your own model, you may want to change the default configuration values for its wake to more closely match the 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 User 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 112).

Figure 112. 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 41 describes each parameter.

Table 41. Ship Wake visual definition attributes

Parameter	Description
beamwidth	The width of the ship at its widest point.
bowsize	The size of the bow in world units. This affects the wavelength of the bow wave and the initial spread of spray particles at the bow. Use 0 for a pointy bow.
bowsprayoffset	An offset from the ship position from which to emit spray particles.
bowwaveoffset	An offset from the ship position, along the direction of travel, at which bow waves will originate.
bowwavescale	A scaling factor for the bow wave's amplitude at the bow of the ship.
draft	The depth of the hull underwater.
hullsprayend	The offset from the ship at which hull spray effects end.

Table 41. Ship Wake visual definition attributes (continued)

Parameter	Description
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*.
maxanglebetweenshipandwake	Maximum angle between the ship's heading and the wake's velocity before the wake is not rendered. Unless specified, the value used is 45.
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.

Table 41. Ship Wake visual definition attributes (continued)

Parameter	Description
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.
wakerenderspeedcutoff	Prevents wake visuals from rendering when the entity is moving under the specified speed. Speed is measured in meters per second.

15.8.1 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 20 m 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 User Guide*
- "Configuring Marine Conditions" in the *VR-Forces User 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.

15.8.2 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.

15.9. 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 113) 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 soiltype for this model definition to match with. See "Entity Movement and Soiltype" in the *VR-Forces User Guide* to learn what soiltypes 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 114).

For information on model and element definitions, see "Chapter 12. Model Definitions" on page 303 and "Chapter 13. Element Definitions" on page 313.

Figure 113. Persistent track parameters in Decal Schema

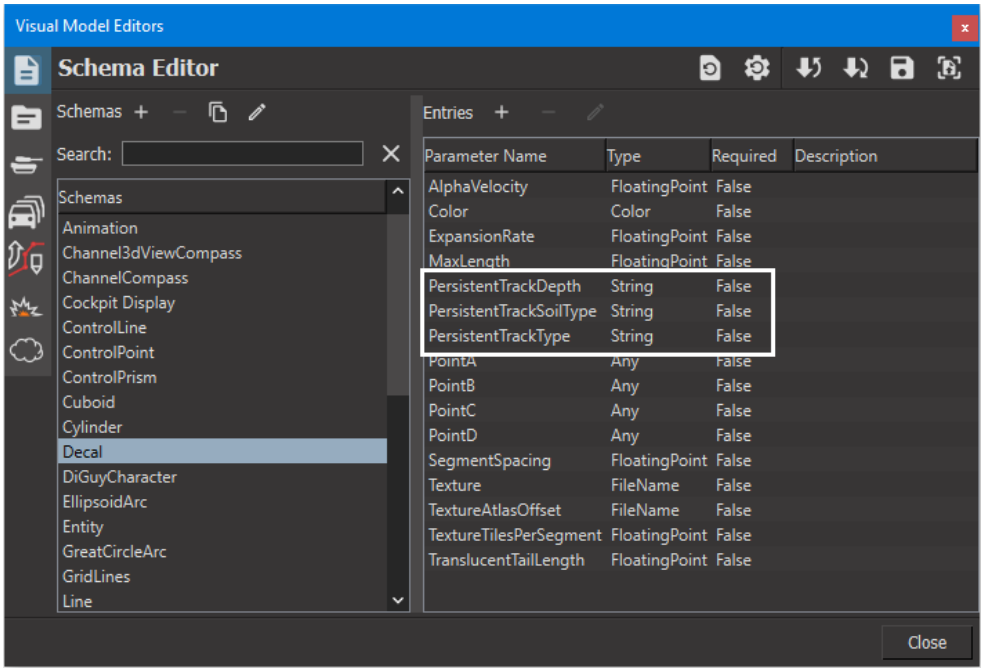
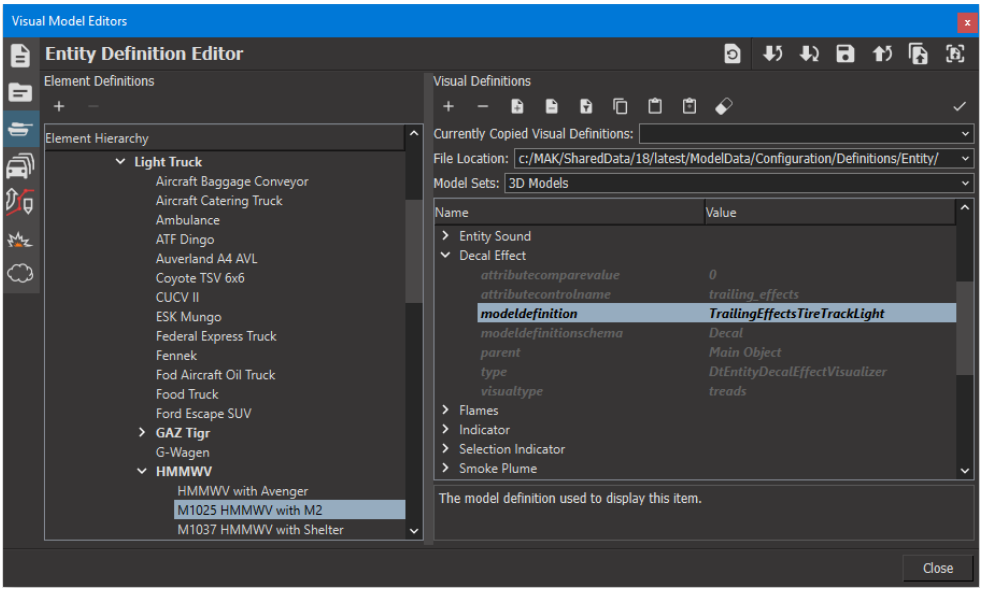


Figure 114. Decal Effect definition in Entity Definition Editor



15.10. Configuring Floating Lifeforms

Some entities have swimming animations. By default, most lifeforms do not float.

➤ **To configure a lifeform to float:**

1. Decide whether you want to export the edited model definition to its own file or to an existing file. (For details about the model definition workflow, see "12.3. Model Definition Files" on page 306.)
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 86 on page 304).
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, see "12.10. Importing Model Definitions" on page 312.)
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, see "12.7. Filtering the List of Model Definitions" on page 309.)
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, see "13.11. Importing Element Definitions" on page 327.)

11. Select Entity Definition Editor page (Figure 91 on page 317).
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**.

15.11. 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 7.6, Transforming Nodes Programmatically, in the *VR-Forces Developers Guide: Vantage*.

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 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 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 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/19/latest/TerrainData/Terrain/Hawaii/Geometry/OpenFlightAlaMonana/AlaMoanaGeocentric.meta*
- *./SharedData/19/latest/ModelData/Structures/Poles/FlagPole/FlagPoleFlag.meta*
- *./SharedData/19/latest/ModelData/Structures/Poles/WindSock/WindSock.meta*
- *./SharedData/19/latest/ModelData/Structures/Towers/WindTurbine3Blades/model.meta*

15.12. 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.

15.12.1 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 workflow, see "12.2. Workflow Issues for Model Definitions" on page 305.)
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 86 on page 304).
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, see "12.10. Importing Model Definitions" on page 312.)
5. Optionally, filter the model definitions in the list. (For details, see "12.7. Filtering the List of Model Definitions" on page 309.)
6. Select the model definition that you want to edit.
7. Click the Add Parameter button ( ). A list of parameters displays.
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, enter a string that should look for in the metafile to identify the texture pattern to use.
12. Export the model definitions.

15.12.2 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 7.6, Transforming Nodes Programmatically, in the *VR-Forces Developers Guide: Vantage*.

A metafile is associated with a terrain or model in one of these 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.

This 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.

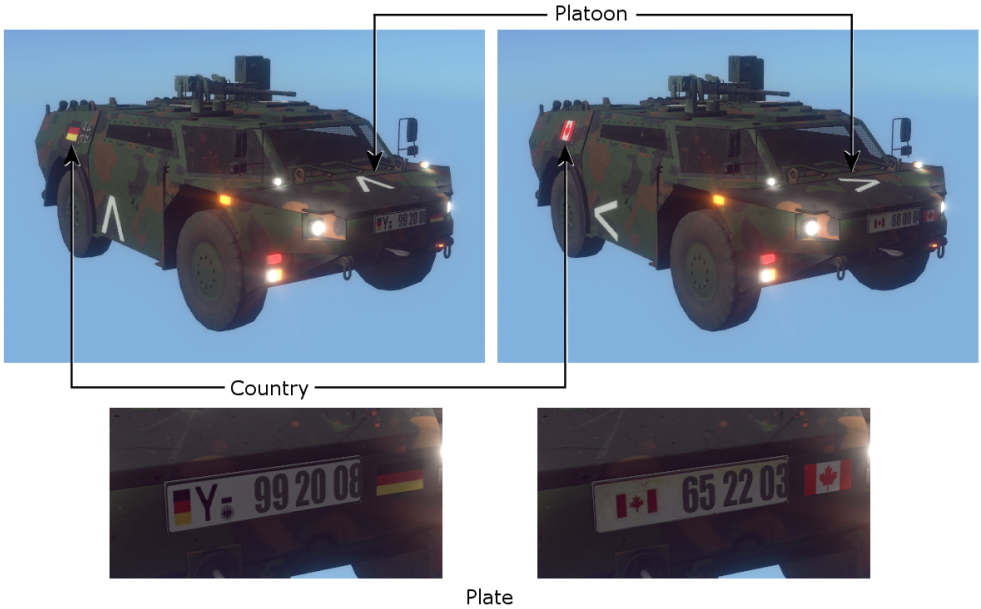
The UV offsets for these models are configured in *Airbus320a.dtxc*. (The file contains explanatory comments.)

```
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
```

15.13. 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 115).

Figure 115. 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, or texture paths for other 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 "15.12.1 Configuring Texture Mapping in the Model Definition" on page 369. 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 User Guide*.)

ⓘ All of the files for configuring markings on the Fennek are in *./SharedData/19/latest/ModelData/Vehicles/Wheeled/FennekArmoredCar*.

15.13.1 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, see "Chapter 17.1.1 The WRM Specification (DIS Notes)" on page 398).

Figure 116 shows the locations of polygons on the Fennek Armored Car for which annotations have been specified.

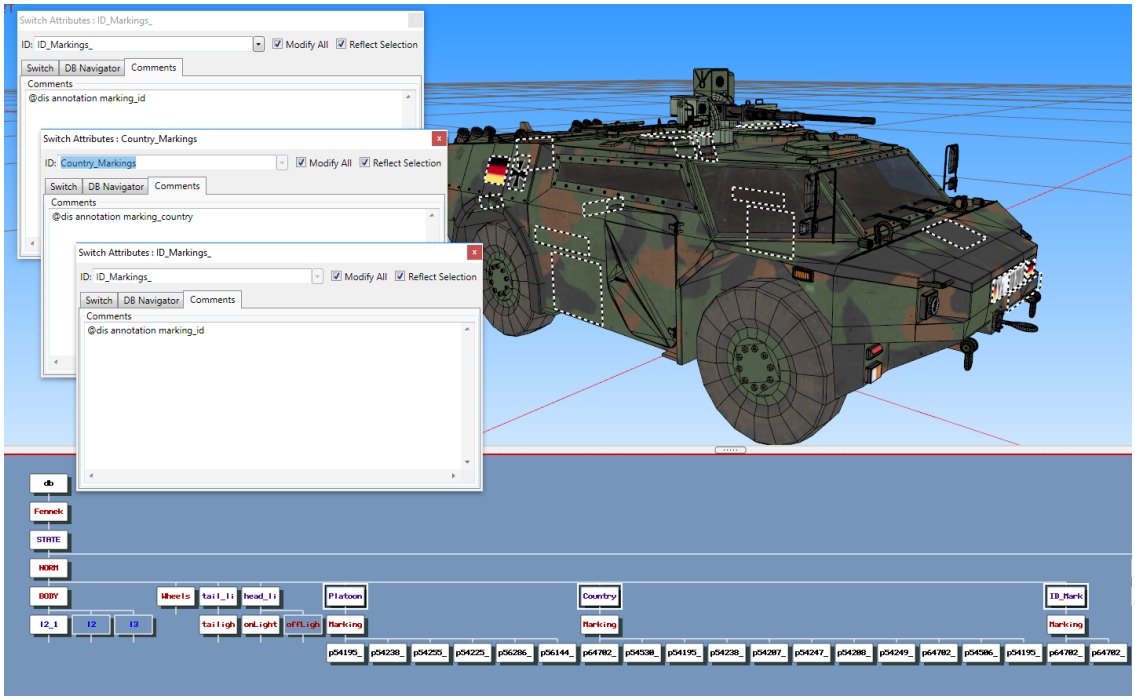
The annotations are in the format:

```
"@dis annotation annotation_name"
```

The specific annotations for the Fennek are:

```
"@dis annotation marking_platoon"
"@dis annotation marking_country"
"@dis annotation marking_id"
```

Figure 116. Specifying @dis annotations in modeling tool



15.13.2 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/19/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"
              }
            },
...
          ]
        }
      ]
    }
  ]
}
```

15.13.3 The Texture Mapping File

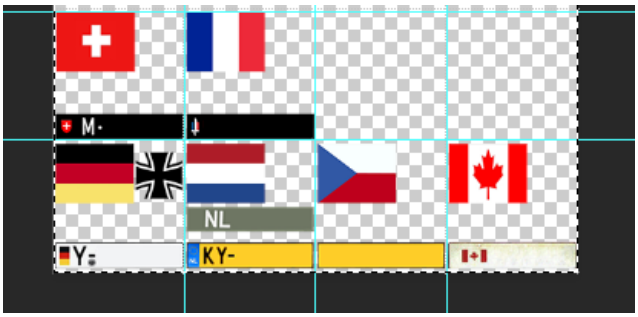
For each type of marking you want to use, you need a *.dtxc* file. It maps the marking annotations to the UV coordinates of the textures in the texture file or the texture path for additional textures.

Markings

The file *CountryMarkings.dtxc* maps the textures for the flags in *CountryMarkings_png.meif* (Figure 117). (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 117. CountryMarkings textures



Camouflage

The file *Camouflage.dtxc* maps the textures for camouflage to the directory where the camouflage files are stored. The paths are relative, allowing you to define folders of the same name with different files for each model.

```
camouflage,COLPAT,Null,""
camouflage,COLPAT,Desert,"/Desert"
camouflage,COLPAT,Forest,"/Forest"
camouflage,COLPAT,Winter,"/Winter"
```

15.14. Adding a Depth Water Mask to a Model

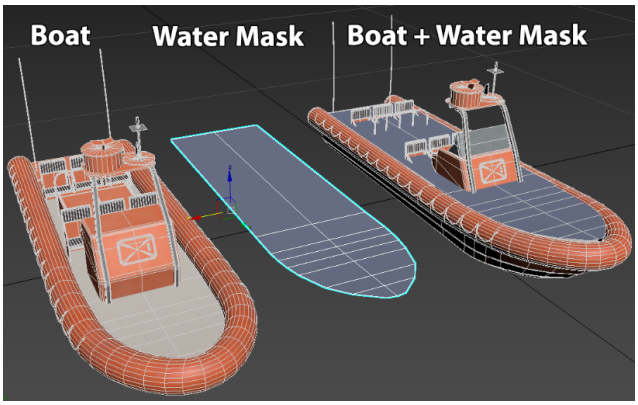
By default, when something is below mean sea level, the dynamic ocean displays. To prevent the dynamic ocean from drawing inside a model, such as watercraft, when the bottom is blow the waterline, you must create a polygonal mask, attribute it with a comment, and mark it as hidden.

Figure 118. Boats with and without depth water mask

0

You do this by adding the comment "`@dis annotation write_depthmask`" to the Comment Field or the Object Properties of the geometry node in the model, and then setting the alpha of polygons or vertices to zero or marking the polygon as hidden.

Figure 119. Example of boat and water mask



See "8.13. Masking the Ocean" on page 257 to learn how to mask water from parts of the terrain.

15.15. Flipping DDS Textures for a Model

If a model displays 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 "8.9. Displaying DDS Textures Correctly" on page 251.)

➤ **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 86 on page 304).
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.

15.16. 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 42 describes the settings you can change. For details about clipping planes, see:

- "Setting the Clipping Planes" in the *VR-Vantage User Guide*
- "Setting the Clipping Planes" in the *VR-Forces User Guide*

Table 42. 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.

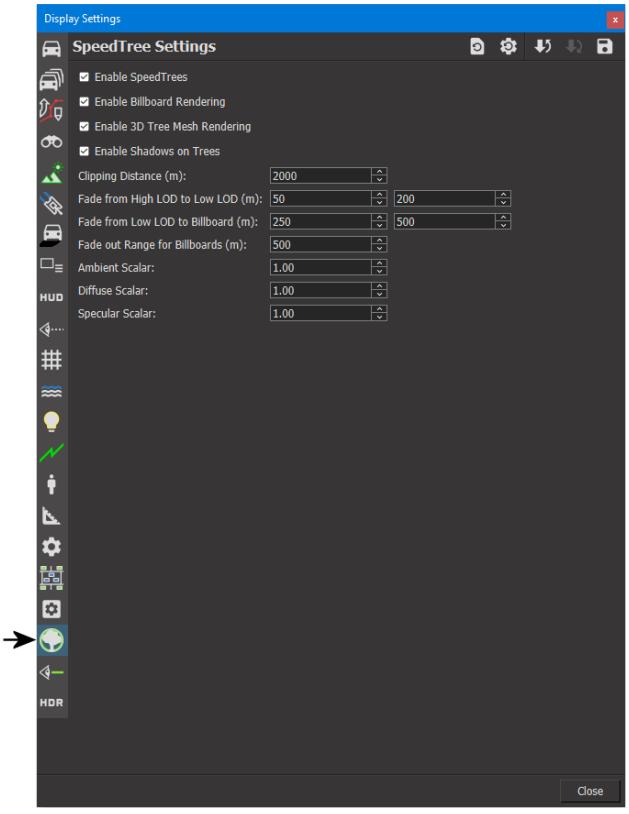
Table 42. SpeedTree settings (continued)

Parameter	Description
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.
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 120).

Figure 120. SpeedTree Settings page



3. Update parameters as desired.
4. Click Close.

15.16.1 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:

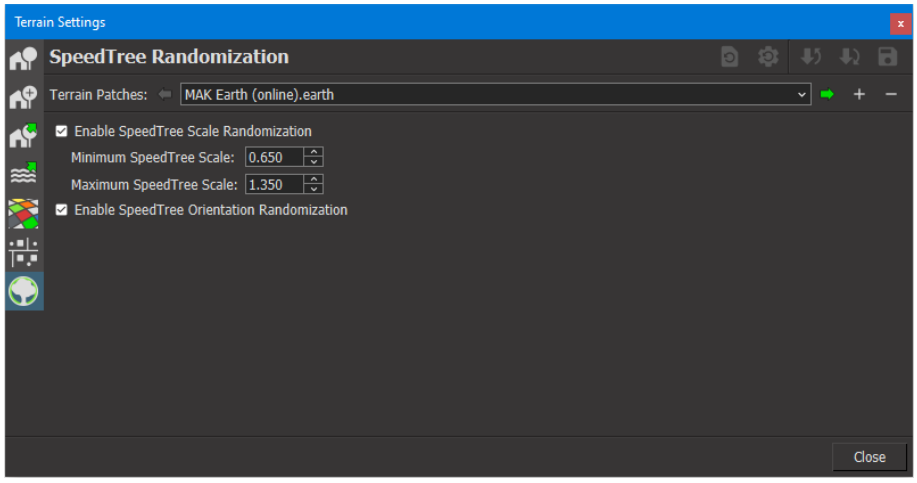
- The **scale** attribute in its model definition. If the scale attribute is not part of the model definition, the value defaults to 1.0.
- The **marker-scale** attribute in the **model** element that configures the trees.
- A random scale factor as specified on the SpeedTree Randomization page on the 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 121).

Figure 121. 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.

15.16.2 Displaying SpeedTrees in Planar Reflections

By default, SpeedTrees are not displayed in planar reflections. This is due to their impact on performance in MAK ONE applications.. You can enable their display. For details, see either:

- "Configuring Ocean Planar Reflections" in the *VR-Vantage User Guide*
- "Configuring Ocean Planar Reflections" in the *VR-Forces User Guide*

15.17. 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. Optionally, you can use the double-sided attribute on polygons by adding the comment "@dis Culling_Off" to the Object Node of the model.
  - 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, allowing 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.

15.18. 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.

15.19. Compressing Model Files (medfTool)

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:

```
medfTool [--ignoreNonBaseTex --dds_flip --directory directory
  --extendLodsEqual distance --extendLodsGreaterOrEqual distance
  --extendLodsTo distance --file filename --filter paths
  --ignore_errors --genDDS -m extensions --norecurse --onlyIfNew
  --threads <integer> --forceOpaque simple_filename --removeNodes
  #comment --removeHighestLod --search directory --scaleLod scalar
  --verbose --extensions extensions --optimize optimization
  --ignore_rest --version --help]
```

Table 43 describes the arguments.

Table 43. medfTool arguments

Argument	Description
(--dds_flip | -d)	Flip DDS images before encrypting them.
(--disableVertexColor | -c)	Ignores the vertex color or alpha in an FBX model file. Most renderers do not support vertex or alpha color, so use this to override it when it is in the model.
--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.
(--extensions | -x) *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.
(--file | -f) *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.

Table 43. medfTool arguments (continued)

Argument	Description
(--forceOpaque | -q) *simple_filename*	Specifies a filename and extension, but no path, to force polygons with these filenames to be opaque. Accepted multiple times.
--genDDS	Generate DDS files.
(--ignore_errors | -i)	Ignore errors in the encryption process and continue without exiting.
(--ignoreNonBaseTex | -b)	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.
(--image_extensions | -m) *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.
(--onlyIfNew | -o)	Only generates an encrypted file if the expected output file does not exist.
(--optimize | -z) *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.
--outputDirectory *filepath*	The directory where files should go.
--removeHighestLod	Removes the highest LOD under a group if it has more than one LOD.
(--removeNodes | -r) *#comment*	Removes 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.
--scaleLod *scalar*	Scales the switch in/out distance of all the children under an osg::LOD node by the specified value.
(--search | -s) *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.

Table 43. medfTool arguments (continued)

Argument	Description
(--threads | -p) *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.
(--verbose | -V)	Verbose output.
--writeNodeInfo *string*	Prints all nodes with comments.
(--ignore_rest | --)	Ignores all command-line options that are listed after this argument.
(--version | -v)	Displays version information.
(--help | -h)	Displays command-line usage information.

This 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
```

