VR-Vantage Capabilities

This document summarizes the features of the VR-Vantage applications and toolkit. VR-Vantage is VT MÄK's line of visualization products. It is designed to meet your needs for visualizing the simulated world. The VR-Vantage product line consists of the VR-Vantage Toolkit and the following applications that are built with it:

- VR-Vantage Stealth. VR-Vantage Stealth is your 3D information station. It is optimized for customers who need a 3D environment for situational awareness, simulation debugging, or after action review. The Stealth provides the most data about your networked virtual world, and presents it in a clear and accessible way.
- VR-Vantage IG. VR-Vantage IG is a configurable desktop image generator for customers who need an out-the-window (OTW) view or a picture from a remote camera. Vantage IG is designed for easy integration into desktop trainers and war gaming simulations.
- VR-Vantage PVD. VR-Vantage PVD provides a 2D plan view display (map) of terrain and simulated objects. Plan view displays provide the "big picture" of simulated environments. They sacrifice the situational awareness of the 3D environment, but let you see the entire simulated world at a glance. Entities are represented using MIL-STD 2525B icons that are color coded to represent their force.
- VR-Vantage XR. VR-Vantage XR (for eXaggerated Reality) combines the 3D capabilities of VR-Vantage Stealth with the 2D plan view display of VR-Vantage PVD. Then it adds 3D colorized models to provide both an immersive 3D experience and a common operational picture of the simulated world.
- VR-Vantage FreeView. VR-Vantage FreeView is a free terrain and 3D model viewer that introduces customers to some of the features of VR-Vantage while providing some useful functionality. VR-Vantage FreeView is not documented in this manual. For more information, please see the VR-Vantage FreeView First Experience Guide and VR-Vantage FreeView online help.

Document ID: VRV-1.3-9-110420

The VR-Vantage Toolkit is an API that comes with all the software you need to completely rebuild visual applications like VR-Vantage Stealth and also allows you to customize and extend these applications to fit your unique requirements. The toolkit also gives you the power to embed any of VR-Vantage's capabilities directly into your simulation applications. The VR-Vantage Toolkit is based on OpenSceneGraph[1], so you can leverage value-added plug-ins built by the OSG community and MÄK partners.

VR-Vantage Capabilities Matrix

Table 1 lists VR-Vantage application capabilities and the VR-Vantage applications that support them. Many of the capabilities are described later in this document.

Table 1: VR-Vantage capabilities (Sheet 1 of 3)

Feature	IG	Stealth	PVD	XR	FreeView
2D entity labels			●	●	
2D icons			●	●	
3D colorized model set				●	
3D entity labels	●	●		●	
3D model set	●	●		●	
Aggregate visualization	●	●	●	●	
Attach modes	●	●	●	●	
Compass display	●	●	●	●	●
Contour lines	●	●	●	●	●
Contrails	●	●		●	
DI-Guy characters	●	●		●	
Ephemeris model	●	●		●	●
Explosions	●	●		●	
File caching	●	●	●	●	●
Fire and detonation lines	●	●	●	●	
Flames	●	●		●	
Footprints	●	●		●	
GL Studio cockpits	●	●		●	
Grid lines			●	●	●

1. "OpenSceneGraph is an Open Source, cross-platform graphics toolkit for the development of high-performance graphics applications such as flight simulators, games, virtual reality and scientific visualization. It is based on the concept of a SceneGraph, providing an object-oriented framework on top of OpenGL." (http://www.openscenegraph.org/projects/osg/wiki/About/Introduction)

Table 1: VR-Vantage capabilities (Sheet 2 of 3)