This 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
```

16. Material Classification for SensorFX

SensorFX is a plug-in to MAK ONE applications. It provides physics-based sensor views for Long Wave Infrared (LWIR), Midwave Infrared (MWIR), and night vision goggles (NVG). In physics-based sensor views, the rendering of terrain and simulated objects takes into account the material of the object, air and material temperature, ambient light, and the characteristics of the sensor.

This chapter provides a brief example of material classifying a vehicle model.

16.1. Material Classification Process Overview 386
16.1.1 Software Tools 386
16.2. Files for Model Material Classification 387
16.2.1 Material System File 387
16.2.2 Active Region File 389
16.2.3 Intensity Modulation File 391
16.3. Material Classification of Models 391
16.4. Debugging Material Classification 392
16.5. Model Definitions for Materially Classified Models 393
16.5.1 The msFilename Parameter 393
16.5.2 The powerPlantActiveRegions Parameter 393
16.5.3 The movingActiveRegions Parameter 393
16.5.4 The weaponFireActiveRegions Parameter 394
16.6. Configuring Particle Systems 394

16.1. Material Classification Process Overview

The overall process for materially classifying a model is:

1. Determine what 3 or 6 materials you want to use for your classified model and then manually create a material system (*.ms*) tab-delimited text file to associate each material's name with a number (0–5).
2. Bring the original visual texture into Adobe Photoshop (or your preferred modeling tool that supports editing an image's independent channels).
3. Create one or two *.png*, *.rgb*, or *.dds* files with three color channels—red, green, and blue—to store information about the materials. These files are referred to as EMAT textures, which are materially classified versions of the visual RGB textures. They should use the standard file naming convention, for example, *MyModelName.emat2.png* and *MyModelName.emat3.png*.

Table 44. Texture file mapping

Trace all areas that use material #	Such that they have a value of 255 in the channel	Of texture
0	Red	1 (emat2)
1	Green	1 (emat2)
2	Blue	1 (emat2)
3	Red	2 (emat3)
4	Green	2 (emat3)
5	Blue	2 (emat3)

16.1.1 Software Tools

Models require configuration to display correctly in SensorFX. To materially classify models, you need:

- A standard modeling tool such as Adobe Photoshop, GIMP, GenesisMC, or your preferred software product.
- A spreadsheet program (such as Microsoft Excel) or a text editor. (If you use GenesisMC, this is optional.)

Many people, including MAK developers, prefer to use Photoshop and Excel. If you do not have access to or expertise in these tools, you can use GenesisMC. GenesisMC generates the *.ms* file and allows you to edit the *.emat* files.

If you want to look at the files used in the examples, you can download them:

- Photoshop users: https://vtmakcepstgagt01.blob.core.windows.net/productinstallers/GenesisMC/VWBus-Photshop.zip

- GenesisMC users: https://vtmakcepstgagt01.blob.core.windows.net/productinstallers/GenesisMC/VWGunBus-Genesis.zip

16.2. Files for Model Material Classification

The model used as the example in this chapter is the Volkswagen Type 2 Bus.

The files for the model and the texture are:

- Model: *VWVan.medf* (converted from *VWVan.fbx*)
- Associated texture: *VWVan.png*

Figure 122. Model files for Volkswagen Type 2 Bus

The files required to configure material classification on the Volkswagen Type 2 Bus:

- Material System file (.ms): *VWVanEmat.ms*
- Active Region file (.ar): *VWVanEmat.ar*
- First emat: *VWVan.emat2.png*
- Second emat: *VWVan.emat3.png*

- SensorFX uses the names of the textures and models to look for the materially classified configuration files.
- The convention for the texture files is to include "`emat2`" and "`emat3`" in the file name. While GenesisMC can create an emat1 texture file, SensorFX does not use it. Instead, to save texture memory, SensorFX uses an optional *.imod* (intensity modulation) file to fulfill the same purpose.
- To look at the files used for this example, see the links under "16.1.1 Software Tools" on the previous page.

16.2.1 Material System File

SensorFX uses the material system file (*.ms*) to map a channel of an emat texture to a particular material. It specifies up to six materials, with three color channels in each of up to two *.emat* files that are used on the model, and also their attributes. The **.emat.ms* file is a tab-delimited text file that you can open with a spreadsheet program, such as Excel, or a text editor. (The former is typically easier to work with than the latter.)

Each material is listed in a separate row, and the values in the first column are arbitrary names. The best practice is to enter meaningful names. The values in the other columns include information used by MAK ONE applications to achieve the desired materially classified effects. The order of the rows is significant and is used to match texture layers in *.emat* files. Each row can be generated by GenesisMC, or you can cut and past from a database of material configurations found in this file: https://vtmakcepstgagt01.blob.core.windows.net/productinstallers/GenesisMC/ExampleAssets/MaterialLines.csv. (The latter method is preferred by MAK.)

Figure 123 is the material system file for the Volkswagen Type 2 Bus, *VWVanEmat.ms*. Note that the order is important; refer to this example and Table 44 on page 386.

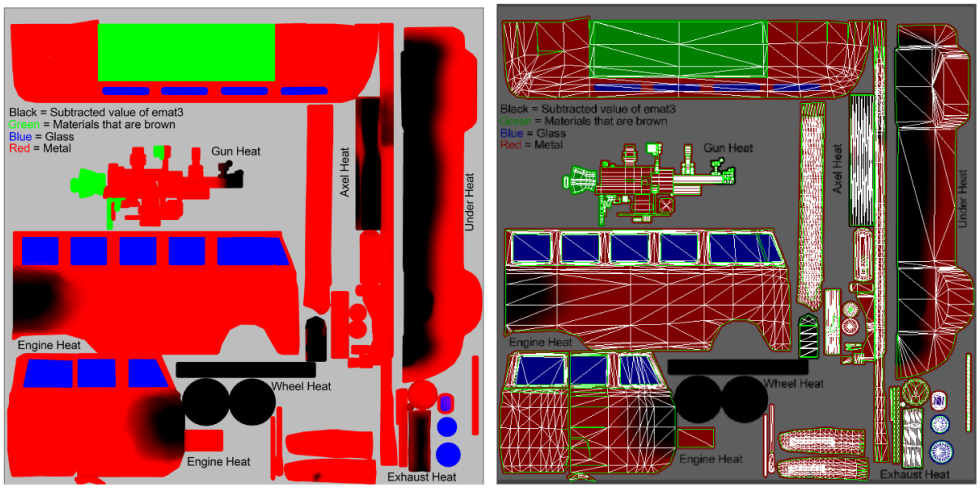
Figure 123. Material system file example

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Volkswagen Type 2 Bus																					
2	Material	Top_Surface_Boundary	Bottom_Surface_Boundary	Surface_Normal	Layer_1																	
3	Name	Alt	BC1	T1	V1	Q1	BCn	Tn	Vn	Qn	modelT	Shadowflag	i	j	k	Num_Layers						
4	GalvanizedSteelPaintedGreen	0	1	0	0	0	2	15	0	0	2	0	0	0	1	2	PaintGreenP005	0.02	MetalSteelGalvanized	0.005		
5	Cotton	0	0	25	0	0	2	32	0	0	2	28	0	0	1	2	FabricCottonGray	0.1	FabricCottonOlive	0.001		
6	ConstructionPlateGlass	0	3	0	10	0	2	25	20	0	2	0	0	0	1	1	ConstructionGlassPlateWindow	0.005				
7	EngineThermal	0	1	0	0	0	2	150	100	0	2	0	0	0	1	3	PaintGreenP005	0.005	MetalSteelGalvanized	0.01	ConstructionFoamBlack	0.007
8	WeaponThermal	0	1	0	0	0	2	200	20	0	2	0	0	0	1	2	PaintGreenP005	0.005	MetalSteelGalvanized	0.005		
9	BlackWeatheredRubber	0	1	0	0	0	2	100	75	0	2	0	0	0	1	1	RoofingRubberBlackWeathered	0.005				

In Figure 123, the first three materials map to the RGB channels of emat2 (Figure 124).

- Red Channel: GalvanizedSteelPaintedGreen (body of the vehicle)
- Green Channel: Cotton (mat on the vehicle's roof)
- Blue Channel: ConstructionPlateGlass (the vehicle's windows)

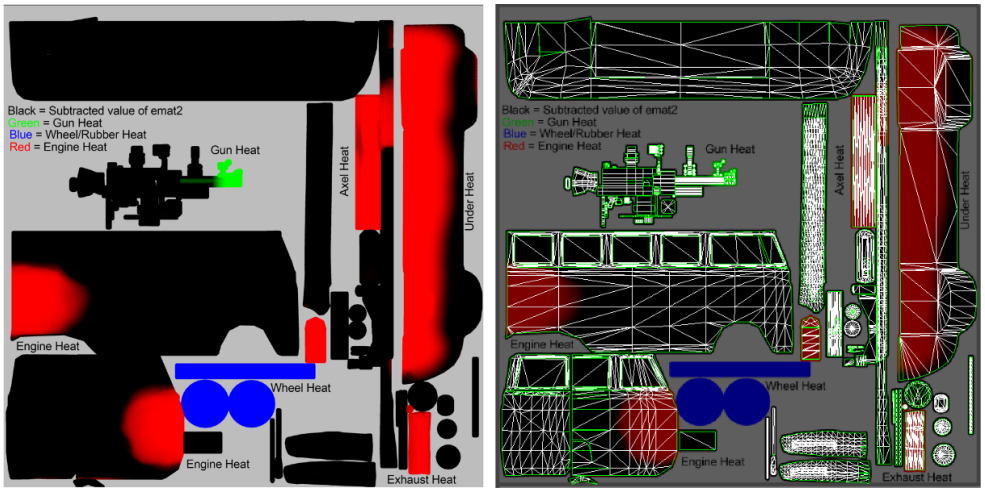
Figure 124. Example of the *VWVan.emat2.png* file



In Figure 123, the second three materials map to the RGB channels of emat3 (Figure 125):

- Red Channel: EngineThermal (vehicle's engine)
- Green Channel: WeaponThermal (weapon on the roof)
- Blue Channel: RubberRubberBlackWeathered (tires)

Figure 125. Example of the *VWVan.emat3.png* file



Gradients show where the falloff of thermal temperatures. This allows you to realistically show where the heat source affects nearby parts of the vehicle or weapon but they are not as hot.

Figure 126. Medium-range infrared (MWIR) rendering in VR-Vantage with SensorFX, no thermals

16.2.2 Active Region File

The active region file (*.ar*) is also a tab-delimited text file that you can open with a spreadsheet program or text editor. You use it to define which materials are dynamic. (Active regions are also called hotspots.) If the model does not have any active regions then it does not need an active region file.

The *.ar* file defines a material for the off state and another material for the on state. It maps which material is the active region material for Powerplant, Moving, and WeaponFire thermal. It also specifies the ambient material when it is not active. There is an initial state and time to heat up attribute.

The *.ar* file must have the same name as the *.ms* file it corresponds to, except that the extension is changed to *.ar*.

- 0: GalvanizedSteelPaintedGreen
- 1: Cotton
- 2: ConstructionPlateGlass
- 3: EngineThermal
- 4: WeaponThermal
- 5: BlackWeatheredRubber

In Figure 127, the active region file specifies which materials are the active: Engine, Artillery, or Friction Thermal.

Figure 127. Example of the *VWVanEmat.ar* file

	A	B	C	D	E
1	Active_Region_Name	Active_Material	Ambient_Material	Initial_State	Impulse_Delta_Time(s)
2	powerPlant	3	0	0	
3	moving	5	0	0	
4	weaponFire	4	0	0	10

Figure 128. Engine thermal (left) and engine and wheel thermal (right) MWIR rendering in VR-Vantage with SensorFX

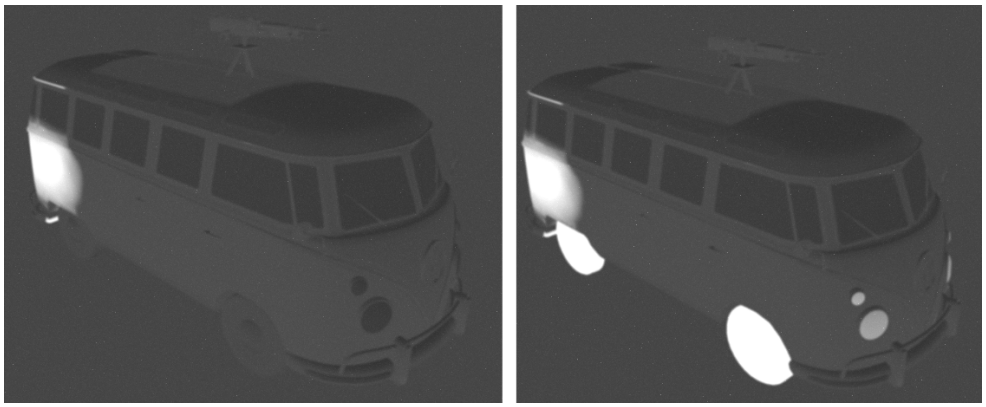
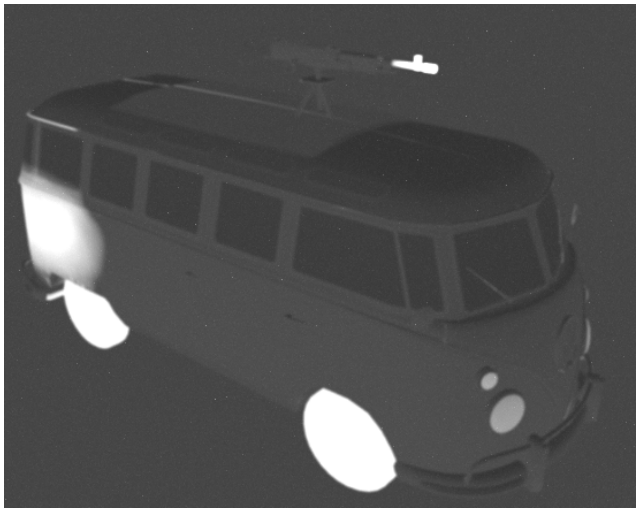


Figure 129. Engine, wheel, and weapon thermal MWIR rendering in VR-Vantage with SensorFX



- PowerPlant active region is turned on and off based on the entity's power plant status appearance bit.

  For information on appearance bits, including the power plant, see Table 50 on page 412, Table 51 on page 414, Table 52 on page 417, or Table 54 on page 419.

- Moving active region is turned on and off based on whether the entity has velocity or not.
- WeaponFire active region is turned on/off based on the fire DIS PDU.

16.2.3 Intensity Modulation File

An intensity modulation file or *.imod* file is an optional file that defines how much of the RGB texture to mix with the sensor response in the final result. It allows the intensity modulation to be defined on a per-material and per-sensor band level. The file must have the same name as the material system file, except that the extension must be *.imod* instead of *.ms*. This file must be in the same directory as the material system file.

16.3. Material Classification of Models

You can create the *.ms* and *.emat* files in the GenesisMC application. You can also create the *.emat* texture files in any image editor that supports editing independent channels of an image, such as Photoshop. They are 24-bit raster files and can be saved as *.png*, *.rgb*, or *.dds*.

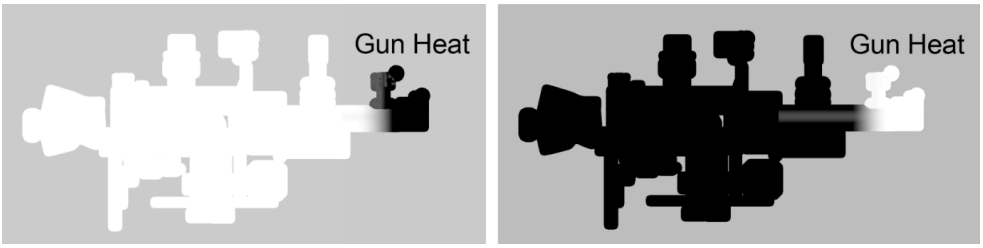
The *.ms* files are text files and you can created them in any text editor by combining material entries from any of the other VR-Vantage vehicle *.ms* files located in the subdirectories of *./SharedData/19/latest/ModelData/Vehicles*.

The *.ar* file is also a text file and configured to map the according active region channels.

When painting each material, it should be in a separate channel with a value of 255. The specific channel used can be arbitrary per material, but the RGB channels must map to the order in the *.ms* file.

The active region materials can have a gradient (0-255) representing the falloff of heat. The value represented in active region material should be subtracted from the ambient material.

Figure 130. GalvanizedSteelPaintedGreen (Ambient Material Channel) and WeaponThermal (Channel)



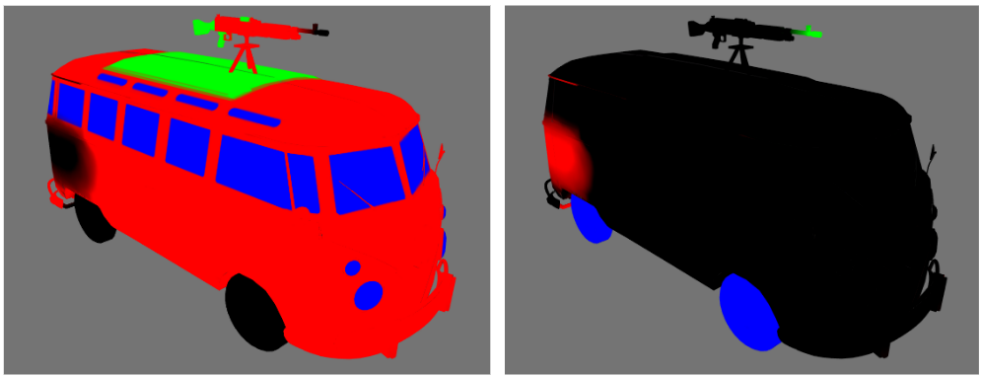
16.4. Debugging Material Classification

In VR-Vantage with SensorFX, you can visualize the EMAT configuration on the model. If the EMATs render properly, then your model is configured correctly. If a file has been misplaced or is not configured correctly, the EMAT textures will not render.

➤ **To display EMAT debug visuals:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Sensor Settings page.
3. Clear the Enable Selected Sensor Effects check box. This disables the specific sensor mode, particularly gain, level, blur, and noise.
4. Select the Render Settings page.
5. From the Debug Visuals list, select either (SensorFX) EMAT2 or (SensorFX) EMAT3.

Figure 131. EMAT2 and EMAT3



16.5. Model Definitions for Materially Classified Models

If you have generated material classification files for a model that is already configured in VR-Vantage, all you have to do is put the required files in the model directory as described in previous sections. You do not have to edit the model definition.

If you have created a new model, you must create a model definition, as described in "Chapter 12. Model Definitions" on page 303.

16.5.1 The msFilename Parameter

The **msFilename** parameter lets you point to a material system file that does not use the naming convention described in "16.2. Files for Model Material Classification" on page 387. Possible reasons for this are if there are two models that need to use the same material system file or there are multiple variations of the same model that need different material systems.

> This is an optional approach to pointing to a material system file. The default and simplest way is to give the file the same name as the model file name minus the extension and add *Emat.ms*. This file must be in the same directory as the model file.

The **msFilename** parameter is not available by default. You must add it to the model's schema before you can use it. See "11.6. Adding a Parameter to a Schema" on page 300.

The parameter must be called **msFilename** and have its type set to string.

To determine which schema to edit, look at the schema listed for the model definition for the model you are editing. The most likely schemas are Entity and Watercraft.

16.5.2 The powerPlantActiveRegions Parameter

The power plant active regions are active regions that should be turned on when the model's power plant is turned on and turned off when the model's power plant is turned off.

In the active region file, the first column of each line specifies the name of an active region. The SH-60 has two active regions, EngineGlass and EnginePaint. In order to map them to the power plant active state in VR-Vantage, the model definition needs a parameter added called **powerPlantActiveRegions**. The value of that parameter is a comma-separated list of active regions that should map to the power plant state.

The default value for the **powerPlantActiveRegions** parameter is **powerPlant** if the parameter is not specified for a given model definition. If the model definition specifies a **powerPlantActiveRegions** parameter, then it overrides the default and **powerPlant** is no longer a valid active region unless it is also specified in the parameter value.

16.5.3 The movingActiveRegions Parameter

The moving active regions are active regions that should be turned on when the model is moving and turned off when the model is stationary. *VWVanEmat.ar* is an example of an active regions file that has a moving active region. The Track active region corresponds to the material on the tracks of the tank. Therefore it must be mapped to the moving active

region in the model definition. The **movingActiveRegions** parameter in the model definition for the T-72 has a value of Track. It maps the Track active region to the moving state of the tank.

The default value for the **movingActiveRegions** parameter is **moving** if the parameter is not specified for a given model definition. If the model definition specifies a **movingActiveRegions** parameter, then it overrides the default and **moving** is no longer a valid active region unless it is also specified in the parameter value.

16.5.4 The weaponFireActiveRegions Parameter

The weapon fire active regions are active regions that should be triggered when the model fires its weapon. *VWVanEmat.ar* is an example of an active regions file that has a weapon fire active region. The Artillery active region corresponds to the material on the weapon of the tank. SensorFX supports only one weapon system per entity for Active Regions, typically the main gun.

The default value for the **weaponFireActiveRegions** parameter is **weaponFire** if the parameter is not specified for a given model definition. If the model definition specifies a **weaponFireActiveRegions** parameter then it overrides the default and **weaponFire** is no longer a valid active region unless it is also specified in the parameter value.

16.6. Configuring Particle Systems

Particle systems are configured slightly differently than other models in SensorFX. Instead of using emat textures to define the materials that are used, they use PIP files to define how the particle system should be handled in sensor modes. In order for SensorFX to know that the model is a particle system instead of a regular model, the MS file must end in *.specialeffect.ms* instead of *.ms*. These MS files are searched for by taking the model name, removing the extension and replacing it with *.specialeffect.ms* (for example, *smoke.osg* would look for *smoke.specialeffect.ms*).

Once SensorFX determines that there is a *.specialeffect.ms* file, it looks for the corresponding *.pip* file. First it looks for the same filename as the MS file except without the *.ms* extension (for example, for *smoke.specialeffect.ms* it would look for *smoke.specialeffect.pip* in the same directory as the MS file). If it does not find that PIP file, it looks for a keyword that it can match to a default PIP file. The default PIP files are in *./data/SensorFX/particleEffects*.

Continuing our example, assume the *smoke.osg* model has a *smoke.specialeffect.ms* file. SensorFX looks for *smoke.specialeffect.pip*. If it does not find it, it uses *smoke.pip*, because it matches the smoke keyword in *smoke.specialeffect.ms*.

Table 45 lists the default PIP files.

Table 45. Default PIP files

Keyword	PIP file
dust	*dust.pip*
smoke	*smoke.pip*
flame	*flame.pip*

Table 45. Default PIP files (continued)

Keyword	PIP file
plume	*plume.pip*
flare	*flare.pip*

17. MAK Specification Notes

This chapter contains information that can help you build entity models that comply with the WRM or FBX specifications, or add that compliance to existing models.

17.1. Overview of MAK Specification Notes 398
17.1.1 The WRM Specification (DIS Notes) 398
17.1.2 The MAK FBX Specification 399
17.1.3 Modeling Best Practices for Efficient Mapping 401
17.2. Model Coordinate Systems 402
17.2.1 Location of the Origin for Models 403
17.3. The Simulation Attribute Lexicon (SAL) 404
17.3.1 SAL Syntax 405
17.3.2 SAL Keywords 405
17.4. Articulated Parts 406
17.4.1 Embedding Simulation Articulation Data in the Model 406
17.5. Entity Appearances 408
17.5.1 Node Hierarchy Example 409
17.6. Animation Sequences 410
17.7. DIS Keyword Values 411
17.7.1 DI-Guy Articulated Part IDs 435

17.1. Overview of MAK Specification Notes

17.1.1 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 correctly and to switch among different entity appearances.

This section 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.

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, ensuring 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. 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, your models should conform to as much of the specification as is appropriate for the model. This ensures the model will be compatible with future versions of MAK ONE applications and also with visualization applications from other vendors.

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.

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.

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.

17.1.2 The MAK FBX Specification

This section describes the FBX specification developed and used by MAK to use FBX models in distributed simulations.

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 how the specification developed by MAK:

- Provides support for FBX as a lower-cost, industry standard.
- Ensures compatibility with existing software and content.
- Is consistent with the WRM specification, allowing it to be made publicly available as a companion to the original WRM specification.

The MAK FBX 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.

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 Freedom (DOF), Level of Detail (LOD), or External Reference nodes.

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.

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 to 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 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.

When you create FBX models for use in MAK ONE applications, following the MAK convention allows 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 must map the nodes in your FBX files to DIS notes by hand as described in this chapter.

FBX nodes are mapped in the file *./appData/importConfig/FBX_NodeNameMap.csv*.

You can edit this file in a text editor or spreadsheet application. There are four columns:

- 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.

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*.

17.1.3 Modeling Best Practices for Efficient Mapping

This section describes how to map models efficiently.

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.

Specific Articulation Keyword Values used by MAK

The bottom of Table 63 includes the supported values for the `@dis articulated_part` keyword (page 435). String values are part of the WRM specification, but are not currently supported by MAK.

Shader-based Effects Maps in FBX

The MAK FBX Specification supports use of shader-based effects maps on models the same way that MAK ONE applications support them on OpenFlight models. For details, see "8.4. Using Shader-based Effect Maps" on page 243 and "8.4.1 Composite Effects Maps" on page 245.

Using Texture Atlases, Metafiles, and Markings

The MAK FBX Specification supports use of texture atlases, metafiles, and dynamic markings the same way that MAK ONE applications support them on OpenFlight models. For details, see "15.12. Using Texture Atlases to Skin Models" on page 368 and "15.13. Adding Dynamically Changeable Markings to Models" on page 371.

Using Material-Classified (Sensor) Textures

SensorFX is a plug-in to VR-Vantage and VR-Engage that lets them 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 of 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.

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 46. 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.

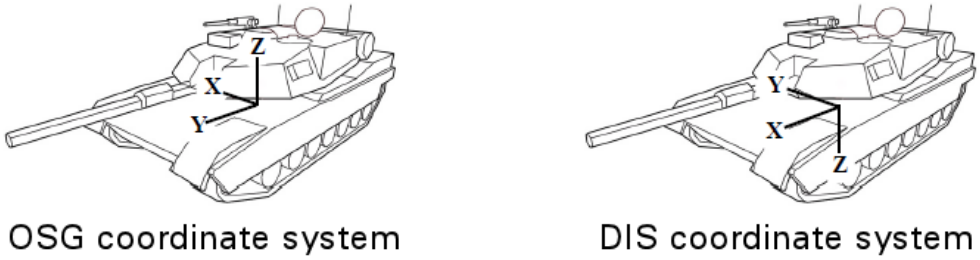
17.2. Model Coordinate Systems

OpenSceneGraph (OSG) and the DIS Standard use different right-hand coordinate systems. OSG uses East, North, Up (ENU). DIS uses North, East, Down (NED). Table 47 describes the two systems. Figure 132 illustrates them.

Table 47. Model coordinate systems

Coordinate	OpenSceneGraph	DIS
X	Right	Forward
Y	Forward	Right
Z	Up	Down

Figure 132. Model coordinate systems



The WRM standard made a modeler's use of OpenFlight and FBX intuitive with respect to a model's coordinate system. When you create models, use the OSG coordinate system. The MAK ONE application converts the coordinates:

- xmodel = yDIS.
- ymodel = xDIS.
- zmodel = -zDIS.

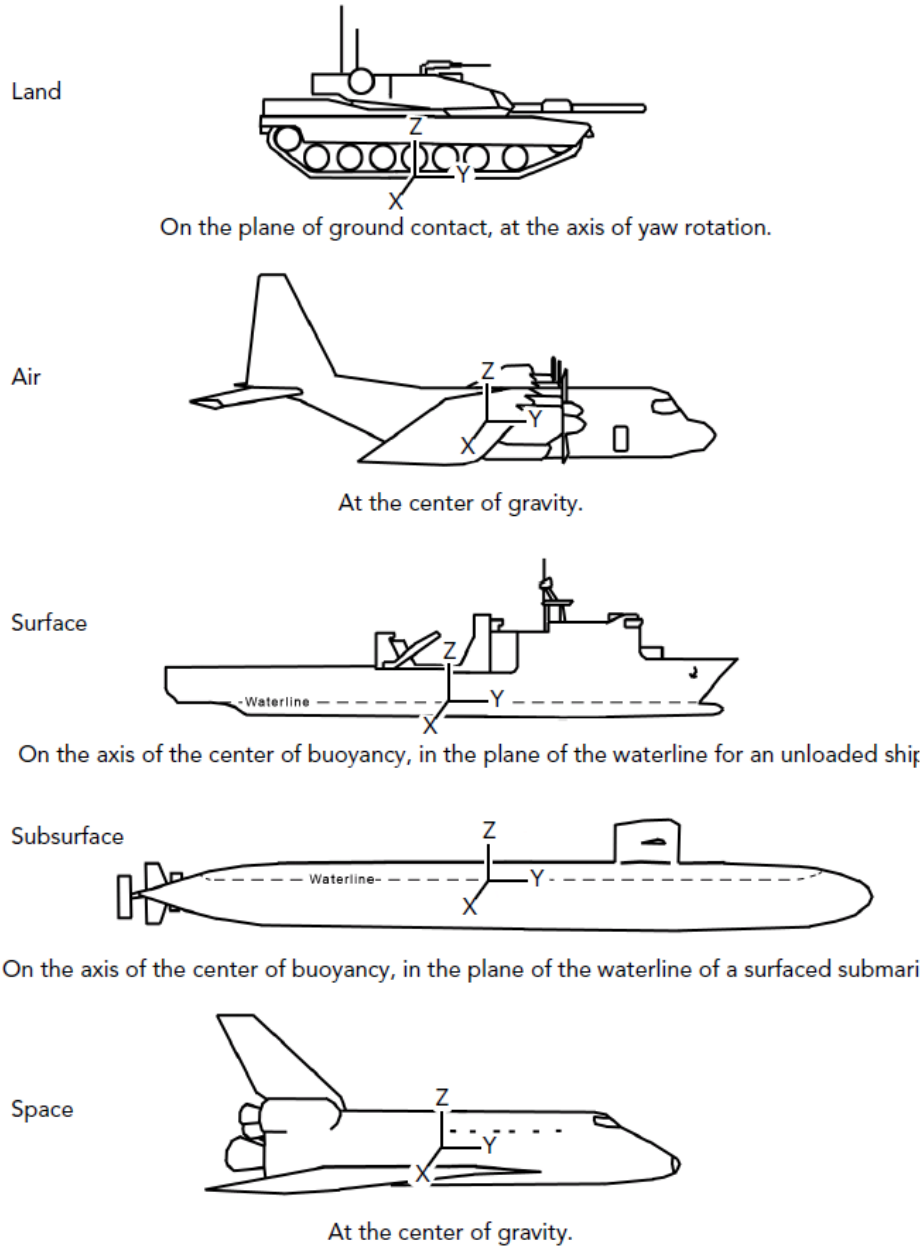
17.2.1 Location of the Origin for Models

Table 48 describes the location of the model origin by domain. Figure 133 illustrates the location of a model's origin for each platform domain.

Table 48. 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 133. Location of origin by domain



17.3. The Simulation Attribute Lexicon (SAL)

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 require customized software to interpret the multiple potential model formats that might otherwise exist.

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 are implemented within the existing file structure, primarily using the node comment fields.

17.3.1 SAL Syntax

The SAL syntax is simple and straightforward:

```
@dis keyword value # comment
```

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.)
- # indicates the start of a comment. Comments are case sensitive. A comment continues until the next `@dis` entry.

17.3.2 SAL Keywords

The WRM Spec supports these 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 Reference for Enumerations for Simulation Interoperability, Articulated Parts. Parts supports by MAK ONE applications are listed in Table 63 on page 429.

  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 Reference for Enumerations for Simulation Interoperability, Entity Appearance, 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: gm_ for guided munitions and lf_ for lifeforms. For a list of supported values, see "17.7. DIS Keyword Values" on page 411.

- `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, see "17.7. DIS Keyword Values" on page 411.

17.4. Articulated Parts

For MAK ONE applications to properly depict the state of articulated parts, models must conform to these specifications:

- There must be one DOF (degree of freedom)/dummy/locator 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/dummy/locator node must have all of the geometry and other nodes associated with the part underneath it. This 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 turret's node must be commented with:

  ```
  @dis articulated_part 4096
  ```

  The gun's 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. 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.

17.4.1 Embedding Simulation Articulation Data in the Model

The SISO Enumeration specific model features are identified by comments in the Null node.

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 Table 63 on page 429 for type values.

For example, comment the DOF/dummy/locator 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 49); *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 these examples:

- Example 1: Turret

  ```
  @dis articulated_part 4096 rotate_min 0,0,-180
      rotate_current 0,0,0 rotate_max 0,0,180
      rotate_step 0,0,45
  ```

- Example 2: Gun

  ```
  @dis articulated_part 4128 rotate_min 0,0,-180
      rotate_current 0,0,0 rotate_max 0,0,180
      rotate_step 0,0,45
  ```

Table 49 lists the kind values for articulated parts.

Table 49. 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.

17.5. 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.

Define a switch node by adding the `@dis` attribute on the switch/dummy/locator node, for example:

```
@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.

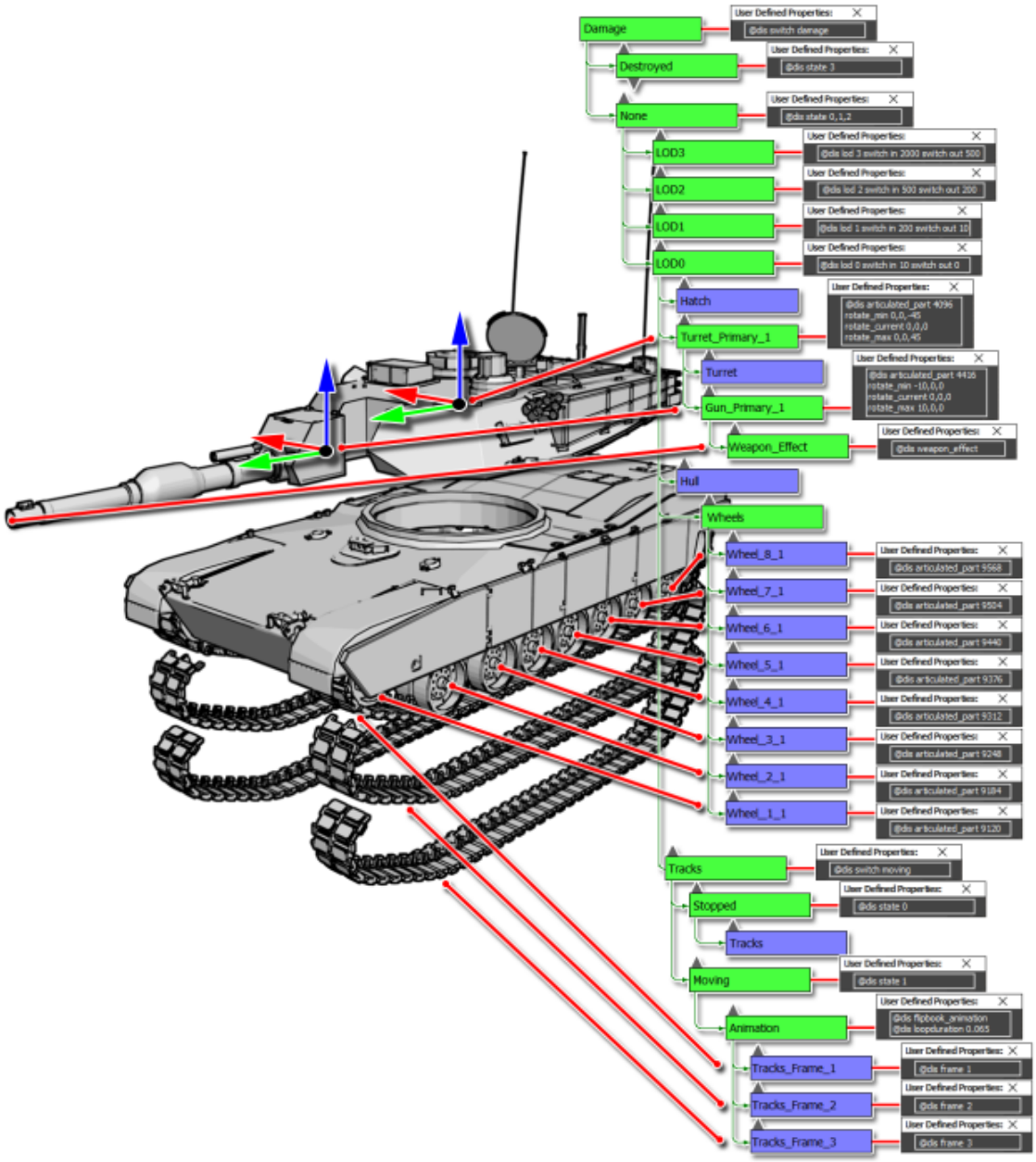
17.5.1 Node Hierarchy Example

In Figure 134, the turret DOF/dummy/locator 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/dummy/locator node that is flagged with animation. The comment on this node is `@dis switch moving`. This switches between two 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 node with the animation flag set to on. This creates a flipbook animation of all nodes below this node. There are three meshes in different states of moving: Frame 1, Frame 2 and Frame 3. At runtime, MAK ONE applications 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 134. 3D model structure



17.6. Animation Sequences

MAK extended the original WRM specification to support 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 a WRM animation comment in the animation node's comment field.

The comment syntax is:

```
@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 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
```

17.7. DIS Keyword Values

Table 50 through Table 62 list the values that you can use for the `@dis switch` keyword by kind and domain. The Bit columns 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 columns, parentheses enclose older values that still work but have been replaced by the value above them. For any switch keyword, the supported state values (`@dis state [value 1],[value 1]...`) are listed in the State columns.

DIS appearance and states:

- Table 50 on the next page – platform kind, land domain
- Table 51 on page 414 – platform kind, air domain
- Table 52 on page 417 – platform kind, surface domain
- Table 54 on page 419 – platform kind, subsurface domain
- Table 55 on page 420– platform kind, space domain
- Table 56 on page 421 – lifeform kind
- Table 57 on page 423 – munition kind
- Table 58 on page 424 – environmental kind
- Table 59 on page 425 – cultural feature kind
- Table 60 on page 425 – sensor kind
- Table 61 on page 427 – supply kind
- Table 62 on page 428– expendable kind

Table 63 on page 429 DIS Articulations lists the supported values for the `@dis articulated_part` keyword. String values are part of the WRM specification, but are not currently supported.

Table 65 on page 436 DIS Attachments lists the values for the `@dis attachment_station` keyword. None of the values are currently supported.

Table 66 on page 437 DIS Animations lists the values for the `@dis animation` keyword. String values are not currently supported.

Table 50. 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
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 50. 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_on) (spot_lights)	Describes whether spotlights are on or off.	0 = Off 1 = On

Table 50. DIS appearance and states — platform kind, land domain (continued)

Bit	Values	Description	State/ Description
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
The following are MAK extended attributes:			
	left_indicator_lights_on	State of left indicator.	0 = Off 1 = On
	right_indicator_lights_ on	State of right indicator.	0 = Off 1 = On
	emergency/warning_ indicator_lights_on	State of emergency/warning lights.	0 = Off 1 = On

Table 51. 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 51. 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 51. 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 (formation_lights_on)	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 51. DIS appearance and states — platform kind, air domain (continued)

Bit	Values	Description	State/ Description
The following are MAK extended attributes:			
	hdu_lights_on	State of HDU lights	0 = Off 1 = On
	anti-collision_light_ day/night	State of anti-collision light day/night	0 = Off 1 = On
	taxi_lights	State of taxi lights	0 = Off 1 = On

Table 52. 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

Table 52. DIS appearance and states — platform kind, surface domain (continued)

Bit	Values	Description	State/ Description
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
The following are MAK extended attributes:			
	warning/safety_lights_on	State of warning/safety lights	0 = Off 1 = On
	distress_beacons/strobes_on	State of distress beacons/strobes	0 = Off 1 = On
	upper_deck_flood_light_on	State of upper deck flood light	0 = Off 1 = On
	flight_deck_lights_on	State of flight deck lights	0 = Off 1 = On
	anchor_lights_on	State of anchor lights	0 = Off 1 = On
	wave_off_light_on	State of wave off light	0 = Off 1 = On
	not_under_command_lights_on	State of not under command lights	0 = Off 1 = On
	switch horizon_bar_light_on	State of horizon bar light	0 = Off 1 = On
	glide_path_indicator_lights_on	State of glide path indicator lights	0 = Off 1 = On
	ram_lights_on	State of restricted ability to maneuver (RAM) lights	0 = Off 1 = On
	fishing_with_nets_lights_on	State of fishing with nets lights	0 = Off 1 = On
	fishing_with_lines_lights_on	State of fishing with lines lights	0 = Off 1 = On
	mine_hunting_lights_on	State of mine hunting lights	0 = Off 1 = On

Table 53 lists the values that you can use for the `@dis light_group` keyword for platform kind, surface domain.

Table 53. Surface vessel lighting intensity - platform kind, surface domain

Value	Description
flight_deck_lights	Light Group with Flight Deck Lights
glide_path_indicator_lights	Light Group with Glide Path Indicator Lights
horizon_bar_lights	Light Group with Horizon bar Lights

Table 54. 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 54. 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 55. 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 56. 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 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

Table 56. 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)	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 lifeform 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 56. 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 lifeform is in a concealed position.	0 = Not Concealed 1 = Prepared Concealed Position
31	concealed_movement	Describes whether the lifeform uses concealment during movement.	0 = Open Movement 1 = Rushes Between Covered Positions

Table 57. 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 57. 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 58. 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 59. 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 60. 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
3-4	damage	Damaged appearance of an entity.	0 = None 1 = Slight 2 = Moderate 3 = Destroyed

Table 60. DIS appearance and states — sensor kind (continued)

Bit	Values	Description	State/ Description
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.	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 60. 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 61. 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 62. 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 63. 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
3520	launcher_arm_primary	Primary launcher arm

Table 63. DIS Articulations (continued)

Value	String	Description
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
4768	primary_launcher_2	Primary launcher 2
4800	primary_launcher_3	Primary launcher 3

Table 63. DIS Articulations (continued)

Value	String	Description
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

Table 63. DIS Articulations (continued)

Value	String	Description
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
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

Table 63. DIS Articulations (continued)

Value	String	Description
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
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

Table 63. DIS Articulations (continued)

Value	String	Description
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	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
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

Table 63. DIS Articulations (continued)

Value	String	Description
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
The following are MAK extended attributes:		
2144	horizon_bar	Horizon bar
16064	towed_array_decoy	Towed array decoy

17.7.1 DI-Guy Articulated Part IDs

The DIS and HLA connections can send DIS articulated part numbers that DI-Guy can use to control individual DI-Guy links. Table 64 lists the articulated part numbers.

Table 64. DI-Guy articulated part IDs

Value	String	Notes
10240	DIGuyDisFocusTarget	
10272	DiGuyDisArtPartBase	
10304	DiGuyDisArtPartBack	
10336	DiGuyDisArtPartCervical	
10368	DiGuyDisArtPartShoulderLeft	
10400	DiGuyDisArtPartShoulderRight	
10432	DiGuyDisArtPartElbowLeft	
10464	DiGuyDisArtPartElbowRight	
10496	DiGuyDisArtPartWristLeft	
10528	DiGuyDisArtPartWristRight	
10560	DiGuyDisArtPartHipLeft	
10592	DiGuyDisArtPartHipRight	
10624	DiGuyDisArtPartKneeLeft	
10656	DiGuyDisArtPartKneeRight	
10688	DiGuyDisArtPartAnkleLeft	
10720	DiGuyDisArtPartAnkleRight	
10752	DiGuyDisArtPartBackLow	

Table 64. DI-Guy articulated part IDs (continued)

Value	String	Notes
10784	DiGuyDisArtPartBackHigh	
10816	DiGuyDisArtPartHead	
10848	DiGuyDisArtPartClavicleLeft	
10880	DiGuyDisArtPartClavicleRight	
10912	DiGuyDisArtPartPalmLeft	
10944	DiGuyDisArtPartPalmRight	
10976	DiGuyDisArtPartToeLeft	
11008	DiGuyDisArtPartToeRight	

If you are an API developer, you can find these enumerations in *./include/vrvDiGuy/DtDiGuyArtPartLinkMapper.h*.

Table 65. 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
	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

Table 65. DIS Attachments (continued)

Value	Value/String	Description
	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 66. 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	

18. Special Effects (Particle Systems)

This chapter describes how to maintain special effects using the Particle System Editor.

18.1. Particle Systems 440
18.2. Adding a New Particle System 440
18.3. Editing Particle Systems 442
18.3.1 Editing the Properties of Keyframes 442
18.4. Creating a Model Definition for a Particle System 443
18.5. Saving and Reverting Particle Systems 443
18.6. Previewing a Particle System 445
18.6.1 Viewing a Particle System With or Without Textures 447
18.6.2 Viewing a Particle System's Wireframes 448
18.6.3 Viewing Debug Visuals 449
18.7. Particle System Properties 451
18.7.1 Preview Parameters for Particle Systems 456
18.8. Particle System Collision Objects 457

18.1. 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 "18.7. Particle System Properties" on page 451.

To learn about materially classifying particle systems, see "16.6. Configuring Particle Systems" on page 394.

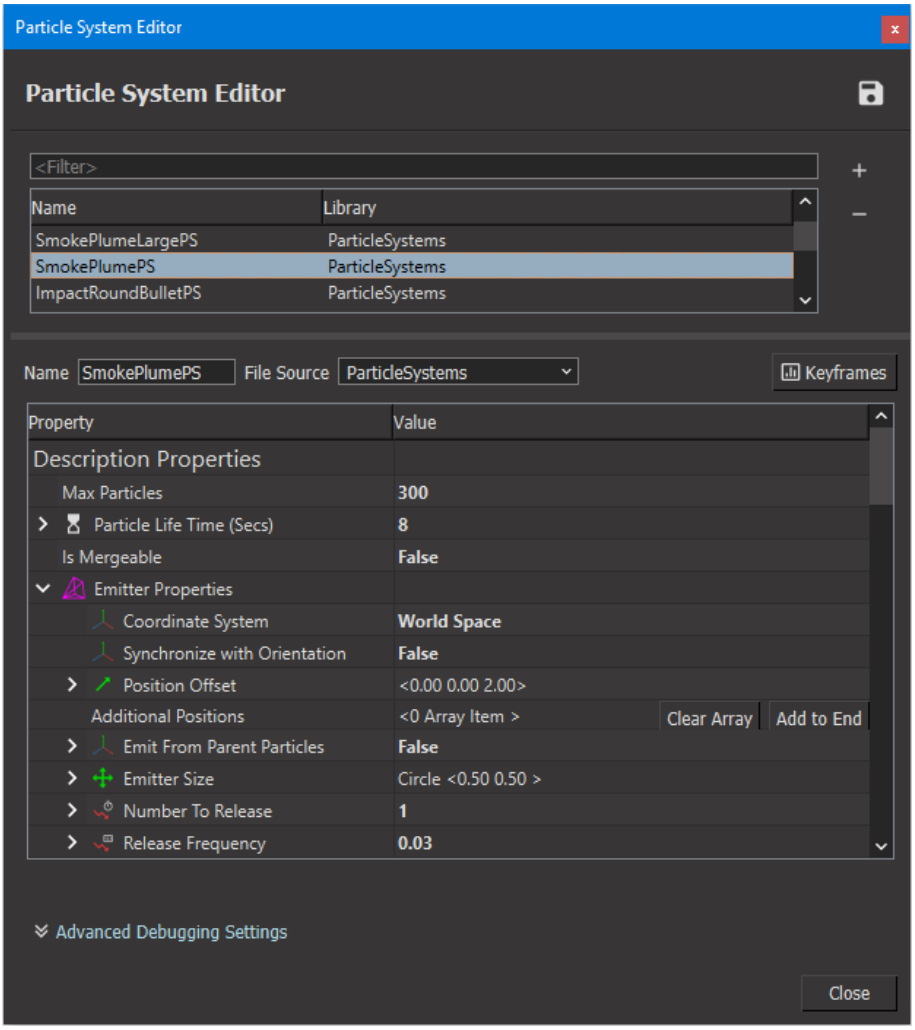
18.2. Adding a New Particle System

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 135).

Figure 135. 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 "18.7. Particle System Properties" on page 451.

18.3. Editing Particle Systems

You can edit the properties of a particle system. For details on the properties, see "18.7. Particle System Properties" on page 451.

➤ **To edit the properties of a particle system:**

1. Choose **Settings > Particle System Editor**. The Particle System Editor opens (Figure 135 on the previous page).
2. Select the particle system for which you want to edit the properties.
3. Change values in the properties table as desired.
4. To save the changes, click Save All ( ).

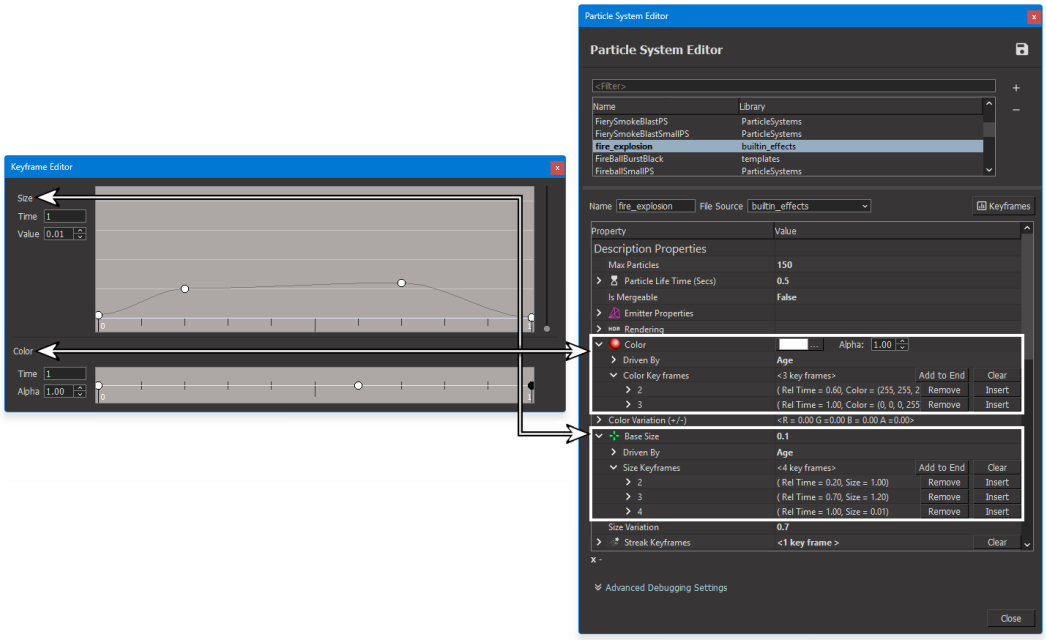
18.3.1 Editing the Properties of Keyframes

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, see Table 67 on page 451.) 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 136).

Figure 136. Keyframe Editor and keyframe properties



3. Change the values as desired.
4. To save the changes, click Save ( ).

18.4. 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 12. Model Definitions" on page 303. 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:**

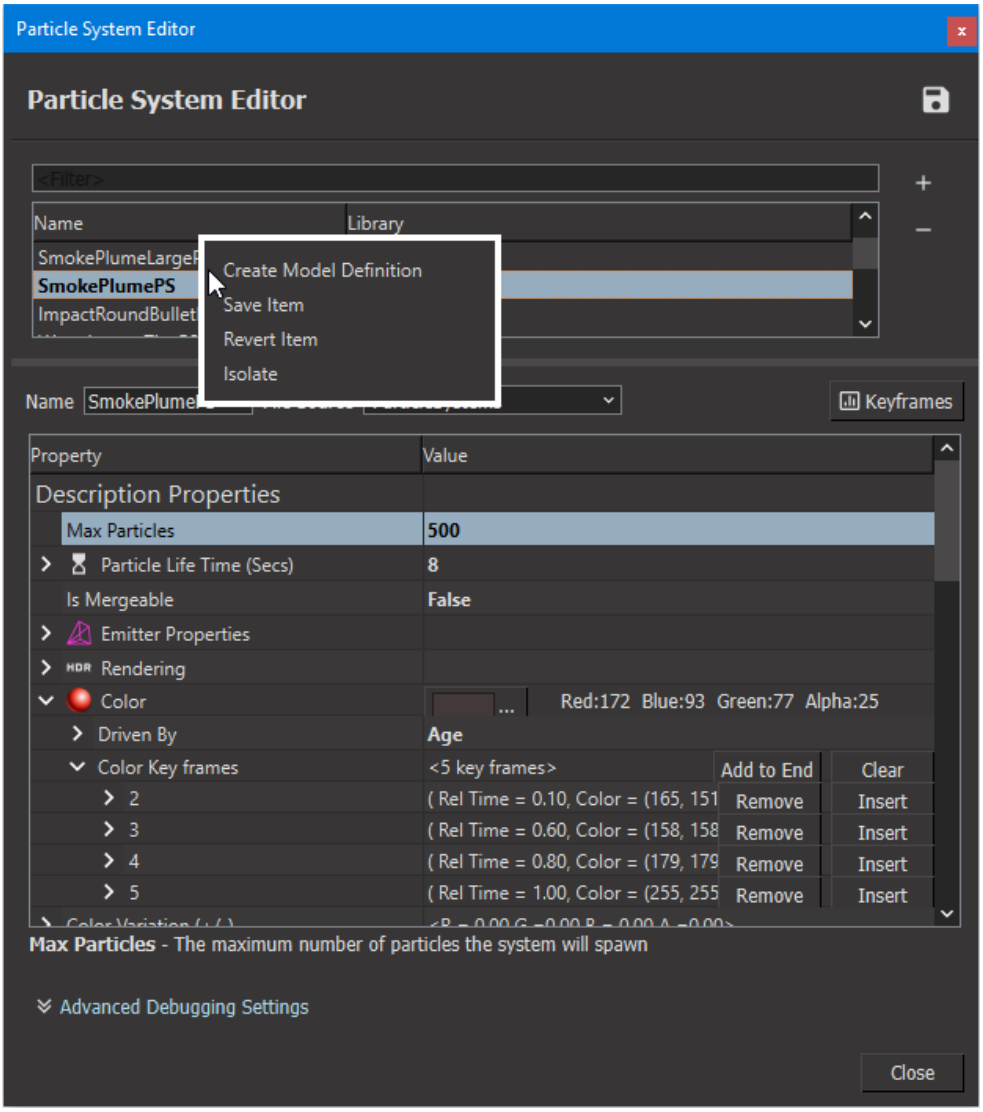
1. In the Particle System Editor, select the particle system for which you want to create a model definition.
2. Right-click and then choose Create Model Definition. The New Name dialog box opens.
3. Enter a name for the model definition (or accept the default name).
4. Click OK. The new model definition 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.

18.5. 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 137).

Figure 137. 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**.

18.6. Previewing a Particle System

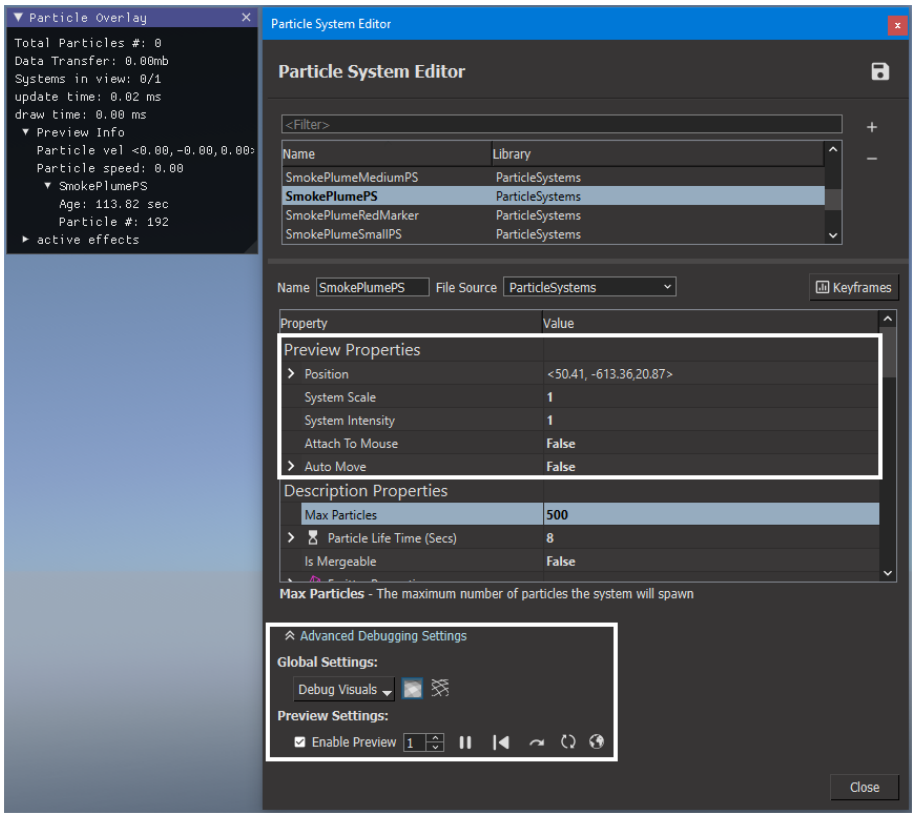
You can preview a particle system in the terrain (Figure 138). You can only test a particle system that has been saved.

Figure 138. 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 to be 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 139).

Figure 139. 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.

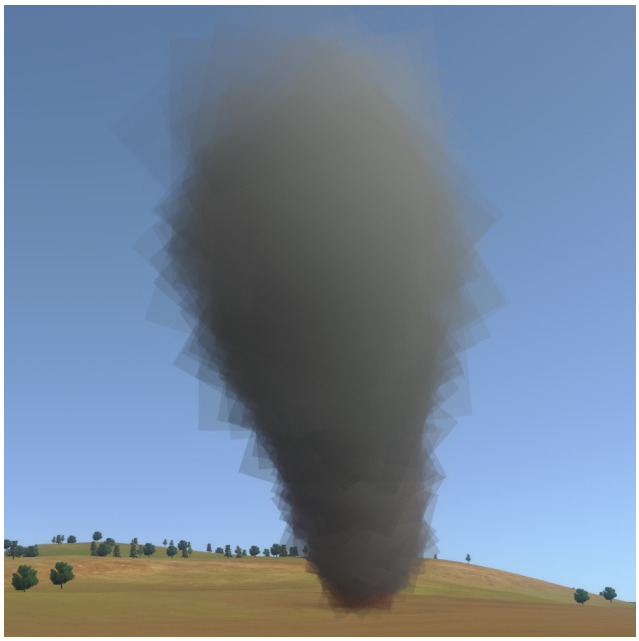
    (Automatically Loop). Loop the animation.

    (Global). The Play/Pause buttons affect any particle effects in the scene, not only any particle systems you are previewing.

18.6.1 Viewing a Particle System With or Without Textures

While previewing a particle system, you can turn the textures on or off (Figure 140).

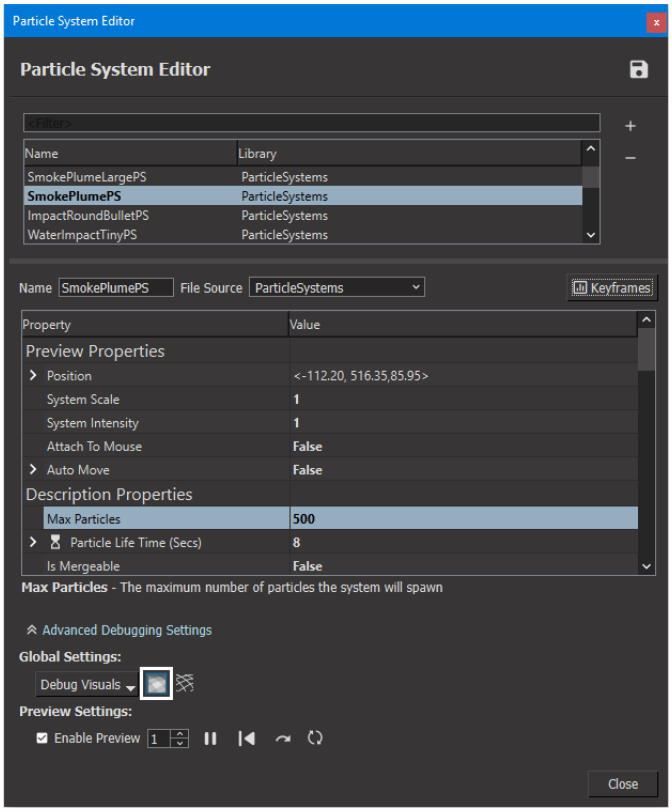
Figure 140. Preview of a smoke plume without textures



➤ **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 141) to toggle the textures on or off.

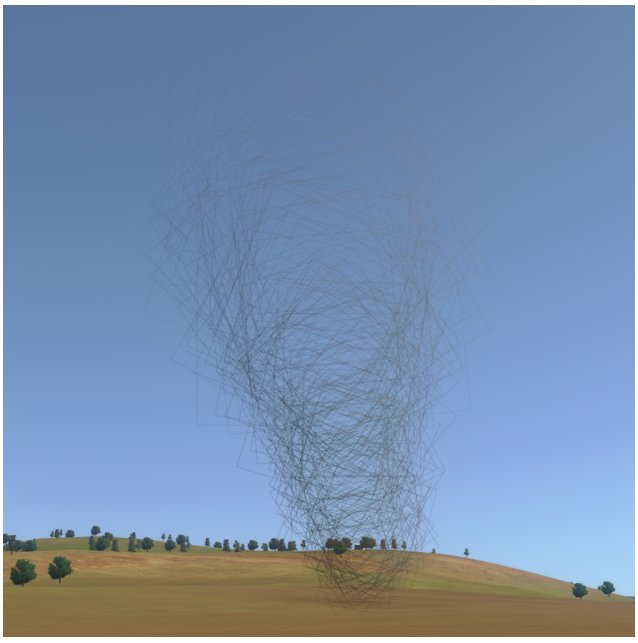
Figure 141. Particle System Editor - Textures



18.6.2 Viewing a Particle System's Wireframes

While viewing a particle system, you can choose to look at the wireframes for the particles (Figure 142).