Feature	IG	Stealth	PVD	XR	FreeView
Ground clamping	●	●		●	
GUI-based model mapping	●	●	●	●	
Height-above-terrain lines	●	●		●	
Icon scaling			●	●	
Inset views	●	●		●	
Instancing	●	●	●	●	●
Interest management	●	●	●	●	
Intervisibility		●	●	●	
Keyboard navigation	●	●	●	●	●
Missile trails	●	●		●	
Model scaling				●	
Model viewer					●
Mouse navigation	●	●	●	●	●
Multiple observers	●	●	●	●	●
Muzzle flash	●	●		●	
Overlays	●	●	●	●	
Persistent settings	●	●	●	●	
Remote graphics	●	●	●	●	
Saved display engine configurations	●	●	●	●	
Saved views	●	●	●	●	●
Scenes	●	●	●	●	●
Sensor volumes	●	●	●	●	
Shadows	●	●		●	
SIlverlining environment	●	●		●	
Smoke plumes	●	●		●	
SpeedTrees	●	●		●	
Statistics viewer	●	●	●	●	●
Stereoscopic display	●	●		●	
Streaming terrain	●	●	●	●	●
Tactical graphics display	●	●	●	●	
Terrain agility	●	●	●	●	●
Terrain coloring by elevation	●	●		●	●
Track histories	●	●	●	●	

Table 1: VR-Vantage capabilities (Sheet 3 of 3)

Feature	IG	Stealth	PVD	XR	FreeView
Trajectory smoothing	●	●	●	●	
Treadmarks	●	●		●	
View control support	●	●	●	●	
Visualization of feature data			●	●	●
Wakes	●	●		●	
Wireframe display	●	●	●	●	●

System Requirements

Table 2 lists supported operating systems and other system requirements.

Table 2: System requirements

	Windows	Linux
OS version	Windows XP Professional Windows Vista Windows 7	Red Hat Enterprise Workstation Linux 5
Compiler	Microsoft Visual C++ v8.0 SP 1 Visual C++ v9.0 SP 1	gcc 4.1
Processor	X86 (32 bit) X86-64 (64 bit)	X86 (32 bit)
RAM	2 GB+	2 GB+
Hard disk space	16 GB+	16 GB+
Graphics	NVIDIA® GeForce 8 series or higher NVIDIA® Quadro FX: "high end" or "ultra-high-end	NVIDIA® GeForce 8 series or higher NVIDIA® Quadro FX: "high end" or "ultra-high-end
Shader support	NVIDIA CG 2.1	NVIDIA CG 2.1

Export Classification

VR-Vantage and all of its modules are classified as EAR 99 by the US Commerce Department.

VR-Vantage Product Licensing

Each VR-Vantage product is sold separately; however, the VR-Vantage Stealth and VR-Vantage XR products include a license to run VR-Vantage IG. Please contact your MÄK salesperson for details. MÄK license management uses FLEXlm. Licenses can be floating, node-locked, or dongle-based.

VR-Vantage Application Core Capabilities

This section describes, at a high level, major functional capabilities of VR-Vantage applications. These core capabilities are more than just "features" (which are described later). They are integral to what a VR-Vantage application is and can do. The components are:

- Terrain Agility and composability
- The display engine
- The observer
- Object modeling.

Terrain Agility and Composability

VR-Vantage allows you to build your terrain at runtime using a variety of database and vector formats. VR-Vantage supports the following broad types of terrain:

- Terrain models. Static terrains, such as OpenFlight, built using terrain construction applications.
- Paging terrain. Large area terrains that page in multiple terrain pages, such as MetaFlight.
- Direct from source. Terrains composed by combining various types of terrain elevation, imagery, and features data.

  VR-Vantage can load OpenFlight DTED, CTDB, MetaFlight, and GDB (MÄK terrain format) databases. It can load feature data from shapefiles and GDB files. You can also superimpose geospatial images such as GeoTiffs or other raster image files on the terrain. You can save these composed terrains in the VR-Vantage MTF file format.

- Open streaming terrain. Terrains that stream data from various public and private data sources, such as VR-TheWorld.

VR-Vantage File Format Support

VR-Vantage supports a large variety of 2D image and 3D model file formats. OSG supplies a QuickTime plugin for loading movie files, and a plugin for loading font files using the FreeType library.