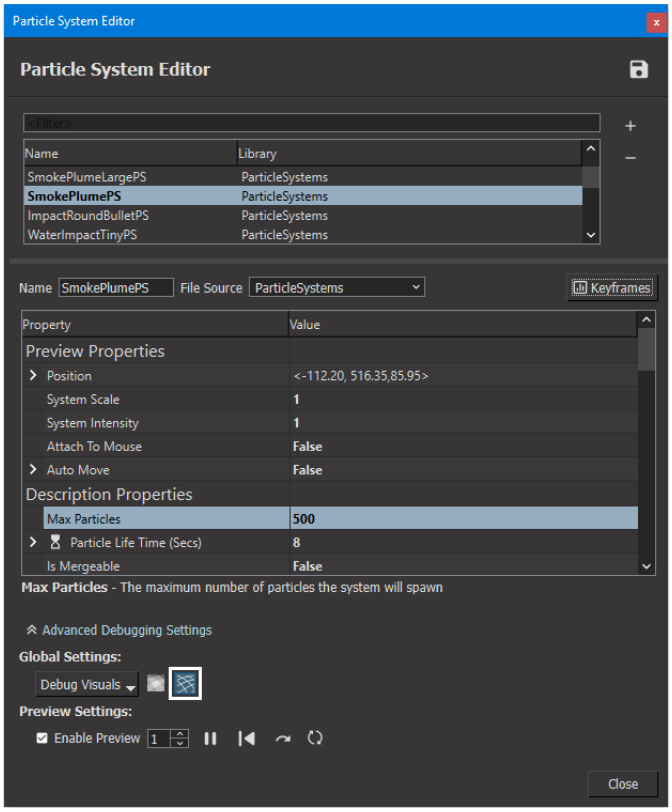
Figure 142. Preview of a smoke plume as wireframes



➤ **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 143) to toggle wireframes on or off.

Figure 143. Particle System Editor - Wireframes



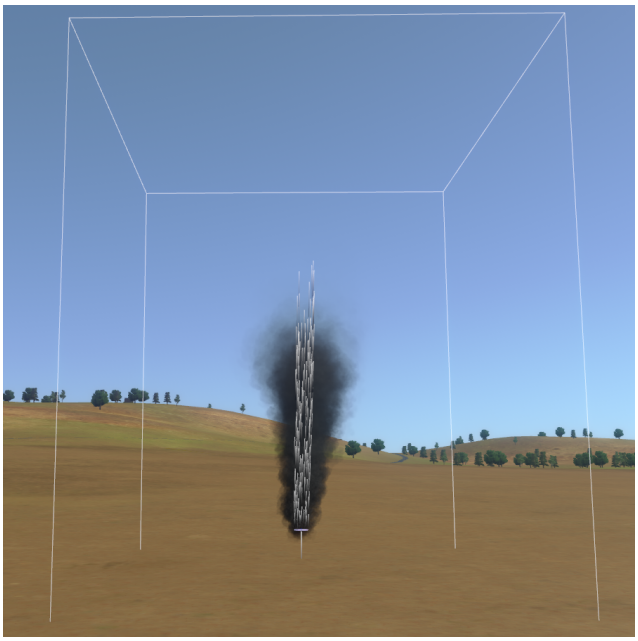
18.6.3 Viewing Debug Visuals

You can visualize these aspects of a particle system:

- Lights
- Bounding boxes
- Emitters
- Forces
- Velocities

They are calculated for each frame.

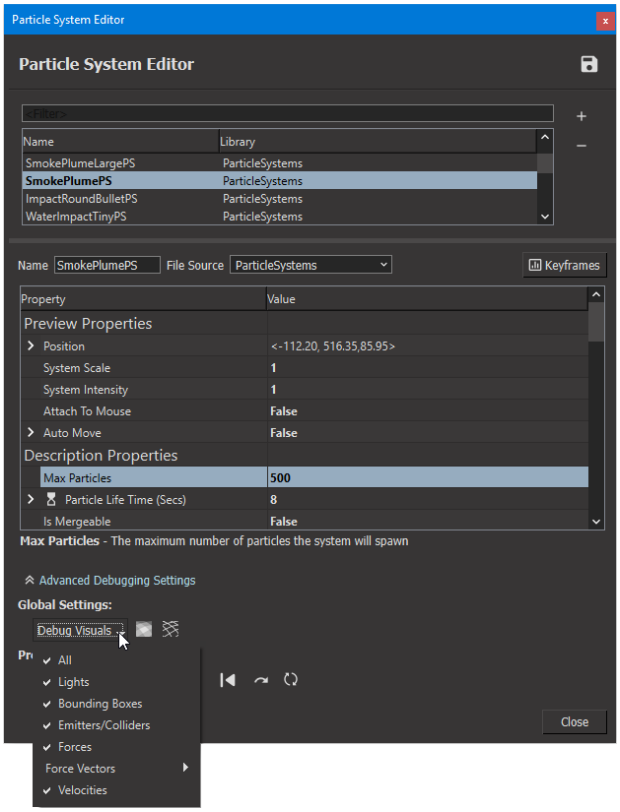
Figure 144. Preview of a smoke plume with all debug visuals enabled



➤ **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 145). You can toggle individual visuals on and off and can toggle all visuals on and off.

Figure 145. Particle System Editor - Debug Visuals



18.7. 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 67 describes the properties.

Table 67. 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 67. 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.
Number To Release	The number of new particles to emit when the system is updated. This is bounded by the release interval.

[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 67. Particle system properties (continued)

Parameter	Description
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.
Rendering: Texture Map	
Texture Map	The texture map to use on the particles. To use your own, add the file to the *./SharedData/19/latest/Particles* directory.

Table 67. 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.
Rendering: 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 67. 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.
Force	Specifies the strength of the force.

Table 67. Particle system properties (continued)

Parameter	Description
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.

18.7.1 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 68. 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.

18.8. 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

Introduction to DI-Guy

DI-Guy Content

Using the Character Viewer

Adding DI-Guy Models

DI-Guy Tutorials

Introduction to 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 Technologies 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.

19. DI-Guy Content

This chapter is an introduction to customizing DI-Guy content.

19.1. Customizing VR-Vantage Content 462
19.1.1 Encrypt Your Final Customized Geometry 462
19.1.2 Compressing Geometry Files 462
19.1.3 Where to Place Customized Files 462
19.1.4 The VR-Vantage Data Directory Structure 463
19.1.5 Understanding DI-Guy Configuration Files 463
19.1.6 Autoload Configuration Files 466
19.2. Customizing Motion 466
19.3. Customizing Texture 466
19.4. Customizing Sound 467
19.5. Creating Custom Hand Items 467
19.5.1 Files Required for a Custom Hand Item 467
19.5.2 Guidelines for Creating Weapon Visuals for Hand Items 469

19.1. Customizing VR-Vantage Content

VR-Vantage is highly customizable. Motions, geometry, textures, characters, and special effects (that is, VR-Vantage content) can all be customized.

Be sure to use the *./custom* directory, as described in this manual, when modifying VR-Vantage. This safeguards your changes to ensure that you can install future releases without losing your customizations. Read "19.1.3 Where to Place Customized Files" below to learn the details of this directory.

> ⚠ When editing configuration files, do not use Notepad. Use a text editor that preserves end-of-line information, such as Notepad++.

19.1.1 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 enter:

```
diguyEncryptGeometryFile filename
```

19.1.2 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 enter:

```
DIGuyGeometryProcessor
```

19.1.3 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 VR-Vantage normally loads *model.flt*, it will first search for the file *./custom/geometry/flt/model.flt* before loading *./custom/geometry/flt/model.flt*.

You can create the *./custom* directory at the top level of your DI-Guy installation or in the DI-Guy subfolder in your *SharedData* directory. For example:

- C:/MAK/diguy#.#/custom
- C:/MAK/SharedData/19/latest/ModelData/Lifeforms/DIGuy/custom
- Another location in your environment and then use the DIGUY_CUSTOM environment variable to point to the root directory of your custom folder, for example, `D:/myGuys/custom`.

Any subfolders you create in the custom folder should mirror the folder structure of the data installation itself. At runtime, a MAK ONE application prioritizes using data from the

./custom directory if configurations of the same name or filename are present there. You can override configurations for existing appearances, hand items, and so on without making changes to the MAK ONE data installation.

- We strongly recommend that you leave all files installed by your MAK ONE application unchanged and place all your modified and new files in a *./custom/* directory. This allows your customizations to be portable to other MAK ONE application installations and helps you upgrade to new versions of your application with minimal or no changes to your custom content.
- If there is a *./custom* directory in both the SharedData directory and your DI-Guy installation and there are files in the same subfolders with the same names, the files in the DI-Guy installation take precedence over those in SharedData. Placing all your custom files in one *./custom* directory is a good practice. If other applications require the *./custom* directory files, SharedData is best.

19.1.4 The VR-Vantage Data Directory Structure

Here is a list of custom directories whose files mask identically named files in the data directory:

- Geometry *./custom/geometry/*
  - Textures *./custom/geometry/rgb/*, *./custom/geometry/png/*
  - Scene Objects *./custom/geometry/sets/*
  - OpenFlight Models *./custom/geometry/flt/*
  - Collada Files *./custom/geometry/dae/*
- Configuration Files *./custom/config/*
- Motions *./custom/motions/*
- Sound Effects *./custom/sfx/*
- Scenarios *./custom/scenarios/*

19.1.5 Understanding DI-Guy Configuration Files

DI-Guy content is declared in and loaded using configuration files. By default, the configuration file *./config/diguy.cfg* is used by VR-Vantage to define DI-Guy content. This file **includes** other configurations files, which may also **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 indentations use the Tab character. If you make edits or a custom file, be sure that you use also use Tab to indent the lines of the file. If there are indentations made with spaces, MAK ONE applications and other applications that use DI-Guy may not load the config file correctly.

These concepts and terminology are used in the DI-Guy configuration file system:

- Actor. The skeleton used for a given character. Motion files reference the original actor. The VR-Vantage 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.

For definitions of other terms used by DI-Guy, see the Glossary in the DI-Guy user documentation.

The entries in *diguy.cfg* can be organized into these 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.

The Main Section

This section of *diguy.cfg* defines most of the DI-Guy content. If you want to customize DI-Guy content, you need to understand this section and the Level Two content it references.

- *diguy/hand_item_shapesets.cfg*. Defines the shapesets for many hand items.
- *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/facefx_shapesets.cfg*. Shapesets for head appearances using the Expressive Faces.
- *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/weapon_supplemental_data.cfg*. Contains weapon supplemental data entries for most of the hand items. These entries define the munition used by each weapon as well as the grip point for the off-hand IK system.
- *diguy/hand_item_appearances.cfg*. Contains the hand item configurations for the majority of the hand items.
- *diguy/default_particle_descriptions.cfg*. Contains the configurations for the standard particle definitions shipped with VR-Vantage.

- *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".
- *diguy/appearance_upgrade.cfg*. Contains upgrade mappings from legacy character types, hand items, and appearances to updated versions that should be used instead. When an application creates a DI-Guy character, the DI-Guy API checks this table and attempts to automatically upgrade any configurations to the newer ones.

The Level Two Files Section

The actor files in this section are the code definitions for the actors (skeletons) used by various character types.

- *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*. These files group common appearances. An appearance consists of its name, the actor with which it is associated, a number of "items" that designate shapesets used by this appearance, and information that helps categorize the appearance for wizard-like queries.

19.1.6 Autoload Configuration Files

Any files placed in the *./custom* directory and its subfolders mask the contents of the same folders in the data directory. These include config, geometry, images, motions, and sfx. This also includes all subfolders, for example, *./custom/geometry/png* and *./custom/geometry/fbx*. For more information on masking the default files with your customizations, refer back to "19.1.3 Where to Place Customized Files" on page 462.

In addition, any configuration file in *./custom/config/* that begins with "*autoload_*" is read automatically during VR-Vantage configuration initialization.

19.2. Customizing Motion

You can customize motion using the VR-Vantage Motion Editor. Please see the *DI-Guy Motion Editor User Guide* for details.

19.3. 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.

ⓘ If you want to customize textures or geometries, you must use the Unencrypted Data Installer with the uncompressed/unoptimized versions of the character geometries and textures. These are files that you are able to open and modify. To learn more and request this installer, contact your MAK distributor or use the Support Portal at https://www.mak.com/mak-one/support/portal.

19.4. 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.

19.5. Creating Custom Hand Items

19.5.1 Files Required for a Custom Hand Item

To define a new hand item, you must create both a `hand_item_appearance` entry and a `shape_set` entry. You should include both entries in a single config file, as both are vital parts of the hand item. These examples apply to soldier_17.

New Hand Item Configuration File

For example, the first file is *./custom/config/autoload_new_hand_item.cfg* and includes the entries:

```
shape_set custom_rifle
  object_type = weapon
  pbr_ready = 1
  filename = golden_gun.fbx
  shape_attachment_data = WeaponNode rifle_joint

hand_item_appearance custom_rifle
  DIS_Septet_category = 32
  type = rifle

  default_items
    item = custom_rifle
    item = small_flash

  parameters
    weapon_supplemental_data = M4A1Carbine
```

The most important part of the configuration is to point the filename for the shapeset to the geometry you want to use for the hand item and make sure the geometry node name is correct in the `shape_attachment_data` entry.

- The geometry node (named `WeaponNode` in the example above) is the node in your geometry file that contains the geometry for the weapon. This can be anything, however, it is important that the name of the node in the geometry file match the name given in the `shape_attachment_data` entry. This ensures that DI-Guy can attach the geometry correctly.
- The second value on that line is the joint to which you want to attach the item (`rifle_joint` in the example above).

The `hand_item_appearance` entry sets up the newly configured geometry as a hand item that DI-Guys can equip and gives it a weapon type as well as some additional supplemental data (for now, simply pointing to the existing M4A1 carbine config). The main configuration of note here is the `item = custom_rifle` entry in the `default_items` section. This tells the hand item to use the shapeset you configured first, and it's important that the item name given here matches the shapeset name.

New Hand Item List Configuration File

The next file is *./custom/diguy/autoload_new_hand_item_list.cfg*, which makes the hand items available to character types (`soldier_17` in the example below). It includes the entries:

```
hand_items_list
  class_type custom_hand_items
    character_types
      character_type = soldier_17
    hand_items
      hand_item = custom_rifle
```

In the first file (new hand item), we configured the shapeset and hand item itself. However, we still need to make the new hand item available to the character types we want to use it with. In this file, we set up a new `class_type` (a collection of hand items), include our new hand item, and make the `class_type` available to the character types we want to be able to use the new hand item.

While we have added only one hand item here, if you are adding multiple hand items, you can simply include multiple `hand_item` entries in the `hand_items` section, each pointing to a new `hand_item`.

Supplemental Weapon File

An optional file is *./custom/config/autoload_weapon_supplemental_data.cfg*, which includes an entry for our custom weapon:

```
weapon_supplemental_data custom_rifle
  lhand_offset_1 = 0.45 0.08 0.06
  rifle:munition_type = m4
```

This file provides the left hand grip point for the weapon as well as the munition type. Depending on how much you change your custom weapon from existing configurations, this may not be necessary. If it is similar enough to another configuration, you could also omit this file and simply point the `weapon_supplemental_data config` line in the `hand_item_appearance` entry to the existing supplemental data entry (for example, M4A1Carbine).

Note that hand offset is important for specifying where the weapon is positioned relative to a place other than the grip. It is easiest to make adjustments in the DI-Guy GUI, rather than trying various numbers if they are not already known.

19.5.2 Guidelines for Creating Weapon Visuals for Hand Items

These guidelines cover the basics of creating the weapon visuals for hand items. They are designed for soldier_17. It is assumed that you understand how to use the tools necessary to create these models.

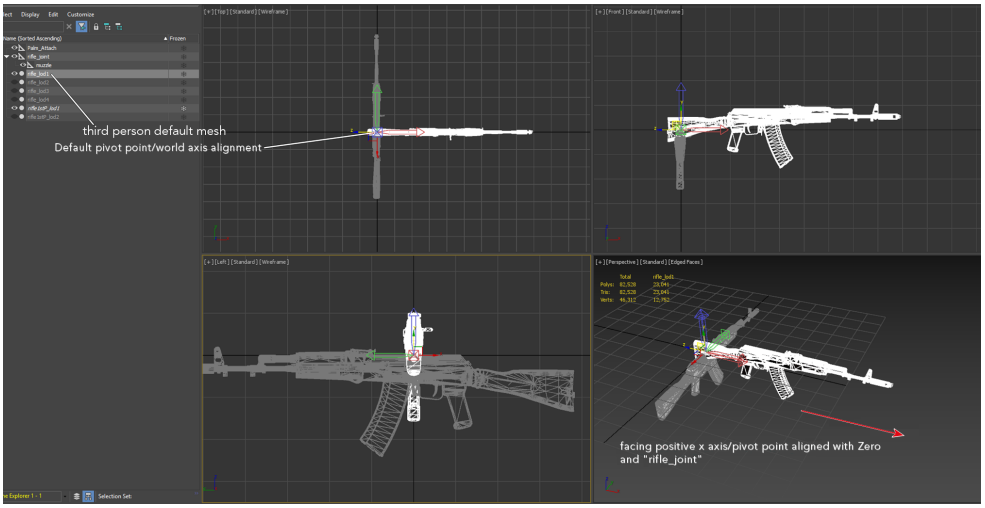
Third-Person

These guidelines are for human characters that will be seen in the third person.

Rifles

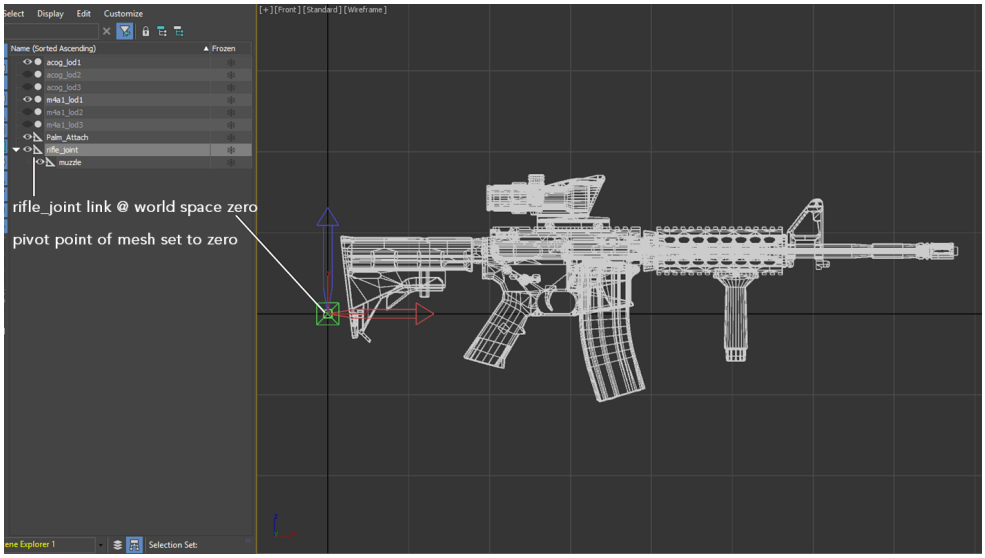
Rifles should be facing down the positive X axis with the Z axis (or Y axis if you use Maya), pointing up relative to the weapon.

Figure 146. Rifle position for third-person character



The geometry must be offset from the origin such that the origin of the model lines up with the rifle_joint position. (For soldier_17, the rifle_joint link is positioned just above the skeleton's right elbow, as show in Figure 148.)

Figure 147. Rifle joint link



The rifle joint to grip point offset depends on the weapon type and the size of the stock. We recommend that you align the trigger with an existing, similar weapon.

Figure 148. Soldier 17 link positions

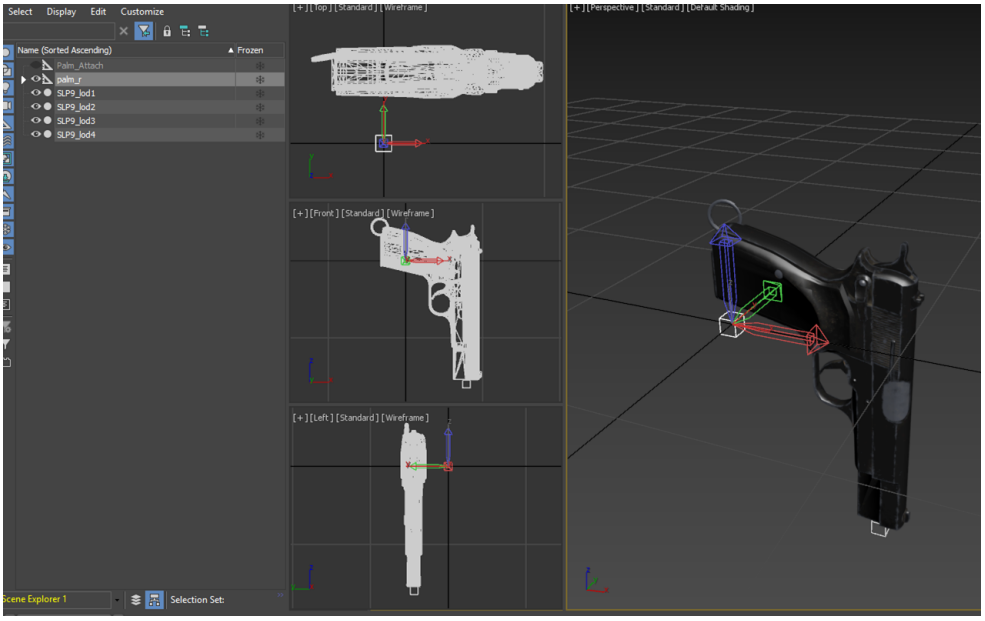
Pistols

Pistols do not attach to the rifle joint but to an additional node, palm_r, with an offset in Y off center.

Pistols should be facing down the Y axis with the X axis pointing up relative to the weapon.

Geometry must have the origin/pivot point at world zero. Align the mesh based on where the trigger grip lines up with an example.

Figure 149. Pistol position for third-person character



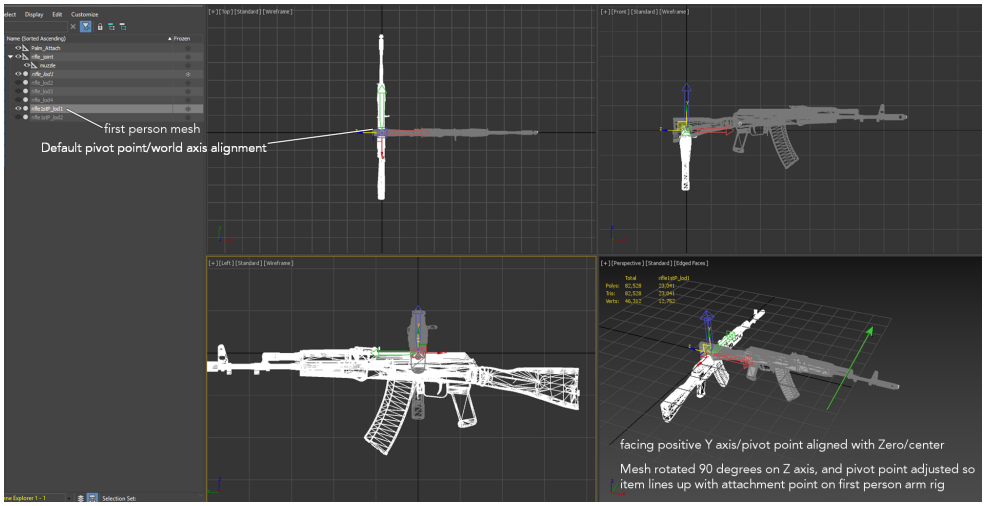
First-Person

These guidelines are for human characters that will be seen in the first person, where role players see the hands and forearms (first-person rig).

Rifles

Rifles should be facing down the positive Y axis (rotate 90 degrees from third person). It does not require any links in export as they are part of the skeletal model for the first-person rig. However, meshes center/pivot point should be centered at world zero once you align the mesh. To align the mesh, line it up with trigger area on example as best as possible. You most likely will need to make further adjustments once it is visible in the sim engine.

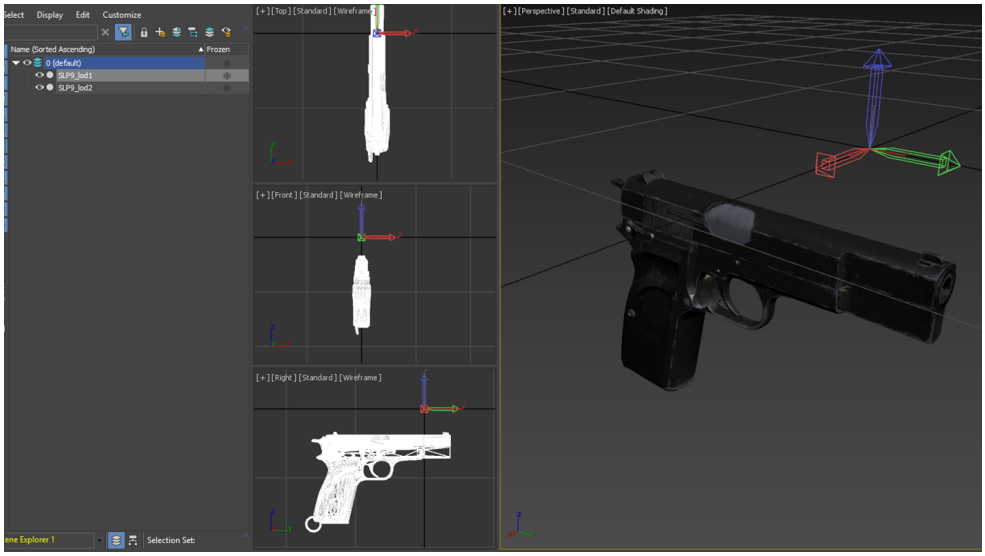
Figure 150. Rifle position for first-person rig



Pistols

Pistols should face down the positive Y axis, similar to the rifle except lowered slightly below the zero plane to align with the attachment node.

Figure 151. Pistol position for first-person rig



20. Using the Character Viewer

This chapter explains how to use the Character Viewer to view characters, appearances, and actions.