VR-Vantage supports the following 3D model formats:

- MÄK terrain database (*.gdb*)
- MÄK encrypted data format (*.medf*)
- MÄK Stealth format (*.mtl*)
- MÄK terrain project or terrain descriptor file (*.mtp*, *.mtd*)
- OpenFlight (*.flt*)
- MetaFlight (*.mft*)
- DTED (*.dt0*, *.dt1*, *.dt2*, *.dted*)
- CTDB 4, 7, 7L (*.c4b*, *.c7b*, *.c7l*)
- ESRI Shapefile (*.shp*)
- Terrex Terrapage (*.txp*)
- 3D Studio (*.3ds*)
- Alias Wavefront (*.obj*)
- Carbon Graphics' Geo (*.geo*)
- OSG (*.osg* and *.ive*)
- COLLADA (*.dae*)
- Quake (*.md2*)
- NewTek LightWave (*.lwo* and *.lws*).

The *.osg* format is an ASCII text representation of a scene graph that you can edit in a text editor. The *.ive* format is a binary format, which is optimized for fast loading.

VR-Vantage has several file formats of its own. MEDF is the MAK Encrypted Data Format. MEIF is the MAK Encrypted Image Format. MTF is the MAK Terrain Format. And MSF is the MAK Scene Format.

You can import images in the following formats:

- GEO Tiff (*.tif*, *.tiff*)
- JPEG (*.jpg*, *.jpeg*)
- PNG
- RGB
- Windows bitmap (*.bmp*)
- GIF.

Props

Props are terrain elements, such as buildings, light poles, and vegetation, that you can manipulate through the VR-Vantage GUI. VR-Vantage can import shape (or other feature data file formats) and create geometry (props) for point features in the source data.

Streaming and Paging Terrain

VR-Vantage handles the paging of tiles in and out through OSG's Database Pager. For MetaFlight, VR-Vantage has its own pager. Streaming terrain is handled by osgEarth's database pager.

VR-Vantage applications can stream elevation and imagery from terrain servers such as VR-TheWorld or other WMS-C (Web Mapping Service-Cached, from Open Geospatial Consortium) and TMS (OSGeo's Tile Map Service) servers.

The VR-Vantage Display Engine

VR-Vantage applications have a component called a display engine. The display engine renders visual data. A display engine can also control the display on remote display engines. VR-Vantage includes an application called the VR-Vantage Display Engine. A VR-Vantage Display Engine has a very limited GUI. Its primary purpose is to provide windows that can be controlled by a VR-Vantage application's (master) display engine.

The master display engine can connect to multiple VR-Vantage Display Engines (typically running on other computers) and control what is displayed in their windows.

VR-Vantage drives all of the display engines it is connected to, as illustrated.

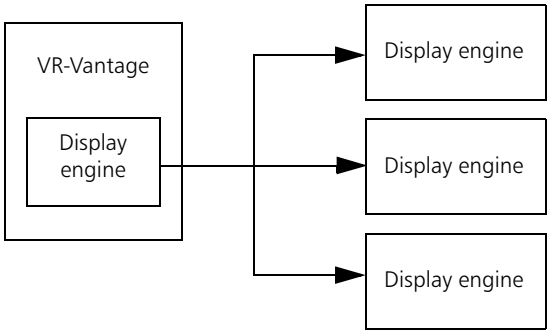

Figure 1. VR-Vantage and display engines

Windows and Channels

A display engine (including the master display engine) can display multiple windows, each of which can have zero or more channels. The windows are all configured in VR-Vantage as part of a display engine configuration. You can save a display engine configuration and load it at a later time.

To view a scene and simulation data, a display engine must have a window. VR-Vantage supports the following types of windows:

- Full screen. A full screen window uses the entire area of a monitor. It does not have any menus, panels, or other GUI controls.
- Resizable. A resizable window can be moved, resized, maximized, and minimized. It does not have any GUI controls. An inset view is an example of a resizable window.
- Embedded. An embedded window is part of an application. The main display window in VR-Vantage is an embedded window. The application provides a variety of controls that affect the view in the window. You could, if you wanted to, change VR-Vantage's window to a different type, in which case it would no longer be embedded in the VR-Vantage window.