20.1. Introduction to the DI-Guy Character Viewer 474
20.2. Viewing Characters, Appearances, Gestures, and Actions 475
20.2.1 Filtering the Character Type and Appearance Lists 476
20.2.2 Changing the Viewing Angle for a Character 477
20.2.3 Demonstrating a Character's Aim and Locomotion Capability 477
20.2.4 Viewing Different LOD Levels 478
20.2.5 Changing How a Character Is Rendered 479
20.2.6 Viewing Multiple Appearances at One Time (Army Mode) 481
20.2.7 Advanced Character Editing 484
20.2.8 Reloading Characters that Have Been Edited 486
20.2.9 Controlling Character Movement 486
20.2.10 Viewing Information About Character Movement and State 486
20.3. Improving Performance 487
20.3.1 Displaying the Performance Profiler 488
20.4. Running a Lua Script 488
20.5. Viewing Character Geometry Details 488
20.6. The Appearance Editor 489
20.6.1 Editing Configuration File Text 490
20.6.2 Editing Appearance Effects 491
20.7. Using the PBR Material Tester 491
20.7.1 Reloading Shaders 492
20.8. The Character Viewer Log Window 493
20.8.1 Setting the Notification Level 493
20.9. Broadcasting the Character Viewer over the Network 493
20.9.1 Joining a DIS Exercise 493
20.9.2 Joining an HLA Exercise 495

20.1. 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 windows in Character Viewer are:

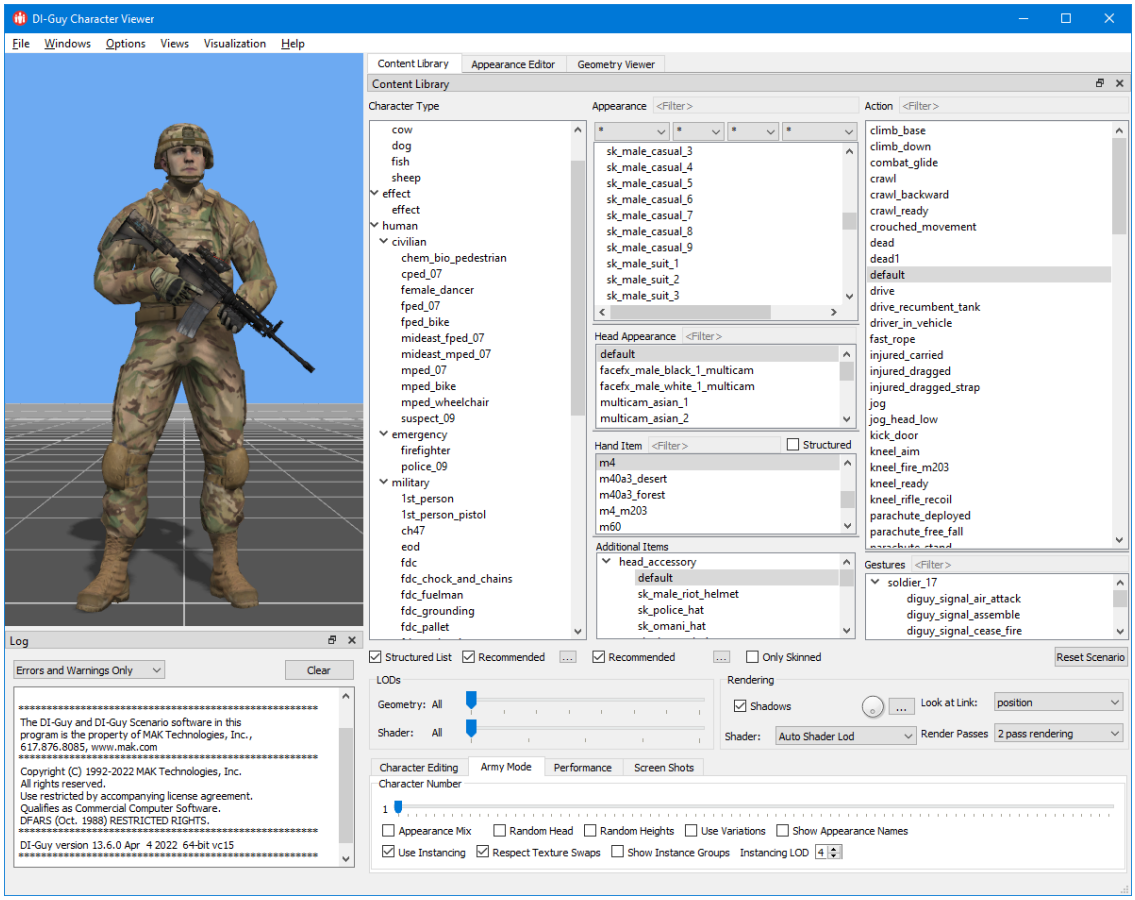
- 3D window. Displays the selected character, appearance, and action.
- Content Library. The Content Library lets you select the character appearances, gestures, and actions you want to view. It also has advanced features for experimenting with lighting and geometry.
- Appearance Editor. The Appearance 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 >** and then the **Tools** folder for the MAK ONE application. In the Tools folder, double-click the DI-Guy Character Viewer shortcut. The Character Viewer opens (Figure 152).

Figure 152. Character Viewer



20.2. 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 lists).
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 any items, they are listed in the Additional Items list.

6. Optionally, select a head appearance.
7. Optionally, select one or more items.
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.

20.2.1 Filtering the Character Type and Appearance Lists

You can filter the Character Type and Appearance Lists:

- 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.

The Recommended check box and browse button for the Appearance list display on the same row of the GUI as those for the Character Type. As a result, it might not display immediately below the Appearance list. Those options do not affect the other lists (Head Appearance, Hand Item, or Additional Items).

20.2.2 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 153).

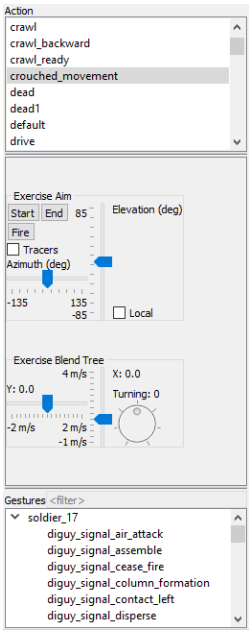
Figure 153. Views menu

- 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.

20.2.3 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 154). For some movement actions, you can vary the blend tree settings.

Figure 154. Exercise Aim panel



➤ **To demonstrate aim and locomotion capability:**

1. Select an aim action or an action that uses a blend tree. A panel is 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.

20.2.4 Viewing Different LOD Levels

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.

You can view a character at the different LOD levels available and you can view different quality of shaders.

➤ **To change the LOD level:**

- In the LODs group box, adjust the Geometry slider to see the different LODs for visual quality.
- In the LODs group box, adjust the Shader slider to see the different LODs for shaders.

➤ **To view all LOD levels for a character:**

1. Choose **Visualization > Show All LODs**. All LOD levels for the character display in the 3D viewer.
2. Adjust the camera to view the different LOD levels.

Figure 155. Showing all LOD levels

20.2.5 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:**

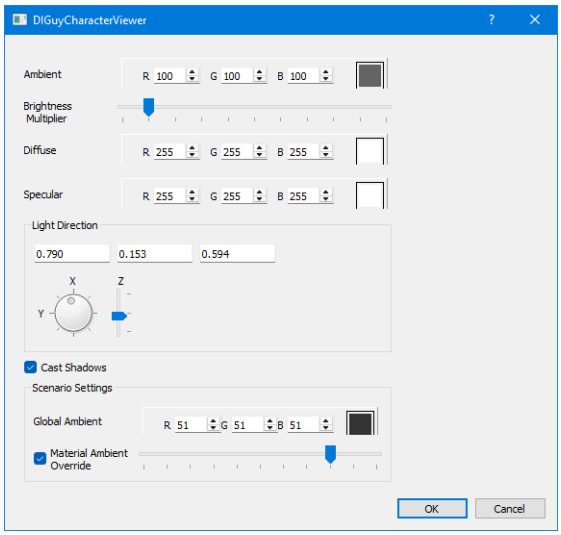
1. In the Rendering group box, select the Shadows check box to enable shadows.
2. Rotate the lighting widget ( ) to change the direction from which light is applied to the character (and therefore, the direction of the shadow).

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 156).

Figure 156. Lighting dialog box



2. Change the color of ambient, diffuse, specular, or global ambient light by changing any of the RGB values or by clicking the color swatch and selecting a different color.
   - Ambient light is general background light in the entire scene.
   - Diffuse light is that which is reflected off of rough surfaces. It is reflected in many directions.
   - Global Ambient is the global amount of ambient light.
   - The Material Ambient Override slider adjusts the amount and color of ambient lighting values as they're applied to the DI-Guy character materials. This is in addition to the Ambient and Global Ambient settings and applies only to DI-Guy characters, whereas those other settings apply to the entire scene.
3. Adjust the Brightness Multiplier slider to increase the brightness of the diffuse and specular light.
4. In the Light Direction group, change both the direction and altitude of the light source.
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, choose **Visualization > Wireframe**.

➤ To view a character's skeleton, choose **Visualization > Skeleton**.

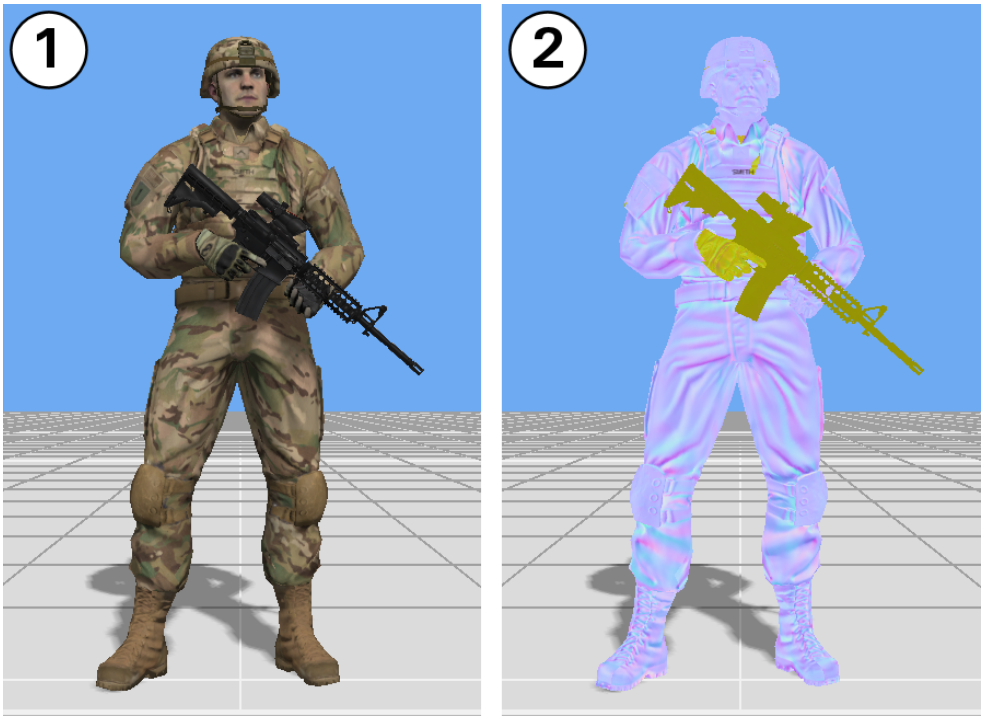
➤ To view the link names, choose **Visualization > Link Names**.

➤ To view the connection names, choose **Visualization > Connection Names**.

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 157 shows two different shader settings.

Figure 157. Shader debugging: Auto Shader Lod (1) and Normal Map (2)



➤ To change the shader being used, in the Rendering group box, select an option from the Shader list.

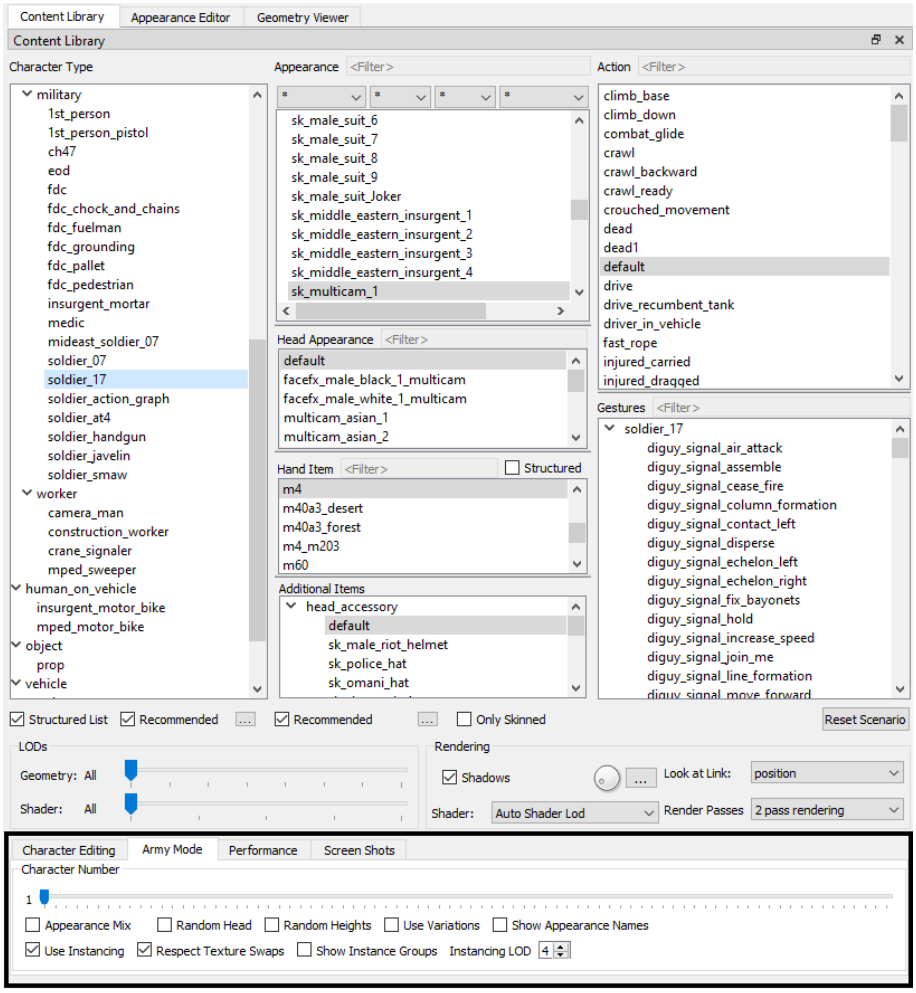
20.2.6 Viewing Multiple Appearances at One Time (Army Mode)

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 158).

Figure 158. Army mode



2. Drag the Character Number slider to increase the number of characters displayed in the viewer window (Figure 159).

Figure 159. Multiple character instances (army mode)

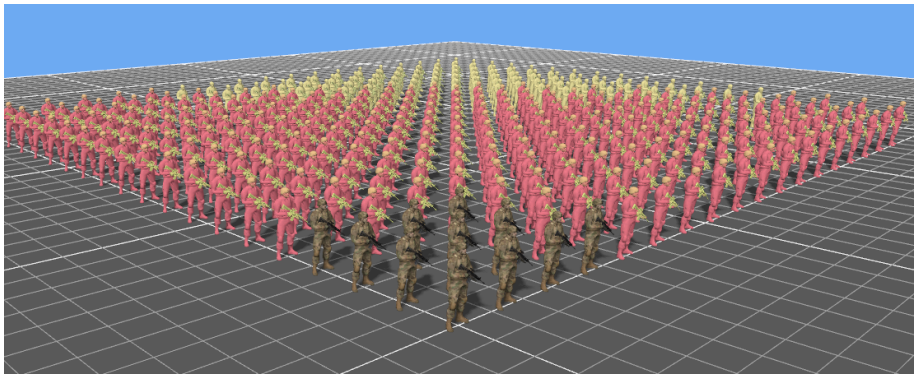
3. Select the desired check boxes to change the variations in appearance of the characters in the viewer.

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 "20.3.1 Displaying the Performance Profiler" on page 488.)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 160).

Figure 160. Instance groups with Instancing LOD set to 4



- Instancing LOD. Specifies the LOD at which instancing turns on.

➤ **To enable instancing:**

1. Select the Army Mode tab.
2. Select Use Instancing.

20.2.7 Advanced Character Editing

The 3D window can display the character bounds (Figure 161 and Figure 162), inverse kinematics (Figure 163), aim vectors (Figure 164), LODs, and more. You can also see how a character looks with different rending approaches.

➤ To view a character's bounds or bounding box, choose **Visualization > Character Bounds**, **Local Bounding Box**, or **Link Bounding Box**.

➤ 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.

Figure 161. Character bounds

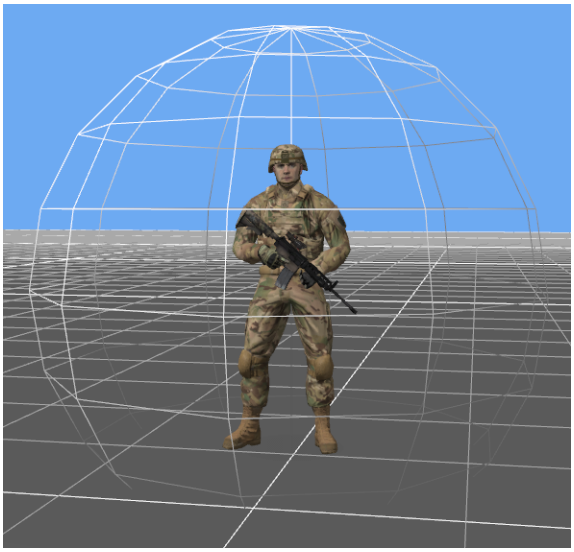


Figure 162. Local bounding box and Link bounding box

Figure 163. Inverse kinematics

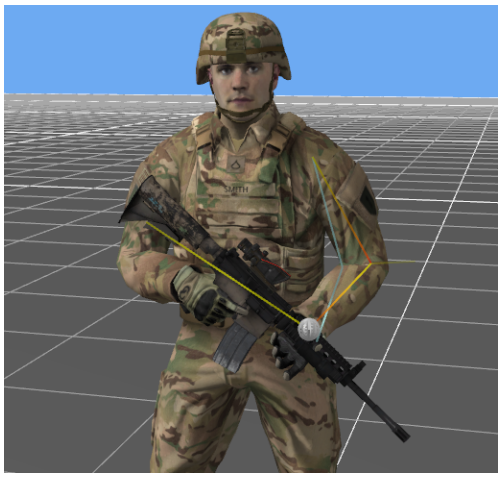


Figure 164. Aim vector

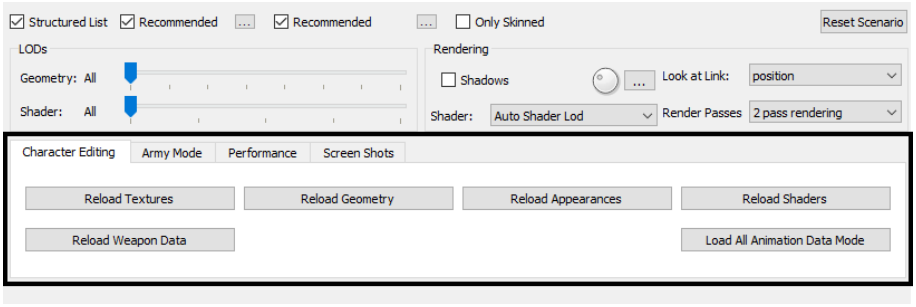
20.2.8 Reloading Characters that Have Been Edited

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 165).

Figure 165. Character Editing Tab



2. Click the button for the type of data that you want to reload.

20.2.9 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 **Windows > 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.

20.2.10 Viewing Information About Character Movement and State

You can view information about the Action the character is currently performing using the Movement Info and State Info options in the Visualization menu.

- Movement Info. Displays arrows that show the character's movements.
- State Info. Displays a label over the character's head with detailed information.

Figure 166. Soldier 17 displaying Movement and State Info

20.3. 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 Developer Debug Overlay.
- Visualize Load Management. Enables debug visuals in the 3D viewer when you make changes to the Load Manager Settings in the Developer Debug Overlay. These visuals reflect the size of the load management zones for the different update rates.
- Activate Tracy Trace. Allows the Tracy diagnostic tool to connect to the Character Viewer.
- Tracy Capture CPU/Stack. Enables Tracy's "Call Stack Sampling" option. Sampling is performed at 8 kHz on Windows and 10 kHz on Linux.

For information about Tracy and MAK ONE products, please see the *VR-Vantage User Guide* or *VR-Forces User Guide*. To learn more about the Tracy Client, read the documentation at https://bitbucket.org/wolfpld/tracy/downloads.

➤ **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.

20.3.1 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.

20.4. 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 **Windows > Script Editor**. The Code Editor dialog box opens.
2. Enter 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:**

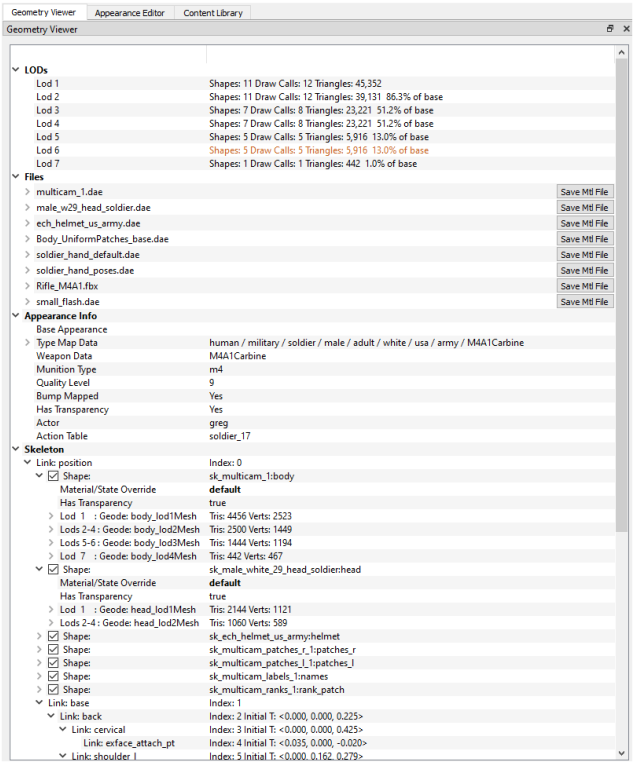
1. Click the Save button. The Save Script window opens.
2. Navigate to the folder where you want to save the file.
3. Click Save.

20.5. 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 167).

Figure 167. Geometry Overview



20.6. 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.

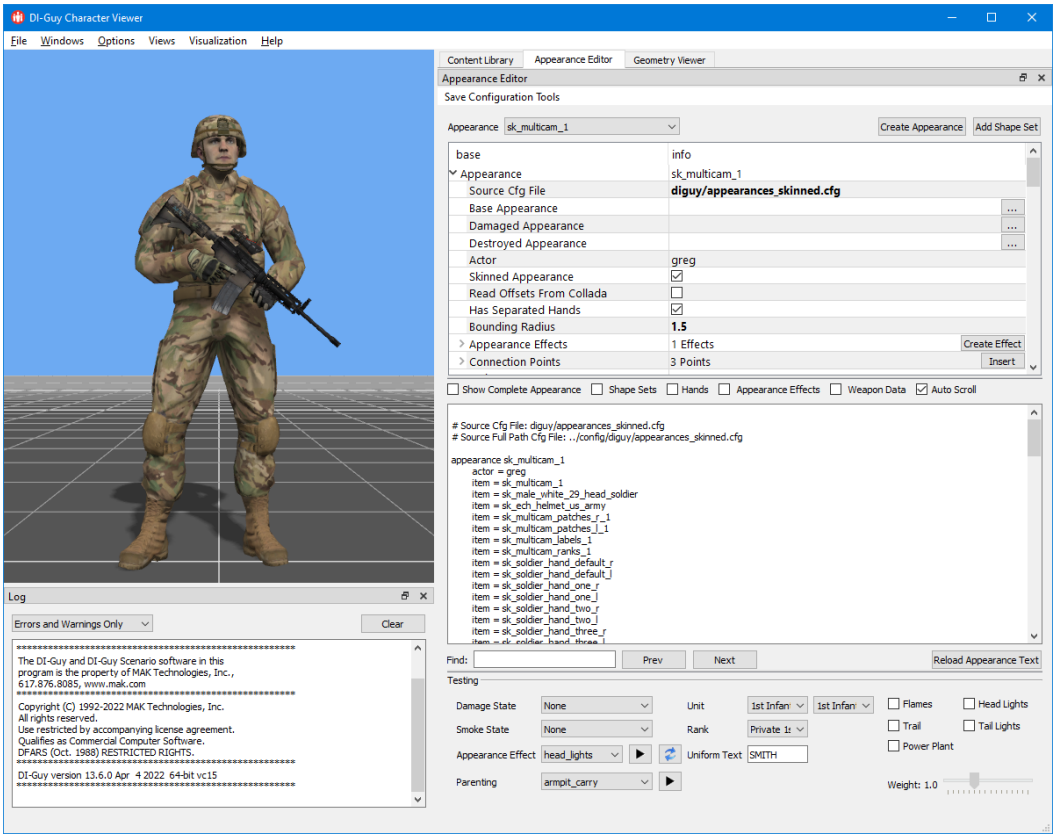
Although the Appearance Editor allows you to edit configuration files without touching the actual files, it also displays the contents of the configuration files, so that you can edit them directly if you want to. Sometimes it is quicker to edit the file, particularly when you want to delete a large 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 168).

Figure 168. 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.

20.6.1 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 do so, you can refresh the editing GUI.

You can control the amount of detail shown in the editing window:

- 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 show hand items, select the Hands check box.

➤ To show information about the weapon (if any), select the Weapon Data check box.

➤ To refresh the editing GUI, click Reload Appearance Text.

20.6.2 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.

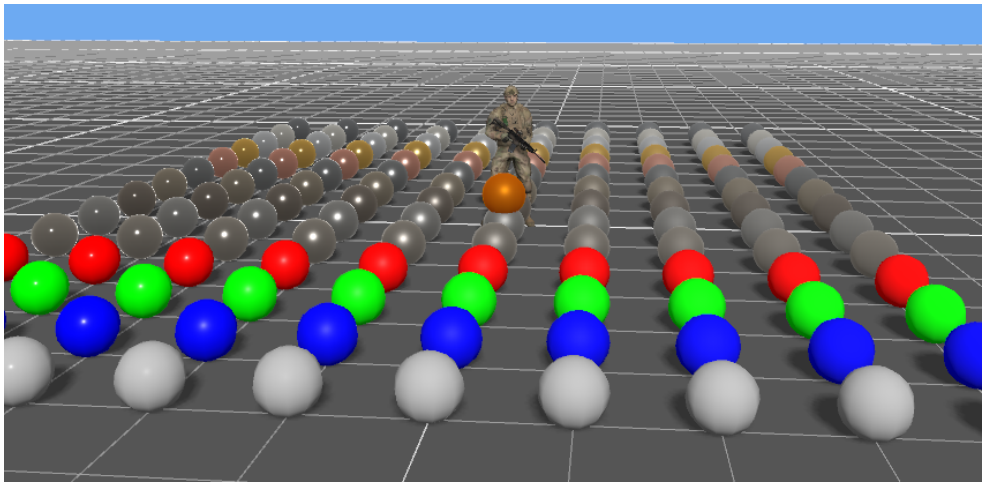
➤ To return to editing the appearance, clear the Appearance Effects check box.

20.7. 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 these shader characteristics:

- Color
- Metalness
- Specular reflection
- Reflectivity
- Diffuse
- Shininess
- Ambient
- Emission
- Alpha.

Figure 169. PBR Material Tester

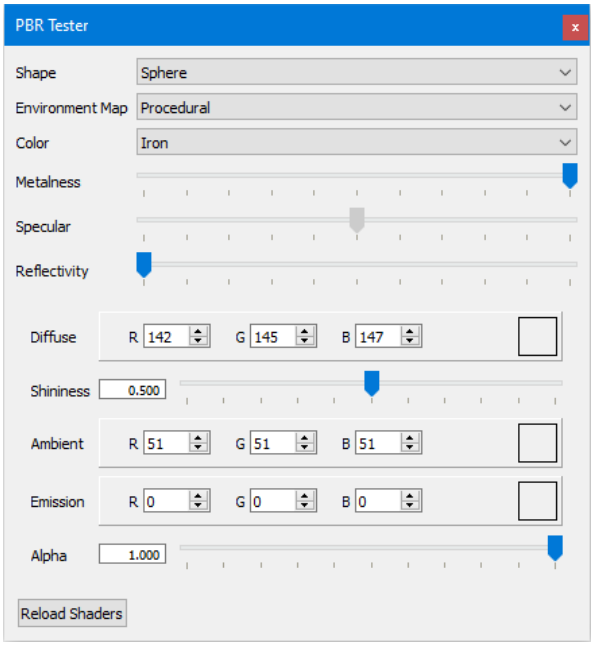


If you change shaders offline, you can reload them from the PBR Tester dialog box.

➤ **To use the PBR Material Tester:**

1. Choose **Windows > PBR Material Tester**. A grid of spheres is added to the window and the PBR Tester dialog box opens (Figure 170).

Figure 170. PBR Tester dialog box



2. Adjust settings as desired.

20.7.1 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 170 on the previous page).
2. Click Reload Shaders.

20.8. 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 **Windows > Log**.

20.8.1 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.

20.9. Broadcasting the Character Viewer over the Network

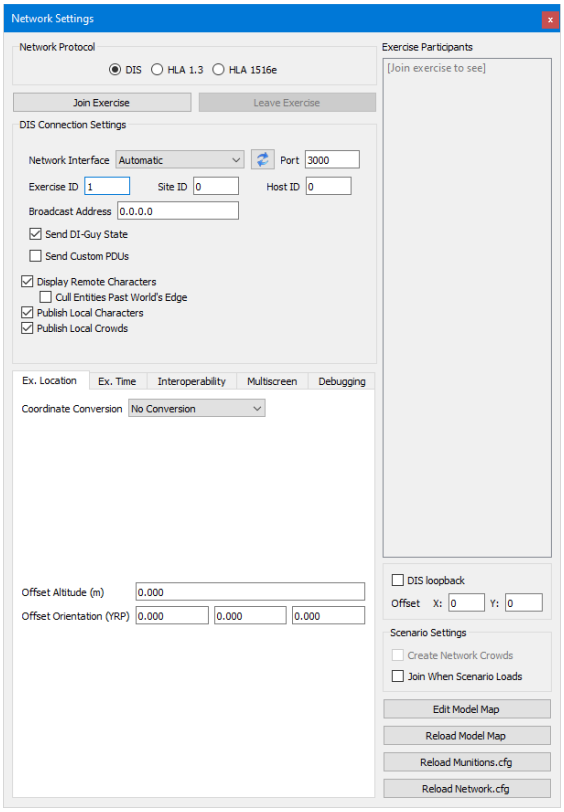
You can broadcast the state of the character in the Character Viewer over DIS or HLA. This allows you to view the character within the context of the terrain in an image generator or similar 3D viewing application.

20.9.1 Joining a DIS Exercise

➤ **To join an exercise using DIS:**

1. Choose **Windows > Network Settings**. The Networking Settings dialog box opens.
2. In the Network Protocol group box, select DIS (Figure 171).

Figure 171. Networking Settings dialog box for DIS



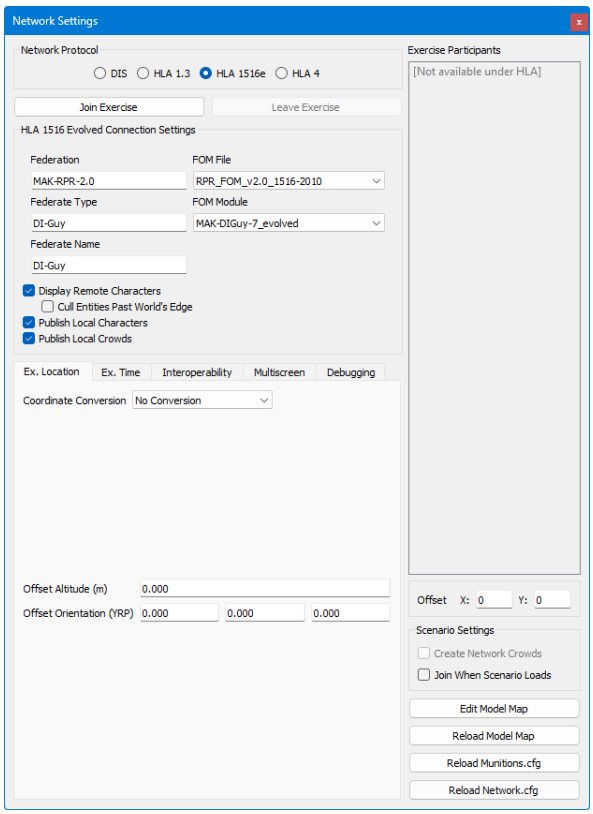
3. In the Network Interface list, select the network interface address that matches the one you are using for your IG's DIS connection.
4. In the Exercise ID box, enter the number of the exercise you want to join. Exercise 0 connects you to all DIS traffic on the network. Exercise 1 or above connects 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 VR-Vantage application.
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. In your VR-Vantage application, the entity in the Character Viewer should be added to the Objects List.

20.9.2 Joining an HLA Exercise

➤ **To join an exercise using HLA:**

1. Choose **Windows > Network Settings**. The Networking Settings dialog box opens.
2. In the Network Protocol group box, select one of the HLA options (Figure 172).

Figure 172. 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.

21. 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.

21.1. How MAK ONE Applications Map DI-Guy Models 498
21.2. Adding New DI-Guy Models to VR-Forces 498
21.2.1 Copy DI-Guy Files to the Data Directory 498
21.2.2 Add a New DI-Guy Entity in the Simulation Object Editor 499
21.2.3 DI-Guy Configuration Files 499
21.3. Adding New DI-Guy Models to VR-Vantage 500
21.3.1 Copy DI-Guy Files to the Data Directory 500
21.3.2 Add Model Definitions and Element Definitions 501

21.1. 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.

21.2. 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:

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.

21.2.1 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.

21.2.2 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.