A window can have zero or more channels. Each channel defines a viewable area on the screen. You must have at least one channel to see a rendered image in the window. Each channel can have its own observer.

The Observer

The observer, or eyepoint, is the location in the 3D or 2D environment from which you observe the scene. You can move the observer (navigate) through the scene. You can attach the observer to an entity or prop. When the observer is attached to an entity, it moves automatically with that entity. You can also control the observer from a remote application using view control messages.

You can have multiple observers in a VR-Vantage session. They might be associated with different windows. For example you could have an observer for the main window and one for an inset window.

Observer Modes

Observer modes let you configure VR-Vantage settings that affect how the observer moves and what it sees. You can then quickly change from one group of settings to another by switching between observer modes. An observer mode specifies:

- A model set. A collection of models of a particular type, such as 3D models.
- A key map. A coordinated set of key mappings to movement functions.
- A projection (2D or 3D).
- Settings for entity graphics and observer movement.

Table 3 lists the observer modes and default settings provided with VR-Vantage applications.

Table 3: Observer modes

Application	Observer Mode	Projection	Model Set	Key Map
VR-Vantage IG	Out-the-Window	3D	3D Models	Observer Frame
	Inset	3D	3D Models	Observer Frame
VR-Vantage Stealth	Stealth Out-the-Window	3D	3D Models	Observer Frame
VR-Vantage XR	Stealth Out-the-Window Inset	3D	3D Models	Observer Frame
	XR	3D	3D Colorized Models	Observer Frame
	Plan View	2D	2D Icons	2D Level Observer Frame
VR-Vantage PVD	Plan View	2D	2D Icons	2D Level Observer Frame
	Inset	2D	2D Icons	Observer Frame
VR-Vantage FreeView	Out-the-Window	3D	Not applicable	Observer Frame
	Plan View	2D	Not applicable	2D Level Observer Frame

The observer modes all support the three model sets: 3D models, 3D colorized models, and 2D icons. You can edit observer modes and add new ones.

VR-Vantage Object Modeling

VR-Vantage applications can display the simulated world using different projections (2D and 3D). In each projection, it can display appropriate types of models (2D icons, 3D models, and so on). Models are organized into model sets, and at any one time only the models in a particular model set are used.

VR-Vantage can apply various types of visual effects to a model, such as trailing effects, height-above-terrain lines, track histories, and so on. Some of these effects are appropriate to a particular projection, some are available for both projections. Visual effects are implemented using visualizers.

Visual effects and model sets are managed by the integration of schemas, model definitions, element definitions, and model mapping.

- A schema is a set of parameters for a scene element (such as an entity). VR-Vantage uses schemas to validate model definitions.
- A model definition specifies values for the required and optional parameters for an instance of a schema. For example the model definition for an M1A2 tank is an instance of the Entity schema. The model definition for a lifeform is an instance of a DiGuyCharacter schema. Of particular importance, a model definition specifies the 3D model to be used to represent an element in the scene. (2D icons are specified by the Military Symbol Icon Visualizer in the entity definition.)
- Element definitions (where elements can be entities, tactical graphics, or interactions) configure visual definition schemas on scene elements. For any given element, you can create a definition for each model set in which you want the element to be visualized. For example, you might want to visualize a tank in the 3D Models, 2D icons, and 3D Colorized Models model sets. You might want a tree to be visualized only in the 3D Models model set.
- Model definitions are associated with particular entity types by mapping an element definition to an entity type enumeration.

Figure 2 illustrates these relationships using an M1A2 entity. The model definition called Tracked_M1A2_DESERT uses an Entity schema. One of the parameters of the schema is the filename, which for this model definition is defined as *M1A2.medf*. The entity definition called M1A2 Abrams has a Model Visualizer. The modelDefinition attribute in the visualizer is specified as Tracked_M1A2_DESERT, which means that it will use the *M1A2.medf* model. The entity type 1:-1:-1:-1:-1:-1:-1:-1 is mapped to the M1A2 Abrams entity definition. So, when VR-Vantage discovers an entity with the enumeration 1:-1:-1:-1:-1:-1:-1:-1, it will look up the M1A2 Abrams entity definition, which will tell it to use a Model Visualizer for this entity that uses the Tracked_M1A2_DESERT model definition, which specifies that VR-Vantage use the *M1A2.medf* model to render the entity in the scene.

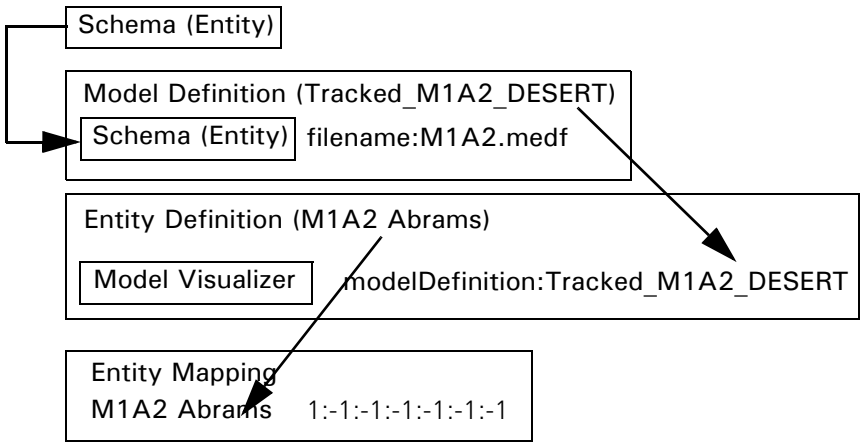

Figure 2. Element definitions and model mapping

All model definitions and model mapping is configured in the graphical user interface (GUI).

Model Sets

Each VR-Vantage application includes one or more of the following model sets:

- 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 icons. This is the default model set for the Plan View observer mode.

Table 4 lists the VR-Vantage model sets and the VR-Vantage applications that support them.

Table 4: VR-Vantage model sets

Model Set	Application
3D Models	VR-Vantage Stealth VR-Vantage IG VR-Vantage XR
3D Colorized Models	VR-Vantage XR
2D Icons	VR-Vantage XR VR-Vantage PVD

Mapping Models to Entities and Objects

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

DIS and HLA objects are specified by an entity type enumeration. An entity type enumeration is a list of seven components as specified by the DIS protocol and HLA RPR FOM. If none of the seven components contains a wildcard (-1) value, then the entity type refers to a specific entity. If some components contain a wildcard, then the entity type refers to a class of entities. A larger number of wildcards indicates a broader, more general class.

All model mapping in VR-Vantage applications is conducted through the graphical user interface (GUI). THere are no configuration files to edit.

Third-Party Software and Content

VR-Vantage includes software and content licensed from third parties, including:

- SilverLining™: real-time sky and 3D cloud rendering from Sundog Software
- GL Studio®: interactive cockpit instrumentation and HMI from DiSTI®
- DI-Guy™: realistic human character animation from Boston Dynamics
- SpeedTree®: animated, 3D foliage from Interactive Data Visualization (IDV)
- 3D models, terrain, and graphical content from Simthetiq, RealDB, TurboSquid, Terrex (now Presagis), and TerraSim
- OpenSceneGraph: an open source 3D graphics toolkit hosted at http://www.openscenegraph.org
- osgEarth: an open source streaming terrain plug-in by Pelican Mapping, at http://osgEarth.org.

The run-time and developers' rights for VR-Vantage customers vary from vendor to vendor.

SilverLining

VR-Vantage uses SilverLining software and content to compute lighting and to render the sky, clouds, sun, and moon. SilverLining is developed by Sundog Software (http://www.sundog-soft.com).