21.2.3 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 VR-Vantage and the VR-Forces GUI. 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 GUI, 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 GUI, see "DI-Guy Animation", "DI-Guy Characteristics (Appearance, Head, Body Weight)", and "DI-Guy Hand Item" in the *VR-Forces User 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 User 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")
    )
...
```

21.3. 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:

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.

21.3.1 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.

21.3.2 Add Model Definitions and Element Definitions

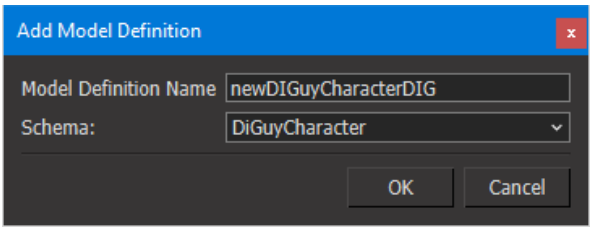
To visualize a DI-Guy character in MAK ONE applications, you must add a model definition and element definition for each character.

Create a DI-Guy Model Definition

➤ **To create a model definition for a DI-Guy:**

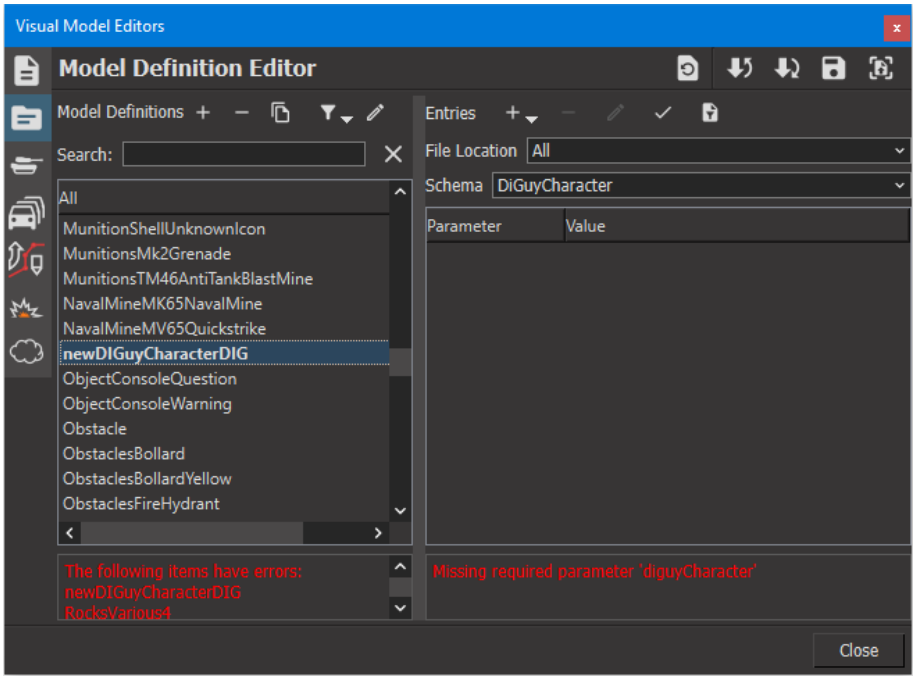
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page (Figure 86 on page 304).
3. Click the Add button (+) next to the Model Definitions label. The Add Model Definition dialog box opens.
4. Enter a name for the new model definition. It must end with the letters DIG.
5. In the Schema list, select DiGuyCharacter (Figure 173).

Figure 173. Add Model Definition dialog box



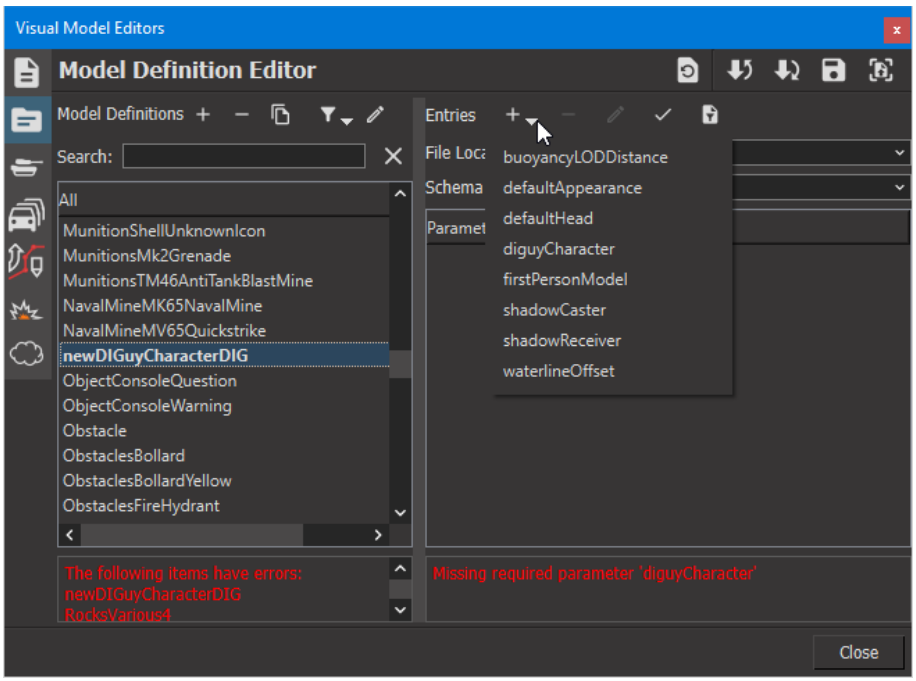
6. Click OK. The new model definition is added (Figure 174).

Figure 174. New model definition



7. Click the Add Parameter button (). A list of parameters displays (Figure 175).

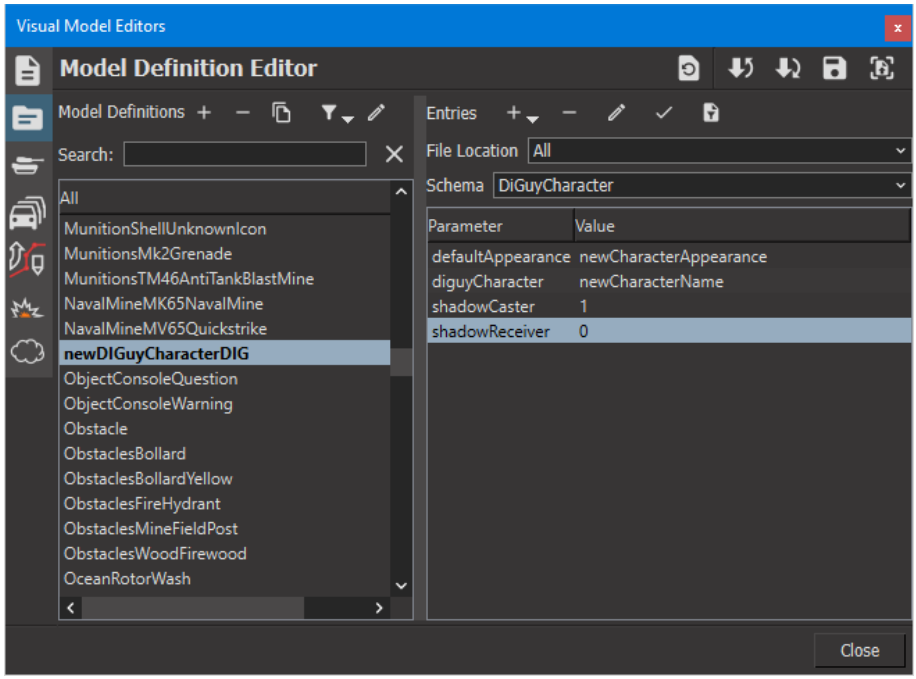
Figure 175. Add parameter list



8. Select defaultAppearance. It is added to the parameter list.
9. Click the value for the parameter. The field becomes editable.
10. Enter the default appearance name for this character (Figure 176).

11. Add a **diguyCharacter** parameter.
12. Click the value for the parameter. The field becomes editable.
13. Enter the character name for this character.
14. Add the **shadowCaster** and **shadowReceiver** parameters (Figure 176).
15. Export the new model definition to a new file.

Figure 176. DI-Guy character parameters

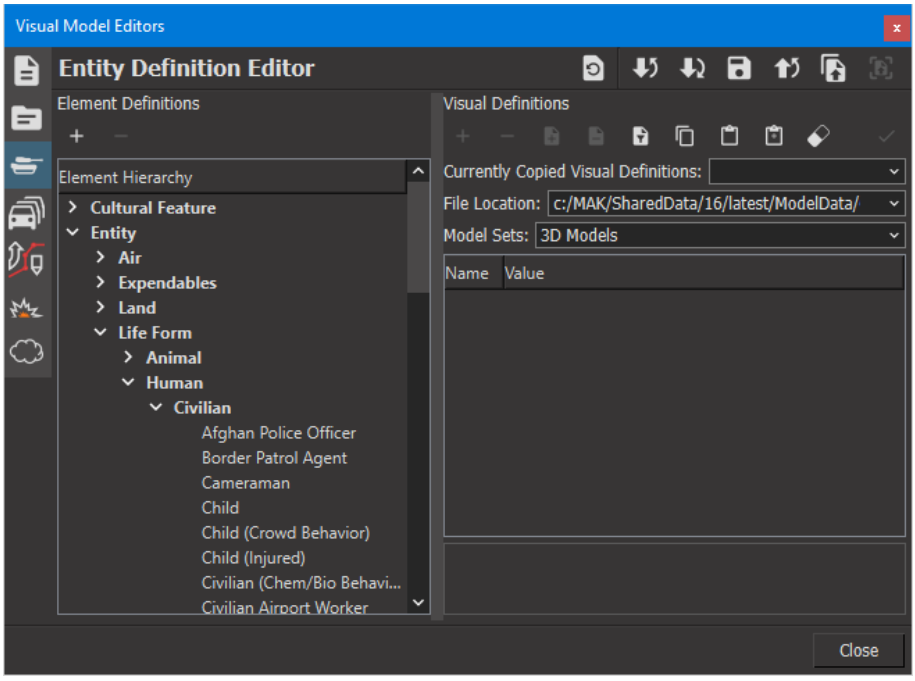


Add an Element Definition

➤ **To create an element definition:**

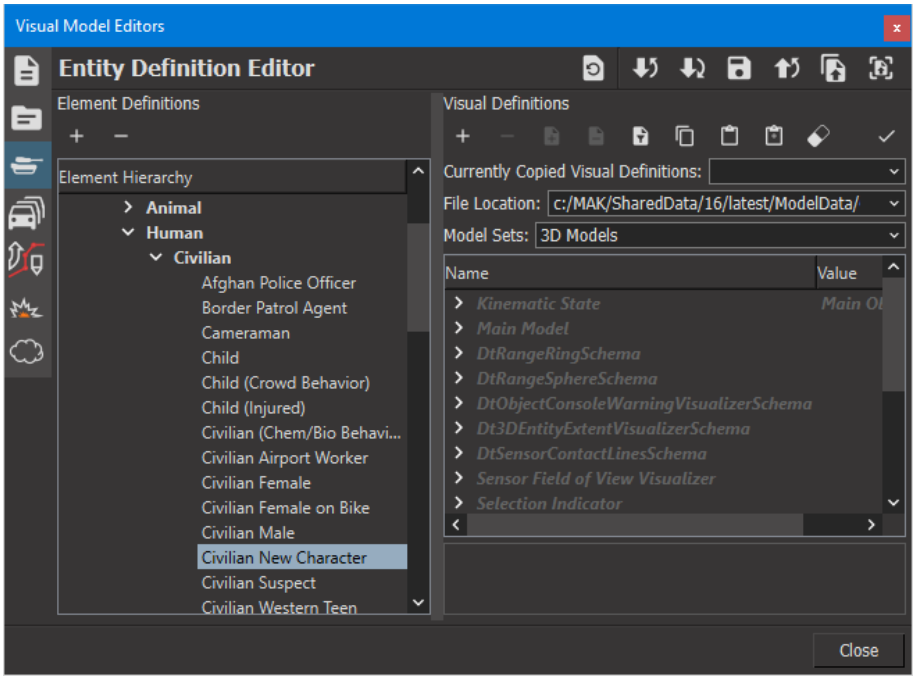
1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Entity Definition Editor page.
3. Expand the element definition hierarchy to the Life Form level and select the type of lifeform you are adding (Figure 177).

Figure 177. Entity Definition Editor page



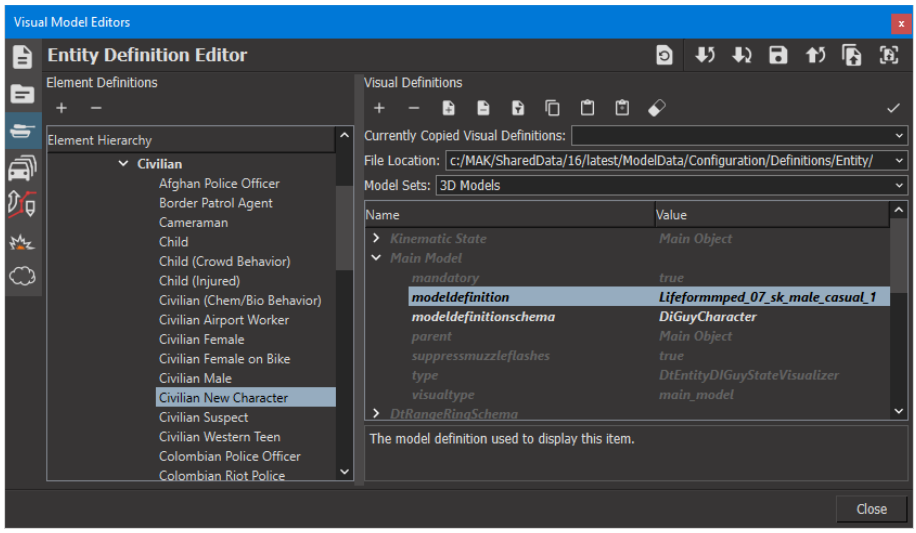
4. Click the Add button (+). The Create Element Definition dialog box opens.
5. Enter a name for the element definition. (You cannot edit the name after you create it, so try to avoid typing errors.)
6. Click OK. The new element is added to the list (Figure 178).

Figure 178. New entity definition added



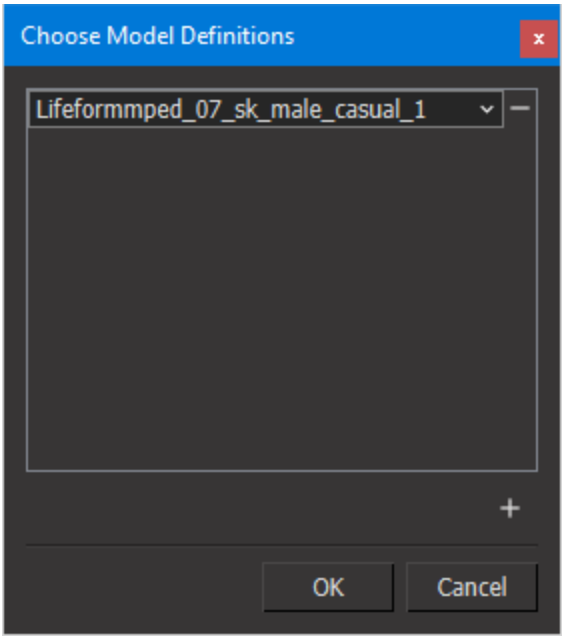
7. In the Visual Definitions list, select Main Model (Figure 179).

Figure 179. Main Model visual definition



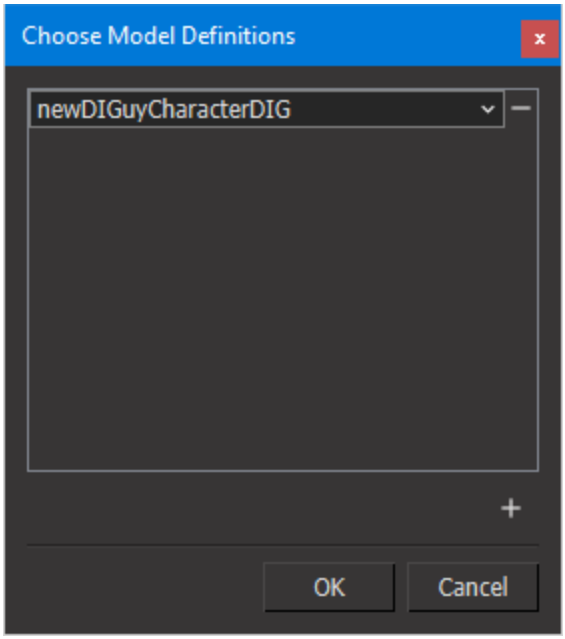
8. Click the Value for the **modeldefinition** attribute. The Choose Model Definition dialog box opens (Figure 180).

Figure 180. Choose Model Definition dialog box



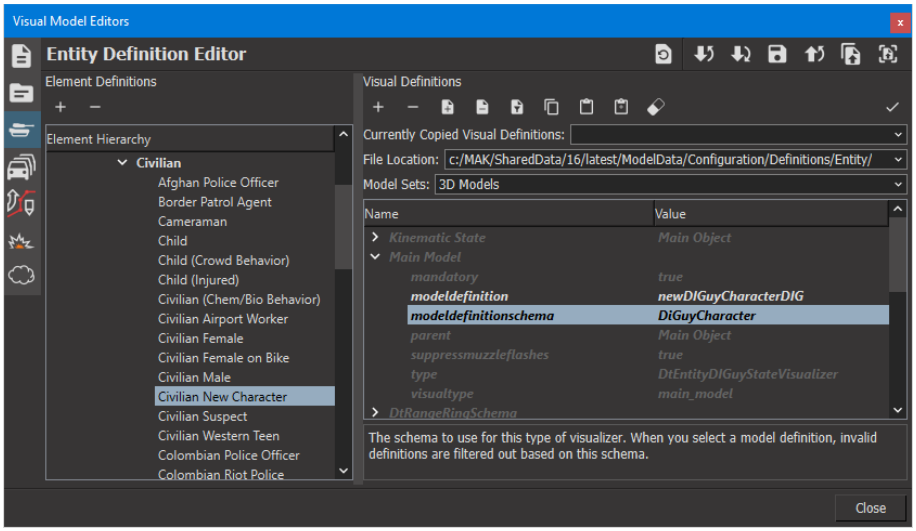
9. Select the model definition that you created for this character (Figure 181).

Figure 181. New model definition selected



10. Click OK. The model definition is updated (Figure 182).

Figure 182. 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.

22. DI-Guy Tutorials

This chapter has examples of customizing DI-Guy content.

22.1. Customizing Appearances Using Texture Swaps 508
22.1.1 Select an Appearance to Modify 508
22.1.2 Change the Texture Map 512
22.1.3 Create Your New Texture Swap and Declare Your New Appearance 512

22.1. Customizing Appearances Using Texture Swaps

Perhaps the easiest and most popular customization is to simply take an existing VR-Vantage 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.

VR-Vantage 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. This tutorial guides you through the process of creating a new appearance.

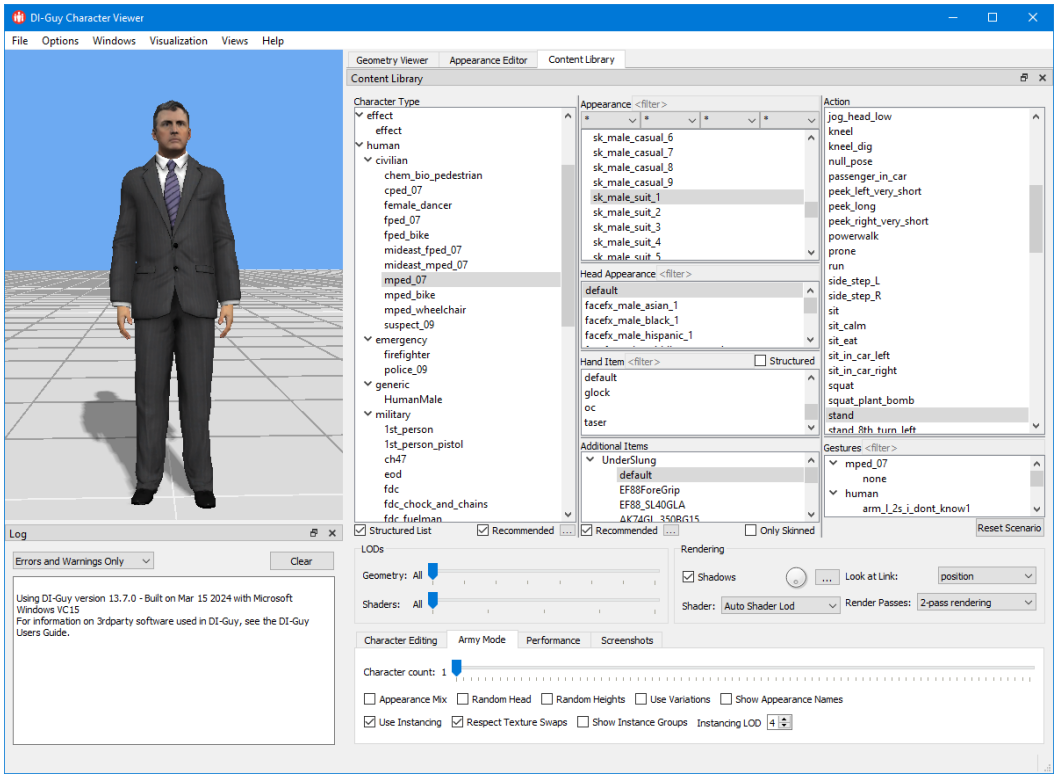
Figure 183. Texture swapping to create different appearances

This tutorial references *.dae* files. The *.dae* files are in the DI-Guy data installer.

22.1.1 Select an Appearance to Modify

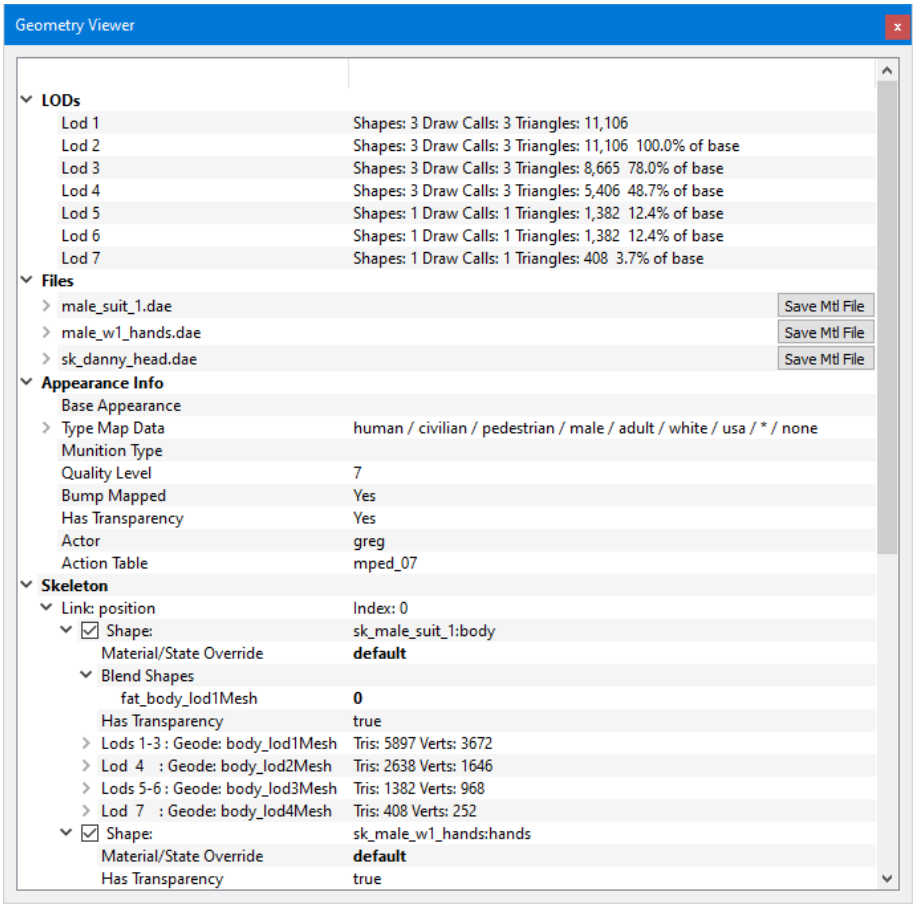
1. On the Start menu, choose **MAK Technologies** and then the **Tools** folder for the MAK ONE application. In the Tools folder, double-click the DI-Guy Character Viewer shortcut. The Character Viewer opens.
2. In the Character Type list, select mped_07.
3. In the Appearance list, select sk_male_suit_1.
4. Hold down the left and right mouse buttons, and rotate the character so that it faces you (Figure 184).

Figure 184. Character viewer



5. Select the sk_male_suit_5 appearance to see the differences.
6. If the Geometry Viewer window is not visible, choose **Windows > Geometry Viewer**. The Geometry Viewer window opens (Figure 185). (It may be tabbed.)

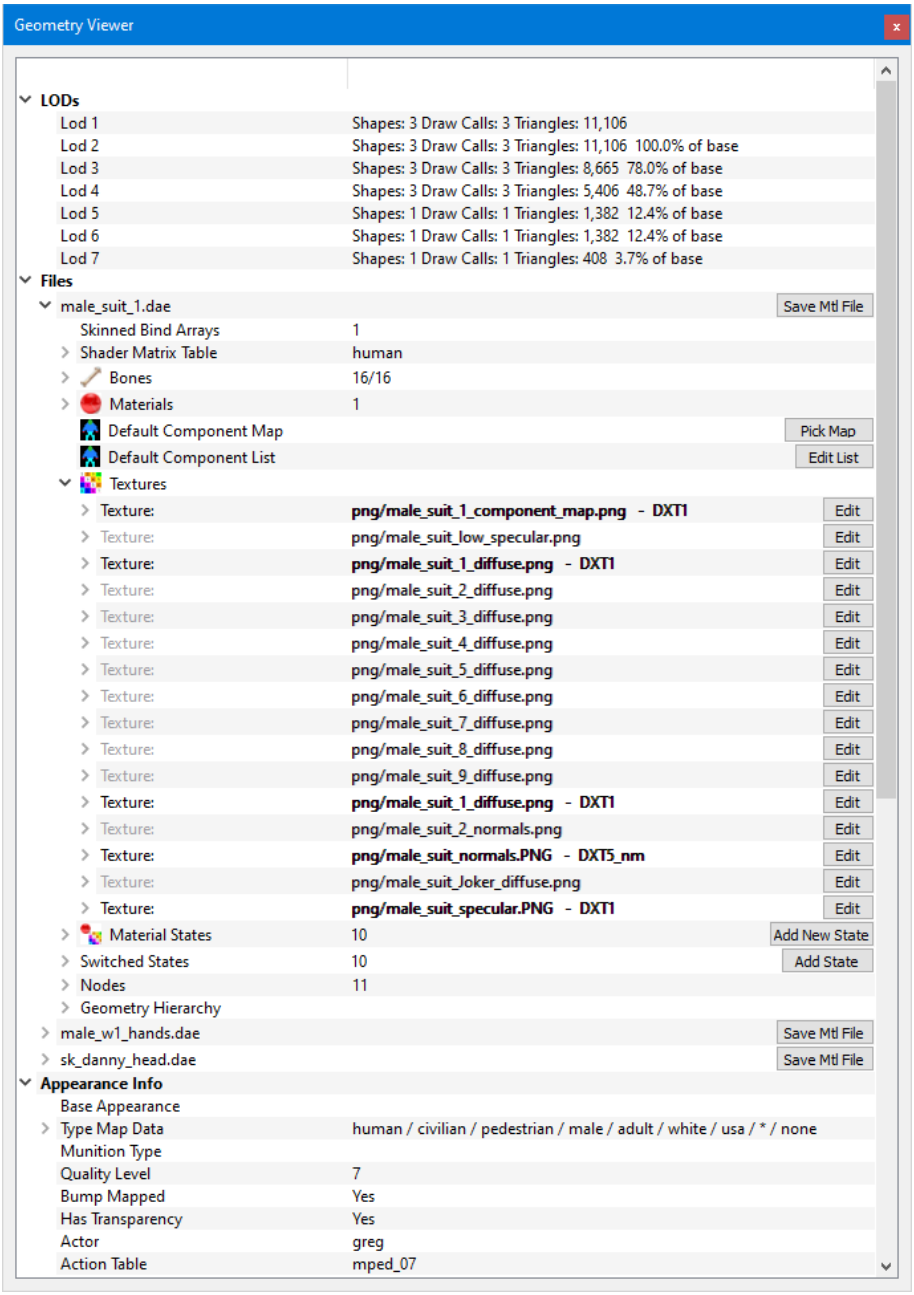
Figure 185. Geometry Overview window



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 186. male_suit_1 textures



8. We need to make a copy of the male_suit_1 configuration file. However, to do that, VR-Vantage requires that the *.dae* file be in the custom directory.
   a. In *./custom/geometry*, create a new directory called *dae*.
   b. Copy *./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.

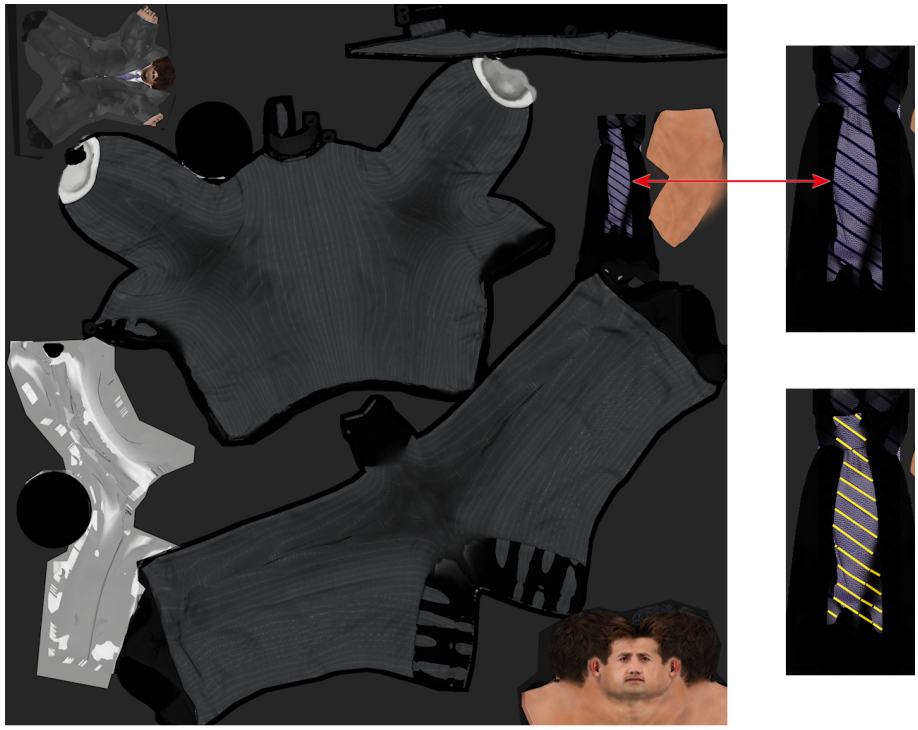
10. Click Yes. The file *./custom/geometry/dae/male_suit_1_mtl.cfg* is created.
11. Use a text editor to open the file *male_suit_1_mtl.cfg*. Near the top of the file, are some texture entries. One of them is *male_suit_1_diffuse.png*. We will edit this file.

22.1.2 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/facefx/male_suit_1_diffuse.png* (Figure 187). The tie is in the upper right quadrant of the file.

Figure 187. ***male_suit_1_diffuse.png***



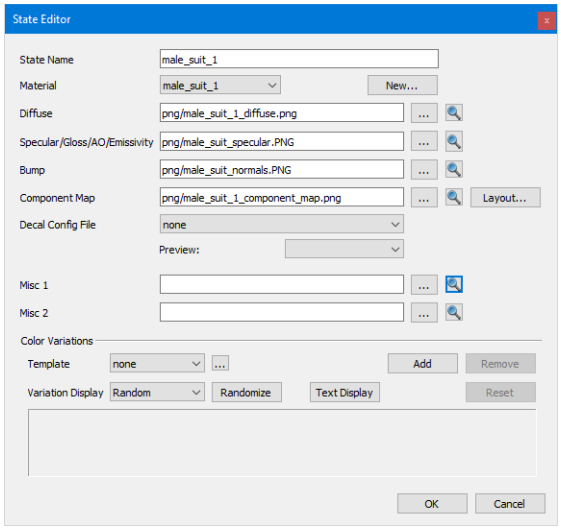
2. Change the tie's stripes to yellow.

   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*.

22.1.3 Create Your New Texture Swap and Declare Your New Appearance

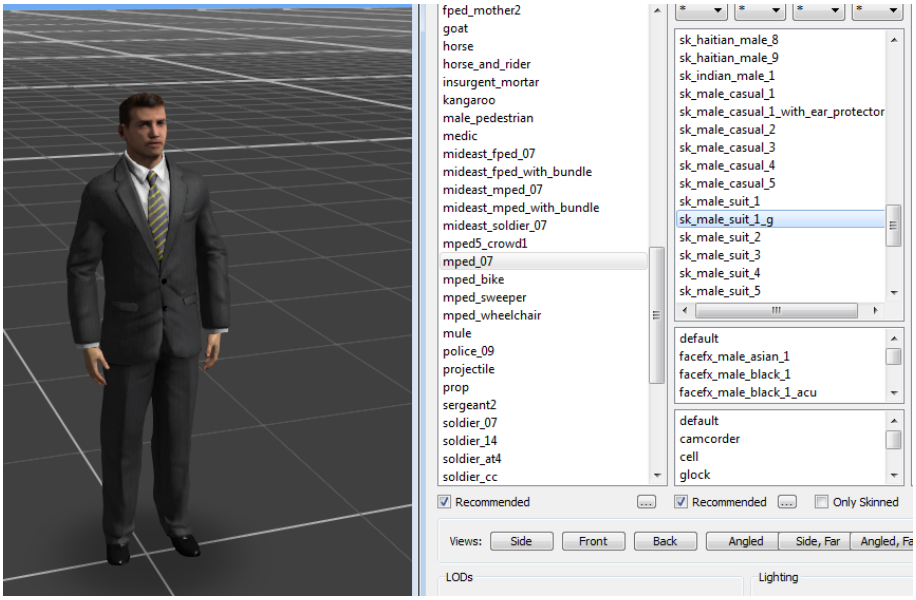
1. In the Geometry Viewer, in the Material States line for male_suit_1.dae, click Add New State. The State Editor opens (Figure 188).

Figure 188. State Editor



2. Set the State Name to "male_suit_1_g".
3. In the Diffuse field, click the browse button and then navigate to *./custom/geometry/png/male_suit_1_g_diffuse.png*.
4. Click Open.
5. In the State Editor, click OK. The *male_suit_1.dae* is highlighted and has an asterisk.
6. Click Save Mtl File. You are prompted to save to the custom directory.
7. Click Yes.
8. Create a new appearance configuration entry for your character:
    a. In a text editor, open *./config/diguy/appearances_skinned.cfg*.
    b. Copy the entire multi-line entry for appearance sk_male_suit_3 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 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.
9. Restart the Character Viewer. The new appearance should now be available with a beautiful new tie (Figure 189). Compare it to Figure 184 on page 509.

Figure 189. Character with modified tie



MAK Products Glossary

This glossary defines terms used in MAK ONE product documentation.

A

absolute dead reckoning

A method of dead-reckoning in which an application uses the timestamps contained within state updates if the sender is using absolute timestamping. 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

See 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

See 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 ONE 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 MAK Products Glossary. VR-Link applications also read the FED file.

federate

A connection to the MAK Products Glossary. 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 and HLA 4, 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 ONE 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 message describing a simulation event. An interaction describes an event, it does not update an objects state.

L

Logger control PDUs

A set of DIS PDUs or HLA interactions that let you control the MAK Data 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

MÄK Technologies Lisp

An adaptation of the Lisp language used in configuration files for MAK ONE 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 MÄK Technologies Lisp.

N

NETN FOM

The NETN FOM (NATO Education and Training Network Distributed Simulation Federation Object Model) is an extension of the RPR FOM 2.0, allowing other federates using pure RPR FOM to interoperate with applications using the NETN FOM.

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 ONE 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.

Real-time 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 Federation Object Model 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 Real-time 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

#

2D icon
  configure, 341
  edit, 341
2D map, 157
2D View Layer
  overview, 157
  Test-2DViewLayers terrain, 157
3D tile configuration, 147
900_ExtImagery.xml, 61
2525B icon, 341

A

action
  viewing, 475
action table, 499
active region
  moving, 393
  power plant, 393
  weapon fire, 394
actor, 464
  appearance, 466
  configuration file, 465-466
  skeletal hierarchy, 466
adding
  attribute, visual definition, 321
  category, Props Palette, 351
  cockpit model, 347
  DI-Guy animation, 499
  DI-Guy character, 499
  entity bitmap image, 347
  feature layer, 183
  model mapping, 333
  ocean layer, 187-188
  parameter
    model definition, 309
    schema, 300
  particle system, 440
  prop, 193
    from feature layer, 195
  terrain patch to terrain, 180
  terrain server, 154
  visual definition, 319
advanced lighting, 479
aerodrome
  lighting, 359
Aeronautical Charts terrain, 29
aim
  vector, 484
aircraft
  clutter entity, 352
  terrain performance, 127
airport
  procedural, 158-159, 162-167
  procedural performance, 173
  touchdown zone, 165
  Z-fighting, 173
Ala Moana terrain, 28
alias, attribute transformation, 231
alpha, 380
  disable bad value, 41
ambient occlusion
  map, 243
Angle of Attack, 350

animation
DI-Guy
adding, 499
choosing, 498
model, 410
AOA updater, 350
appearance, 464
actor, 466
categories, 466
entity, 408
expressive face, 465
filter, 476
list, 475
multiple for character type, 481
node hierarchy, 409
quality, 476
tutorial, 508
viewing, 475
appearances configuration file, 466
area
dynamic terrain, 202
terrain page-in, 156
areal feature
configuration, 97
simulation terrain, 261
army mode, 481
Arrivals Hall terrain, 28
articulated part, 398, 406
data in model, 406
DI-Guy, 435
keyword, 429
atlas texture, 368-369
attribute
BOYSHP, 141
buoy, 141
building model definition, 141
cockpit updater, 347
COLOUR, 141
COLPAT, 141
DESIGNATOR, 142
destroyedsymbol, 346
earth file
differential processing, 150
feature, 230
imagesymbolmap, 345, 347
light characteristic, 147
LITCHR, 147
MAK_EARTH_FILE, 230
MAK_LAYER, 230
MAK_SOURCE_FILE, 231
marker-scale, 379
modeldefinitionschema, 345
OBJNAM, 142
SIGGRP, 147
signal group, 147
signal period, 147
SIGPER, 147
SIGSEQ, 147
TOPSHP, 146
transformation
alias, 231
transforming, 231
type, 344
visual definition, 319
add, 321
delete, 322
WidgetTheme, 345
augmenting terrain, 69
authentication proxy, 155
Auto Generate, 193
auto refresh Simulation Terrain Debug Tool, 269
Azure Maps API, 113

B

back-end
configuring feature data, 183
debugging simulation terrain from console, 269
Bakersfield topo map terrain, 29
bathymetry, 45
beacon, 147
model definition, 140
seasonal, 146
streaming, 139, 143
topmark, 146
bead
switch, 408
berm, 203
best practice for model design, 380
biome
benefits, 121
dataset, 120
default, 118
ecoregion, 120
habitat, 119
procedural vegetation, 116
realm, 118
bitmap, 31
adding new, 347
displaying, 182
icon, 342
blending LOD transition, 249
BMP, 31
boat, See watercraft
Boundary Generation Tool, 114
running, 115

using, 115
boundary.txt, 115
bounding volume, 480
character, 484
drawing for terrain, 254
BOYSHP attribute, 141
Brooklyn terrain, 29
building (structure)
add to terrain, 273
destroy, 203-204
extruded, 108
instanced, 104
parametric, 105
procedural, 101
catalog, 103
configuration, 101
library, 107
prop type, 197
building terrain, 69
bump map, 243
buoy
attributes, 141
daymark, 146
designator, 142
model definition, 139-140
using texture atlas, 143
seasonal, 146
signal sequence, 147
streaming, 139
topmark, 146
unique, 142
buoyancyLODDistance parameter, 366

C

cache
CDB, 61
version compatibility, 62
clearing, 241
deletion, 62, 238
earth file data, 239
file, configuration, 238
location, 239
missing, 62, 238
osgEarth Cache tool, 239
streaming terrain, 241
terrain, 239
terrain server, location, 239
California terrain, 29
camouflage, 375
category
Props Palette, adding, 351
category, appearance, 466
CDB
cache, 61
deletion, 62, 238
version compatibility, 62
configuration files, 59
data, caching, 61
earth file, 43
elevation sub-dataset, 45
geometry, 65
light point, 51-52
properties, 53
lightmap, 46
material classification, 225
3D models, 225
files, 226
raster material, 225
osgEarth, 42
SpeedTree, 66
terrain database, 42
testing, 42
vegetation, 66
water reflection, 67
CDB raster driver, 43
changing
prop
model definition, 198, 281
position or orientation, 198
type, 197
viewing angle, 477
ChannelKeyword parameter, 350
character
appearance
quality, 476
bounds, 484
DI-Guy, adding, 499
geometry, 488
gesture, viewing, 475
reload, 486
skinned, 464
type, 464
quality, 476
sorting, 476
viewing, 475
character type
quality, 476
sorting, 476
Character Viewer, 474
character_animations_table.mtl, 499
character_type
definitions, 465
character_types_list configuration file, 465
child element definition, 314

CIGI
configuring tidal stream speed and direction, 364
class
diguyApp, 488
diguyCharacter, 488
diguyScenario, 488
DtLightSignalSequenceGenerator, 147
DtLightSignalSequenceGeneratorCreator, 147
classification, material, on demand, 208
clipping plane, 377
clutter object, 352
cockpit display, 332, 347
adding new, 347
element definition, 314
installing, 348
model definition, 349
updater, 347
attribute, 347
collision object, 258
color
pattern, 141
COLOUR attribute, 141
COLPAT attribute, 141
column, convert (mftTool), 37
combat engineering object
as feature, 236
command
diguyEncryptGeometryFile, 462
command-line option
-A, 38
-a, 64
--AppDataPath, 64
-b, 65, 383
-c, 37, 64
--CachePath, 64
--col, 37
-d, 65
--disable301TextureOptimization, 64
--disableIndirectEntities, 294
-e, 37
--EarthFile, 64
--encryptImages, 37
--entityDefsDir, 329
--extendLevelZero, 38
-f, 38, 64
--file, 38
-g, 38
--generateFiles, 38
-h, 39, 65
--help, 39, 65
--ignore_rest, 39, 65
--ignoreNonBaseTex, 383
-l, 38, 64
--level, 38
-lodToProcess, 64
-m, 38, 64
--maxModelsPerGsGrouping, 64
--mergeHierarchyEntityDef, 329
--mergeHierarchyTacticalGraphicsDef, 329
--mergeHierarchyUnitDef, 329
--mergeLeafEntityDef, 329
--mergeLeafTacticalGraphicsDef, 329
--mergeLeafUnitDef, 329
--mergeModelDefs, 312
--minCellSizePerGsGrouping, 65
--missing, 38
--noAppDataEntityDefs, 329
--noAppDataEntityTypeMap, 338
--noAppDataModelDefs, 312
--noAppDataTacticalGraphicsDefs, 329
--noAppDataTacticalGraphicsTypeMap, 338
--noAppDataUnitDefs, 329
--noAppDataUnitTypeMap, 338
--optimiseGsModels, 65
--overwriteGsModels, 65
-p, 39, 65
--processEmats, 38
--processGs, 65
--processGt, 65
--publishMftOnly, 38
-r, 38, 65, 383
--regenerateAll, 38
--removeHighestLod, 383
--row, 38
-s, 39, 65
--scaleLod, 383
--searchDir, 38
--subDir, 39
-t, 39
--textureSize, 39
--threads, 39, 65
-ts, 65
-u, 38
-v, 39, 65
-V, 384
--verbose, 384
--version, 39, 65
--vtFile, 39
-w, 38
-x, 39
comment, @dis, 145
common_actors configuration file, 465
common_characters configuration file, 465
common_links configuration file, 465
component visualization, 246
CompositeElevation, 72

CompositeImage, 72
compressing
  geometry, 462
  model and image files, 381
  texture, 240
configName parameter, 368-369, 371
configuration
  DI-Guy, character and appearance, 499
  file, See configuration file
  tire tracks, persistent, 365
configuration file
  actor, 465-466
  appearances.cfg, 466
  autoloading, 466
  character_animations_table.mtl, 499
  character_types_list.cfg, 465
  clutter entities, 352
  common_actors.cfg, 465
  common_characters.cfg, 465
  common_links.cfg, 465
  default_appearance_effects.cfg, 466
  default_particle_descriptions.cfg, 465
  diguy.cfg, 464
  diguy_actors.cfg, 465
  diguy_character_type_maps.cfg, 465
  diguy_characters.cfg, 465
  exface_shapesets.cfg, 465
  facefx_shapesets.cfg, 465
  feature, 232
  gestures.cfg, 465
  hand_item_appearances.cfg, 465
  hand_item_shapesets.cfg, 465
  including configuration files in, 463
  loaders.cfg, 464
  motex_arcs.cfg, 465
  shader_matrix_index_tables.cfg, 464
  shape_and_joint_data.cfg, 465
  shapeset, 466
  Triton, 364
  weapon_supplemental_data.cfg, 465
convert level (mftTool), 38
copying
  model definition, 308
  schema, 299
  visual definition, 323
crater, 203
custom, DI-Guy directory, 462
cut-in site, 114

D

DAE file, 464
data
  caching terrain, 239
    CDB, 61
  debugging, 258
  elevation, 31
  feature configuration, 230
  S-57, 192
  streamed, processing, 99
data_version, 464
database
  CDB, 42
  terrains in MAK ONE data, 28
dataset
  CDB
    elevation, 43
    extended imagery, 58
    geospecific, 49
    geotypical, 54
    imagery, 45
    material, 47
    road, 56
  virtual and tiled, 37
Datasets.xml, 59
daymark, buoy, 146
daymark.dtxc, 146
daymark_attr_map.csv, 146
daymark_layer_list.csv, 146
DDS texture, 241, 251
  flipping, 376
debris, 204, 440
debug
  terrain, 258
debug draw, 258
Debug Simulation Terrain Tool, 258
debugDrawColors.csv, 258
debugging
  data, 258
  earth file, 241
  NavMesh, 264
  osgEarth terrain, 242
  particle system, 449
  shaders, 246, 480
  simulation terrain, 257-258
  terrain, 242, 250
decal effects, 332
default
  particles, 465
default_appearance_effects configuration file, 466
default_particle_descriptions configuration file, 465
definition
  character_type, 465
  element, 314

model, buoy, 139-140
organization, 314
deform terrain, 203
DegPerKPH parameter, 368
DelayBeforeRot parameter, 368
DEM, 31
depth
ocean, 192
designator
buoy, 142
DESIGNATOR attribute, 142
destroyedsymbol attribute, 346
destructible terrain, 29
detonation
effects, 332
developer debug overlay, 51
DI-Guy
action table, 499
adding new characters, 499
animation
adding, 499
choosing, 498
appearance, 489
effect, 491
articulated part, 435
character and appearance, configuring, 499
data directory, 500
model definition, 307
motion file, 499
swimming, 366
diguy configuration file, 464
DI-Guy Motion Editor, 466
diguy_actors configuration file, 465
diguy_character_type_maps configuration file, 465
diguy_characters configuration file, 465
diguyApp class, 488
diguyCharacter class, 488
diguyEncryptGeometryFile command, 462
diguyScenario class, 488
directory
custom, DI-Guy, 462
search (mftTool), 38
DirectX, 251
DIS
encoding information in OpenFlight format, 398
keyword values, 411
WRM specification, 398
@dis annotation, 369, 371-373, 375
@dis comment, 145
@dis navigation_lights, 145
disableIndirectEntities command-line argument, 294
disabling
indirect rendering, 249, 381
entity, 294
model layer, 61
terrain layer, 177
texture compression, 240
displaying
elevation tile, 266
NavMesh, 264
query area, 265
raster maps, 182
terrain layer, 177
terrain skin, 258
ditch, 203
DLL
cockpit display, 347
installing, 348
driver
simple, 150
DtDiGuyArtPartLinkMapper.h, 435
DTED
elevation, 31
multiple files, 81
DtLightSignalSequenceGenerator class, 147
DtLightSignalSequenceGenerator.h header file, 147
DtLightSignalSequenceGeneratorCreator class, 147
dtxc file, 371
dust, 440
dynamic feature, 236
dynamic ocean
layer, 185
wakes, 361
dynamic terrain
area, 202
berm, 203
crater, 203
damage, removing, 205
deformation, 203
ditch, 203
line, 202
manual changes
resetting, 205
overview, 202
switch type, 203
tank scrape, 203
dynamicTerrainInstaller, 99

E

earth file, 71, 208
adding feature layer, 93
adding mask, 114
areal features, 97

buoys and beacons, 139
caching, 239
CDB, 43
debugging, 241
differential processing of attributes and elements, 150
feature patch, 90-92
feature, loading, 81
including files in, 74
layer, 73
elevation, 77
image, 75
raster, 73
types, 73
loading DTED, 81
syntax, 71
template, 74
earth gravitational model, 78
ECW, 31
editing
2D icon, 341
element definition, 318
keyframe, 442
model definition, 304, 309
model mapping, 334
parameter
model definition, 310
schema, 301
particle system properties, 442
terrain
in terrain editor, 177
property, 178
visual definition, 319
parameter, 320
effect
lighting, 243
mapping, 332
model, 288
preview, 445
special, 440
effectTextureInstaller, 99
EGM84, 78
EGM96, 78
EGM2008, 78
Eigen material technique, 208
elem file, 325
element
earth file, differential processing, 150
mask, 114
model, 93, 150, 379
option, 148
parent, 319
element definition, 314
adding visual definition, 319
changing file location, 324
create, 317, 503
deleting, 324
editing, 318
export, 325
exporting hierarchy and leaves, 326
filter list, 318
in Simulation Object Type Mappings, 335
import, 328
importing, 327
interaction, exporting, 325
jump from object type mapping, 336
jumping to model definition, 323
managing, 289
prevent loading, 329
save, 325
simulation object, 314
elevation
CDB
dataset, 43
sub-dataset, 45
data, 31
EGM, 78
layer
index, 79-80
tile, displaying, 266
emat, processing (mftTool), 38
enabling
terrain layer, 177
texture compression, 240
encrypting geometry, 462
entity
add to Props Palette, 305
aircraft clutter, 352
appearance, 398, 408
clutter, 352
element definition, 314
hier file
import, 329
icon, new image, 347
indirect rendering, 294
leaf file
import, 329
model, 288
surface, 361
clutter, 353-354
swimming, 366
type, enumeration, 333
Entity Image Symbol
visual definition, 341-342, 347
enumeration, 333

environment variable
GDAL_CACHEMAX, 241
OSG_CURL_PROXY, 155
OSG_CURL_PROXYPORT, 155
OSG_MAX_PAGEDLOD, 241
OSGEARTH_AZURE_KEY, 114
OSGEARTH_CACHE_PATH, 239
OSGEARTH_CURL_PROXYAUTH, 155
exface_shapesets configuration file, 465
explosion, 440
exporting
element definition, 325
hierarchy, 325
interaction element definition, 326
model definition, 311
object type mapping, 337
props to shapefile, 198
tactical smoke element definition, 325-326
expressive face, appearances, 465
extended imagery dataset
CDB, 58
extended imagery sub-dataset
CDB, 59
external reference, 380, 399
extracting
prop from terrain patch, 193
water texture, 187
extruded building, 108

F

facefx_shapesets configuration file, 465
FBX
model, 288
texture path, 400
feature
altitude configuration, 90
area, 81, 97
associating with style, 212
attribute, 230
configuration file, 232
data
buoys and beacons, 139
configuring, 230
configuring in front-end and back-end, 183
extracting from GDB, 233
layer, 233
ocean, 192
processing, 99
dynamic, 236
extracting as prop, 193
image, 97
line, 81
point, 81
point model, 98
processing, 99
query, 232, 261
querying, 230
referencing, 89
simulation terrain, 261
source, 81
attributes filter, 88
FilteredFeatures, 85
intersect filter, 87
join filter, 87
multiple filters, 88
MVTFeatures, 83
OGRFeatures, 82
script filter, 86
TFSFeatures, 84
WFSFeatures, 83
XYZFeatures, 85
SwitchStateMapper, 370
UvAtlasMapper, 370
VR-Forces simulation, 90
feature layer, 139
adding, 183-184
adding prop from, 195
earth file, 93
removing, 185
feature patch
creation, 92
earth file, 90
setup, 91
feature rasterization, 212