Most VR-Vantage customers do not need to buy a SilverLining license of any kind. In general, a VR-Vantage customer has the right to use the SilverLining technology that is built into VR-Vantage in any VR-Vantage-based application (including custom applications built using the Toolkit).

GL Studio

VR-Vantage uses GL Studio software and content to render interactive cockpit instrumentation displays. GL Studio is developed by DiSTI (http://www.disti.com).

VR-Vantage is delivered with several generic cockpit displays. You can use the built-in cockpit displays in any VR-Vantage-based application (including custom applications built using the Toolkit), without a GL Studio license of any kind.

DI-Guy

VR-Vantage uses DI-Guy software and content for human character animation. DI-Guy is developed by Boston Dynamics (http://www.bostondynamics.com).

VR-Vantage comes with DI-Guy functionality built-in, and with a large set of DI-Guy characters, appearances, and animations. A VR-Vantage customer does not need a DI-Guy license of any kind to use DI-Guy animated characters in a VR-Vantage-based application (including custom applications built using the VR-Vantage Toolkit).

SpeedTree

VR-Vantage uses SpeedTree software and content for animated real-time 3D foliage and vegetation. SpeedTree is developed by Interactive Data Visualization (IDV) (http://www.speedtree.com).

A VR-Vantage customer does not need a SpeedTree license of any kind to use the SpeedTree functionality that is built into the VR-Vantage applications that MÄK delivers. You may build and run plug-ins to our out-of-the-box applications without a SpeedTree license, as long as your plug-ins do not need to directly access the SpeedTree API.

3D Models, Terrain, and Graphical Content

VR-Vantage includes a rich set of 3D models for vehicles, weapons, cultural features and urban clutter (signs, barriers, lampposts, and so on). It also includes several sample terrain databases to help you get started with VR-Vantage, and to help demonstrate VR-Vantage. Much of this content is derived from 3D data that we have licensed from third parties:

- Many of the high-quality vehicle models come from Simthetiq (http://www.simthetiq.com).
- Some models come from Real DB (http://www.realdb.qc.ca/company)
- Some of the middle-eastern building models and urban clutter objects that are used in our sample terrain databases were licensed from Turbosquid (http://www.turbosquid.com)
- The TerraSim_SampleUrban terrain database comes from TerraSim (http://www.terrasim.com)

- The HarrisAfghanVillage terrain database was created using source data from Harris Inc.
- The Berkeley Database was developed by Terrex (now Presagis) (http://www.presagis.com).

All of this content is distributed in the VR-Vantage package solely so that VR-Vantage can use it. Use of any of the VR-Vantage models, textures, terrains, or other content outside of VR-Vantage-based applications is not permitted by the VR-Vantage license.

OpenSceneGraph

VR-Vantage uses OpenSceneGraph for its underlying scene graph representation, rendering, file loading, and many other capabilities. OpenSceneGraph is an open source, cross-platform graphics toolkit for the development of high-performance graphics applications. The OpenSceneGraph source repository is maintained by Robert Osfield at http://www.openscenegraph.org/projects/osg. It is distributed under the OpenSceneGraph Public License (OSGPL), which is based on the GNU Lesser General Public License (LGPL).

osgEarth

VR-Vantage uses osgEarth to import streaming terrain elevation and imagery data. The data can be streamed from external servers and sources or from a directory on the computer running VR-Vantage. osgEarth is an open source plug-in to OpenSceneGraph, maintained by Pelican Mapping at http://osgEarth.org. osgEarth is distributed under the GNU Lesser General Public License (LGPL).

Common Capabilities of VR-Vantage Applications

This section describes features that are common to all of the VR-Vantage applications.

Coordinate Systems

VR-Vantage supports the following coordinate systems: Geocentric, UTM, Lambert, Flat Earth, and Cartesian.

Entity Display

VR-Vantage has many ways to enhance your understanding of entity movement and actions.

Fire and Detonation Lines

When an entity fires a munition, a line is drawn between the entity and the location of the detonation. The line is colored blue, red, or green for the force of the shooter.