Feature_Data_Dictionary.xml, 59
feature_geom, 96, 157, 273
featureconfig.txt
mapping features, 183
featureDataBroker, 99
FeatureImage, 72
Feet Above Ground Level, 350
Feet Above Mean Sea Level, 350
FeetAGL updater, 350
FeetMSL updater, 350
FeetPerMinute updater, 350
file
cache
clearing, 241
configuration, 238
compressed, 381
custom directory, DI-Guy, 462
DAE, 464
dtxc, 371
geometry, 368
leaf, description, 315

file location
element definition, 324
object type mapping, 336
Filename parameter, 349
filename_shp_map.txt configuration file, 183
filtering
element definition list, 335
list of element definitions, 318
model definition list, 309
filtering Appearance list, 475
fire, 440
effects, weapon, 332
flag
movement in wind, 367
flame effects, 332
flip DDS textures, 376
FLT file, 31, 288
format
FBX, 399
OpenFlight, 398
fxSystem parameter, 443

G

GDAL, 233
configuring cache, 241
GDAL tools, 81
GDAL_CACHEMAX environment variable, 241
GDALElevation, 72
GDALImage, 72
GDB terrain, 31
extracting feature data, 233
shapefile, 183
generative vegetation, 116-118, 125
geocentric terrain, 154
geometry, 380
character, 488
compressing, 462
customizing, 466
encrypting, 462
file, 368
processor, 462
Geometry Grid Dataset, 36
geometry_set_priority_list, 464
geospecific
CDB
airport light point sub-dataset, 51
environmental light point sub-dataset, 52
sub-dataset, 50
dataset, 49
GeoTIFF
elevation, 31
imagery, 31
material classification, 213
ocean masking, 257
performance, 252
geotypical dataset
CDB, 54
geotypical sub-dataset
CDB, 56
gesture, 465
character, viewing, 475
gestures configuration file, 465
GGDS, 36
GGDS, MetaFlight file (mftTool), 38
GIF, 31
GL Studio, 347
Gouraud, unsupported, 289
group
map, 157
guidelines
lightpoints, 358
material system file, 387
model creation, 380
model mapping, 401

H

hand item
appearances, configuration file, 465
create, 467
first-person rig, 471
third-person character, 469
hand items
shapesets, configuration file, 465
header file
DtDiGuyArtPartLinkMapper.h, 435
DtLightSignalSequenceGenerator.h, 147
Headlight updater, 351
head-up display, See cockpit display
height
elevation tile, 266
ocean, 187-188
terrain data, 31
height map
ocean
regenerating, 191
resolution, 189-190
texture size, 189, 191
help command-line option, 39, 65
hiding
terrain layer, 177
hier file, import, 328
hierarchy
element definition, 314
export, 326

exporting, 325
file, importing, 328
importing, 327
HUD, See cockpit display

I

icon
2D, 342
bitmap, 342
edit, 341
2525B, 341
bitmap, new, 347
configuring 2D, 341
image, 342
new, 347
image
adding, 347
compressing, 381
displaying, 182
icon, 342
supported formats, terrain, 31
imagery
CDB
dataset, 45
lightmap sub-dataset, 46
sub-dataset, 46
streaming sensorized, 213
imagesymbolmap attribute, 345, 347
ImGui overlay
disable indirect rendering, 249
imod file, 391
importing
element definition, 327-328
hierarchy file, 327-328
leaf file, 327-328
model definition, 312
props from shapefiles, 199
SpeedTrees from shapefile, 199
including XML files in earth files, 74
indirect rendering, 56, 248, 250
disabling, 249, 381
entity, 294
LOD distance scale, 248
LOD transition blending, 249
infrared
light source, 361
inheritance
element definition, 314
instancing, 399
intensity modulation file, 391
interaction
element definition, exporting, 325-326
intersection
disable for model layer, 61
displaying, 267
map layer, 157
simulation object, 61
intersector line
terrain, 250
inverse kinematics, 484
isamphibious parameter, 366
IVE, 155

J

joint, 464
JPEG, 31
JPG, 31

K

keyframe, 452, 454
editing in particle system, 442
keyword
articulated part, 405
DIS
air domain, 414
animations, 437
articulation, 429
attachment, 436
cultural feature, 425
environment, 424
expendable, 428
land domain, 412
lifeform, 421
munition, 423
sensor, 425
space domain, 420
subsurface domain, 419
supply, 427
surface domain, 417
state, 406
support, 405
switch, 405
KilometersPerHour updater, 350
kinematics
inverse, 484
KnotsPerHour updater, 350

L

lamppost
prop type, 197

layer
  earth file, 73
  earth file, debugging, 241
  editing in Terrain Editor, 178
  elevation, 77
    index, 79-80
  enabling or disabling, 177
  feature, 139
    adding, 183-184
    adding prop from, 195
    adding to earth file, 93
    removing, 185
  feature data, 233
  image, 75
  map, 157
  ocean, 185
    adding, 188
  types, 73
    raster, 73
leaf file
  description, 315
  export, 326
  import, 327-328
LettersNumbers.ctxc, 143
level, convert (mftTool), 38
level-of-detail, 377, See also LOD
lifeform
  DI-Guy appearance and character, 499
light
  navigation, 144
  sensor mode specific sources, 361
light characteristic attribute, 147
light point
  add, 358
  airfield, 169
  CDB, 51-52
    properties, 53
  configure, 355
    emissive texture, 357
    group, 357
    source, 357
    type, 357
Light_DefaultValues_3_0.xml, 60
Light_DefaultValues_3_2.xml, 60
light_layer_list.csv, 145
lighting
  advanced, 479
  aerodrome, 169, 359
  airfield, 167, 172
  airport, 358
  airport runways, 167
  applying, 479
  best practices, 358
  effects, 243
  intensity, 170, 359
  model definitions, 169
  runway, 358
  vehicle, 358, 360
lightmap
  CDB, 46
lightPointInstaller, 99
Lights_xxx.xml, 61
line, dynamic terrain, 202
linear deformation, 203
linear feature
  simulation terrain, 261
link, 464
list
  props, 197
LITCHR attribute, 147
loaders configuration file, 464
location
  cache, 239
LOD, 377, 380
  center, 256
  configuring, 241
  level, viewing different, 478
  ocean elevation, 189
  range
    maximum, 256
    minimum, 256
  removing highest (medfTool), 383
  scale for indirect rendering, 248
  scene graph node, 255
  specifying maximum for earth file, 250
  switch scale (medfTool), 383
  transition blending, 249
Lua
  script
    running, 488
LWO file, 31

M

MAK Earth terrain, 29
MAK Encrypted Data Format, 36
MAK_EARTH_FILE attribute, 230
MAK_LAYER attribute, 230
MAK_SOURCE_FILE attribute, 231
map
  texture, 243
    specular, 112
map layer, 157
mapgroup, 157
mapgroup_isect, 157

mapping
  model
    to object type, 333
  object type, 333
  shapefiles to features, 183
  soil types, 252
marker-scale attribute, 379
marking, 375
  model, 371-372
mask
  earth file, 114
  element, 114
  model, 375
  ocean
    model, 375
    terrain, 257
Massachusetts S-57 terrain, 184
material classification
  CDB terrain, 225
  debug, 392
  file, 393
  files, 387
  models, 391
  on demand, 208
  process, 386
  software tools, 386
material dataset
  CDB, 47, 225
material encoded texture, 208
material system
  assignment descriptor, 220
  boundary condition terhaml equations, 221
  common, 219
  overview, 214
  thermal phenomenon, 215
MBTilesElevation, 72
MCOD, 208
MEDF, 36
  creating, 381
MEDF file, 31
medfTool, 40
MEIF
  creating, 381
merging model definition, 312
metafile
  marking, 373
  texture skinning, 368-369
metaFileName parameter, 369, 371
MetaFlight
  building efficient terrains, 35
  file, publish (mftTool), 38
  loading, 35
  processing, 36
mftTool, 37, 39
  threads, 39
MilesPerHour updater, 350
Military Symbol Icon
  visual definition, 341
model
  @dis annotation, 372
  3D, 288
  animation, 410
  annotation, 372
  articulation, 406, 429
    data, 406
  buoy definition, 140
  composable, 290
  compressing, 381
  coordinate system, 402
  creating, 380
  definition, managing, 289
  DI-Guy, choosing, 498
  editing model mapping, 334
  element, 93, 99, 379
  FBX, 288
  file caching, 238
  flipping DDS textures, 376
  intersections, 267
  layer
    disabling, 61
  mapping
    add, 333
    delete, 337
    editing, 334
  markings, 371-372
  masking, 375
  object, overview, 286
  OpenFlight, 288
  prop, 351
  schema, 304
  set, 287, 318
  texture, 368
  texture atlas, 369
model definition, 393
  apply updated, 309
  buoy, 139-140
    building from attributes, 141
    texture atlas, 143
  copy, 308
  create, 304, 307, 501
    cockpit display, 349
  deleting, 307
  DI-Guy character, 307
  edit, 309
  export, 311
  file, 306

filtering list of, 309
import from command line, 312
importing, 312
jumping to from element definition, 323
merging, 312
parameter
adding, 309
deleting, 311
editing, 310
particle system, 443
prevent loading, 312
prop
changing, 198, 281
save, 311
schema, 296
SpeedTree, 307
texture atlasing, 369
model element, 150
model mapping
jump to element definition, 336
Model Specification for DIS Interoperability, 398
Model_PagingDistance.xml, 60
modeldefinitionschema
attribute, 345
motex_arcs configuration file, 465
motion, 466
file, 499
textures, 465
movement
unit
restriction, 234
moving active region, 393
movingActiveRegions parameter, 393
MS file, 394
msFilename parameter, 393
MultiGen-Paradigm
OpenFlight format, 398
multiple parameter values, 320
myExtractionOperations, 193

N

named query, 230
navigation
light, 144
streaming, 144
profile, 202
navigation data
regenerating, 202
navigational aid light, 147
navigationProfiles.mtl, 202
NavMesh
debugging, 264
network
publishing
DI-Guy character, 493
networking, 493, 495
night vision
light source, 361
node
DIS notes, 400
field
comment, 399
name, 399
model, remove (medfTool), 383
switch, 367, 370
normal map, 243
notification level
Character Viewer, 493
NTC (online) terrain, 30

O

OBJ file, 31
object
add to Props Palette, 305
model, 288
overview, 286
object type mapping, 333
change file location, 336
export, 337
import, 338
jump to element definition, 336
save, 337
OBJNAM attribute, 142
observer
ocean LOD, 189
ocean
depth, 192
height, 187-188
height map
regenerating, 191
resolution, 189
texture size, 189, 191
layer, 185
adding, 187-188
masking
Triton.config, 257
Ocean Height Map Regeneration Threshold, 191
ocean height map resolution, 190
Ocean Layer, 189
Ocean LOD elevation, 189
offset
UV, 371
OffsetX parameter, 350
OffsetY parameter, 350

OffsetZ parameter, 350
OGR, 233
ogr2ogr, 233
OGRFeatures, 94
OpenFlight, 288
  converting texture data, 252-253
  disabling indirect rendering for, 381
  file, 31
    format, 398
  format, 398
  model, 288
  terrain patch, 278
OpenGL, 251
option element, 148
orientation
  prop, 198
OSG_CURL_PROXY environment variable, 155
OSG_CURL_PROXYPORT environment variable, 155
OSG_MAX_PAGEDLOD environment variable, 241
osgEarth, 71, 154, 208, 241
  adding terrain servers, 154
  CDB, 42
  debug mode, 242
  indirect rendering, 250
osgEarth Boundary Generation Tool, 114-115
osgEarth Cache Tool, 239
OSGEARTH_AZURE_KEY environment variable, 114
OSGEARTH_CACHE_PATH
  environment variable, 239
OSGEARTH_CURL_PROXYAUTH environment variable, 155
overlay
  developer debug, 51

P

paged terrain
  flip DDS textures, 251
  nodes, 268
  paging manually, 156
  preprocessing, 155
paper map, 157
parameter
  adding to model definition, 309
  buoyancyLODDistance, 366
  ChannelKeyword, 350
  configName, 368-369, 371
  DegPerKPH, 368
  DelayBeforeRot, 368
  deleting from model definition, 311
  Filename, 349
  fxSystem, 443
  isamphibious, 366
  metaFileName, 369, 371
  model definition
    edit, 309
    editing, 310
  movingActiveRegions, 393
  msFilename, 393
  OffsetX, 350
  OffsetY, 350
  OffsetZ, 350
  particle system, 451
  Perspective, 349
  PerspectiveHeight, 349
  PerspectiveWidth, 349
  possiblegroundclampingmode, 366
  powerPlantActiveRegions, 393
  RotDir, 368
  RotOffsetDeg, 368
  RSOName, 349
  schema
    add, 300
    delete, 301
    edit, 301
  visual definition, editing, 320
  waterlineOffset, 366
  weaponFireActiveRegions, 394
  WindSpeed, 367
  WindState, 367
parent
  element, 319
  element definition, 314
Particle Collision Object visualizer, 457
particle system, 440
  adding, 440
  configuring for SensorFX, 394
  debugging visuals, 449
  keyframe, editing, 442
  model definition, 443
  parameters, 451
  pause, 445-446
  preview, 445
  preview parameters, 456
  properties, 451
    editing, 442
  reverting changes, 443
  rotor wash, 457
  save, 443
  textures, 447
    importing, 453
  wireframe mode, 448
particle, default, 465

patch
terrain data, 31
terrain properties, 178
terrain, adding, 278
pattern
color, 141
PBR Material Tester, 491
performance, 189
caching terrain, 61, 238
CDB terrain caching, 61
compressing texture, 240
elevation layers, 77
file caching, 238
indirect rendering, 248
intersections, 61
level of detail, 77
model
best practices, 380
intersections, 61
terrain, 61, 85, 127, 238
Perspective parameter, 349
PerspectiveHeight parameter, 349
PerspectiveWidth parameter, 349
PIP file, 394
Pitch updater, 350
pixel, 380
playback control, 486
PNG, 31
point
feature, simulation terrain, 261
polygon, 380
underwater, 192
port
proxy, 155
position
prop, 198
possiblegroundclampingmode parameter, 366
power plant active region, 393
powerPlantActiveRegions parameter, 393
preprocessing
paged terrain, 155
procedural airport, 158, 166
airports, 162
data, 162
data files, 159
lights, 163
approach, 163
beacon, 163
displaced threshold, 164
edge, 164
end, 165
PAPI, 165
runway centerline, 164
runway end identifier, 164
runway guard, 165
taxiway, 165
threshold, 165
touchdown zone, 165
VASI, 165
NAVAID, 167
pavement, 163
runways, 162
taxiway signage, 163
terminal, 166
visual markings, 163
VRFSIM data, 167
windsocks, 166
procedural building, 101
procedural ground cover, 118
procedural imagery, 117
asset, 130
Copernicus class mapping, 130
procedural terrain
biome, 118, 120-121, 129
ecoregion, 120
example, 121
land cover, 121
major habitat type, 119
multiple, 123
realm, 118
land cover, 132
legacy, 133
trees, 135, 137
vegetation, in VR-Forces, 137
lifemap, 121
real world emulation, 118
trees, 127
procedural vegetation, 116
asset, 116
trait, 121
assets, 121
biome, 117
CDB, 66
legacy, 137
life map, 117, 121
soil type, 125
trait, 121
weighted effect, 123
mask, 124
texture override, 124
prop
adding
from feature layer, 195
to Props Palette, 305, 351
to terrain, 193
creating, 351

edit existing, 197
exporting to shapefile, 198
extracting, 193
importing from shapefile, 199
model definition
changing, 198, 281
saving, 351
selecting, 197
type, 197
viewing list of, 197
propeller wash, 364
property
editing
Terrain Editor in, 178
particle system, 451
editing, 442
terrain
deleting, 178
Props Palette
adding objects to, 305, 351
category, adding, 351
proxy
connecting through, 155
port, 155

Q

quality
character, 476
threshold, 476
query
area, displaying, 265
feature, 232, 261
named, 230

R

raster driver
CDB, 43
raster map
displaying, 182
including, 245
rasterization feature, 212
reflection
map, 243
on water, 112
refreshing
simulation terrain debug display, 269
regenerating
navigation data, 202
ocean height map, 191
registrant_feature_list.csv, 99
rendering
indirect, 248
required_sdk_version, 464
resolution
ocean height map, 190
tree, 377
reusable software object, 347
reverse Z buffer, 173, 186
RGB, 31
road dataset
CDB, 56
Roll updater, 351
RotDir parameter, 368
RotOffsetDeg parameter, 368
rotor wash particle system, 457
roughness
soil, 258
row, convert (mftTool), 38
RSO, 347
RSOName parameter, 349
rubble, 204

S

S-57
buoys and beacons, 139
converting to shapefile, 233
data, 192
signal sequence, 147
s57_attr_map.csv, 141, 144
s57_defaults.csv, 144
s57_layer_list.csv, 139, 143
s57_modeldef_by_name_map.csv, 142
Sadr City terrain, 30
scale
LOD distance for indirect rendering, 248
SpeedTree, random, 379
scene graph node
LOD, 255
node bounds, 255
scene graph nodes
bounding volume, 254
schema
copy, 299
creating, 297
delete, 299
export, 298
model definition, 296
parameter
add, 300
delete, 301
edit, 301
save, 298

saving, 296
visual definition, 296
scrape, tank, 203
screen capture, 26
script
Lua
running, 488
seasonal buoys and beacons, 146
sensor
light source, 361
model apperance, 386
terrain apperance, 208
server, terrain
caching, 239
cutting in site, 114
shader, 243
component visualization, 246
debugging, 246, 480
FBX, 401
PBR, 491
reload, 492
reloading, 247
shader_matrix_index_table, 466
shader_matrix_index_tables configuration file, 464
shadow
viewing, 479
shape, 464-465
shape terrain data, 31
shape_and_joint_data configuration file, 465
shapefile
buoys and beacons, 139
converting S-57 to, 233
exporting props to, 198
importing props from, 199
importing SpeedTrees from, 199
mapping to feature, 183
shapeset, 464
configuration file, 466
ship, See also watercraft
clutter, 353-354
wake, 361
SHP file, 31
SIGGRP attribute, 147
signal
sequence, 147
signal group attribute, 147
signal period attribute, 147
SignificantSize_SwitchDistance.xml, 60
SIGPER attribute, 147
SIGSEQ attribute, 147
simple driver, 150
simulation object
element definition, 314
icon, 347
Simulation Object Type Mappings
filter element definition list, 335
simulation terrain
debugging
features, 261
from console, 269
Simulation Terrain Debug Tool
auto refresh, 269
site
cutting in, 114
size
SpeedTree
randomizing, 379
skeletal hierarchy, 466
skeleton, 480
skinned character, 464
smoke, 440
soil
roughness, 258
type, 252-253, 258
sonobuoy, daymark, 146
sound
customizing, 467
special effect, 440
specular
map, 243
texture, 112
SpeedTree, 99
CDB, 66
configuring, 377
displaying in planar reflection, 380
exporting to shapefile, 198
extraction operation, 193
importing from shapefile, 199
model definition, 307
MTF file, 193
random scale, 379
speedTreeManager, 99
standing_hip_height, 466
streamed data
processing, 99
streaming
beacon, 143
buoys and beacons, 139
navigation lights, 144
sensorized imagery, 213
terrain
configuring cache, 241
style, associating with feature, 212
sub-dataset
CDB
elevation, 45

extended imagery, 59
geospecific, 50
geospecific airport light point, 51
geospecific environmental light point, 52
geotypical, 56
imagery, 46
imagery lightmap, 46
surface characteristic map, 252
surface entity, See also watercraft
clutter, 353-354
wake, 361
surfChar.map, 252
swimming entity
configuring, 366
switch bead, 408
switch node, 367, 370
aerodrome, 359
vehicle, 360
switch type
dynamic terrain, 203
switch, keyword value, 411
SwitchStateMapper
feature, 370

T

tactical graphic, 332
element definition, 314
hier file
import, 329
leaf file
import, 329
tactical smoke element definition
export, 325-326
tank scrape, 203
template
earth file, 74
terrain
2D maps, 157
2D View Layers example, 157
asset, 133
below sea level, 257
bounding volume
drawing, 254
composing, 29
composition, 180
construction, 180
create, 180
custom, 34
damage, 203
database
CDB, 42
databases provided, 28
debug mode, 242
debugging, 250, 258, 261, 264-269
deformation, 203
destructible, 29
displaying raster image on, 182
dynamic, 202
earth file
debugging, 241
editing in terrain editor, 177
export, 138
feature data
configuring, 230
formats, 31
indirect rendering, 248
disabling, 249
intersector line, 250
manual changes, 204
reset, 205
Massachusetts S-57, 184
MetaFlight
designing, 35
loading, 35
processing, 36
open, 34
osgEarth
maximum paged LOD, 250
paged
configuring LODs, 241
flip DDS textures, 251
preprocessing, 155
page-in area, 156
paging
manually, 156
MetaFlight, 35
patch, 31
adding to terrain, 180, 278
extracting props from, 193
procedural, 127
legacy, 135, 137
procedural, legacy, 133
prop
adding, 193
list, 197
querying, 230
saving, 185
sea level, below, 257
server, See terrain server
shape, 31
simulation
debugging, 257-258
debugging from console, 269
streaming
cache, 241

supported formats, 31
underwater, 192
terrain cache
configure, 238
delete, 241
generate, 239
terrain database
Aeronautical Charts, 29
Ala Moana, 28
Arrivals Hall, 28
Bakersfield topo map, 29
Brooklyn, 29
Destructible Terrain, 29
MAK Earth, 29
NTC (online), 30
provided, 28
Sadr City, 30
Truman carrier, 30
VR-Village, 30
Terrain Editor
caching osgEarth data, 239
deleting property, 178
editing terrain, 177
introduction, 176
layer
editing, 178
editing property, 178
opening, 176
types of layers, 177
terrain normal, 258
terrain patch
add, 180
adding to terrain, 278
properties, 178
terrain server, 154
adding, 154
caching data, 239
connecting through proxy, 155
cutting in site, 114
mask, 114
terrain skin
displaying, 258
terrain tile
displaying, 242
TerraPage file, 31, 155
texture, 380
atlas, See texture atlas
compression
enabling and disabling, 240
customizing, 466
data conversion, 252-253
DDS, 251
FBX, 400
flip paged terrain, 251
map, 243
specifying, 453
specular, 112
mapping file, 375
mapping to model, 368
material encoded, 208
missing placeholder, 38
model, 368
motion, 465
non-base, ignore (medfTool), 383
particles, 447
importing, 453
placeholder (mftTool), 38
size
maximum (mftTool), 39
ocean height map, 189, 191
skinning
metafile, 369
model, 368
substitution (mftTool), 39
swap, tutorial, 508
tiled and virtual, 37
water, 185
texture atlas, 368-369
buoy and beacon, 143
TFSFeatures, 94
threads (mftTool), 39
tidal stream
speed and direction
configuring in CIGI, 364
wake, 364
TIF, See GeoTIFF
tile, 3D, 147
tiled texture, 37
format (mftTool), 37
MEDF (mftTool), 38
regenerate (mftTool), 38
tire track, 365
TMSElevation, 72
TMSImage, 72
topmark
buoy and beacon, 146
topmark_attr_map.csv, 146
TOPSHP attribute, 146
tracks, vehicle, 365
trailing effects, 332
transformation
derived attribute, 231
new attribute, 231
transforms, multiple, 232
transition blending
LODs, 249

transparency, 380
tree
  legacy
    procedural, in VR-Forces, 135, 137
  procedural, in VR-Forces, 127
  prop type, 197
    SpeedTree, 193
  SpeedTree, 377
    prop, 193
  windspeed, 377
Triton.config
  ocean masking, 257
  propeller wash, 364
TrueType font
  2D icon, 342
Truman carrier terrain, 30
tutorial
  appearance, 508
  earth file, 272-273, 276
  prop extraction, 278-279, 281
  terrain, 271
  texture swap, 508
TXP file, 31
type
  character, 464
    quality, 476
    sorting, 476
  prop, 197
type attribute, 344

U

unique buoy, 142
unit
  hier file
    import, 329
  leaf file
    import, 329
  movement restriction, 234
updater, 347
  AOA, 350
  attribute, 347
  FeetAGL, 350
  FeetMSL, 350
  FeetPerMinute, 350
  Headlight, 351
  KilometersPerHour, 350
  KnotsPerHour, 350
  MilesPerHour, 350
  Pitch, 350
  Roll, 351
  Value, 351
  Yaw, 351
Use Extracted Geometry, 193
utility
  cdbTool, 61-63
  medfTool, 381
  mftTool, 37
  simTreesTool, 127
UV offset, 371
UvAtlasMapper
  feature, 370

V

Value updater, 351
vegetation
  legacy
    procedural, in VR-Forces, 137
  procedural, 66, 116
vehicle
  lighting, 360
  persistent tracks, 365
  trailing effects, 332
verbose command-line option, 384
version information, 39, 65
viewing angle, 477
virtual texture, 37
Virtual Texture Datasets, 36
visual definition, 296, 314, 318
  adding, 319
  attribute, 319
    add, 321
    delete, 322
  clearing from memory, 323
  copying and pasting, 323
  edit, 319
  editing, 320
  Entity Image Symbol, 341-342, 347
  Military Symbol Icon, 341
  schema, 296
visual quality, 189
visualizer, 314
  Particle Collision Object, 457
volume
  bounding, character, 484
vrfsim:, 150
  enabled, 150
VR-Village terrain, 30
VTDS, 36
VTDS file (mftTool), 39

W

wake, See also propeller wash
  ship, 361
  tidal stream, 364
water
  CDB, 67
  masking
    model, 375
    Triton.config, 257
  reflections on, 112
  texture, 185
    extracting, 187
watercraft
  clutter, 353-354
  wake, 361
  water mask, 375
waterlineOffset parameter, 366
weapon fire active region, 394
weapon supplemental data configuration file, 465
weaponFireActiveRegions parameter, 394
WFSFeatures, 94
WidgetTheme
  attribute, 345
wind
  moving flags and windsocks, 367
  SpeedTree trees, 377
WindDirPart, 367
WindRotPart, 368
windsock
  movement in wind, 367
WindSpeed parameter, 367
WindState parameter, 367
WindSwitchPart, 367
wireframe, 480
wireframe mode
  enabling for particles, 448
WMSImage, 72
WRM Entity Model Specification, 398, 409

X

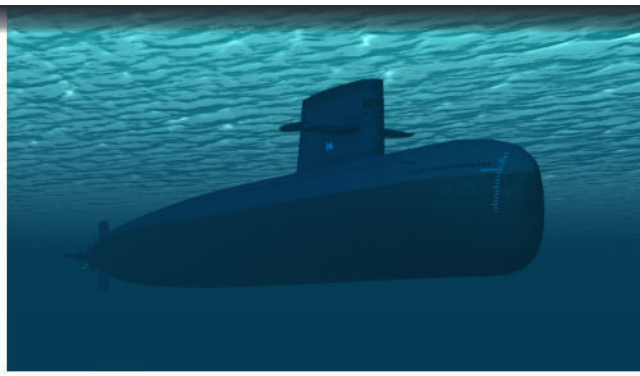
XML, 71
  include files, 74
XXX_SpeedTrees.xml, 61
XYZFeatures, 94
XYZImage, 72

Y

Yaw updater, 351

Z

Z-fighting, 173, 186



MAK TECHNOLOGIES 10 FAWCETT STREET, SUITE 204, CAMBRIDGE, MA 02138 USA | +1 617.876.8085 | www.mak.com