Inset Views

You can display inset views of individual entities. Figure 3 shows a helicopter hovering over the runway in the Makland terrain with an inset view of an M1A2 tank driving through the town.

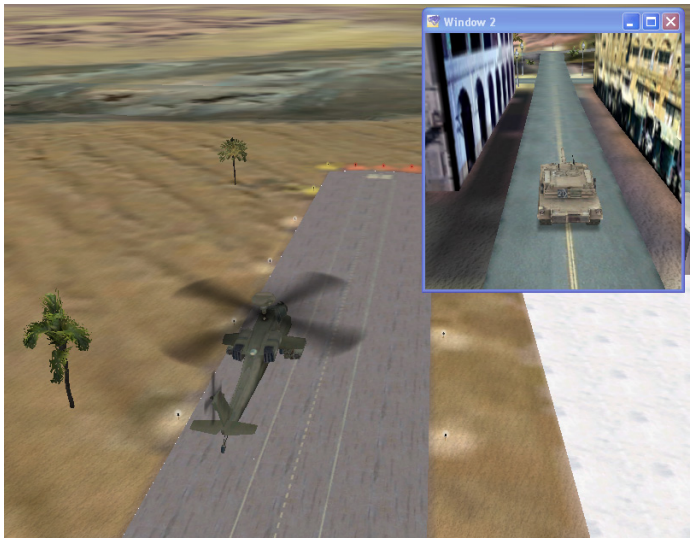

Figure 3. Inset view

You can change the observer position and orientation in an inset view using keyboard controls.

Sensor Volumes

VR-Vantage can display sensor volumes for entities that have emitter systems (Figure 4). You can configure the colors used to represent emitter frequencies and you can configure the size of the sensor volumes.

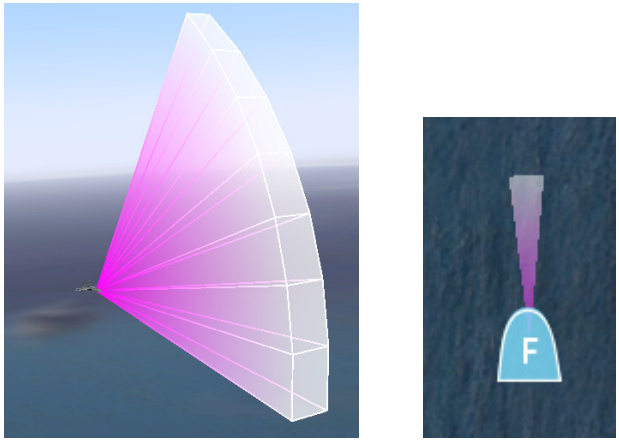

Figure 4. Sensor volumes

Track Histories

Track histories allow you to track the paths of entities during an exercise, including the path of munitions as they travel from the shooter to the target. When track histories are enabled, each entity leaves a track ribbon. The ribbon is colored based on the entity's force.

Filtering Entities Using Interest Management

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

Flexible, Intuitive Graphical User Interface

VR-Vantage's graphical user interface (GUI) allows you to manipulate VR-Vantage from menus, dockable control panels, and the keyboard.

You can hide and display the control panels, keep them docked to the screen, or undock them to make more space available for the display window. The GUI controls can be hidden for full-screen demonstrations.

The GUI follows standard conventions for modern windowing systems.

Persistent Settings

The VR-Vantage GUI remembers the settings as users change them. Settings can be restored back to factory defaults or back to the state they were in when the application was started. Most settings can be exported from one VR-Vantage application and imported into another.

Intuitive Navigation Modes

Control the absolute position of the observer with the keyboard and/ or mouse. Conveniences like speed scaling and terrain constraint make navigation easy. Or attach to various entities for a more focused view.

Trajectory Smoothing

VR-Vantage can smooth the trajectories of moving vehicles to compensate for discontinuous positional data.

Localization

The VR-Vantage graphical user interface can be localized using the Qt Linguist tool, which is included with VR-Vantage.

Multiple Attach Modes

VR-Vantage offers attach modes that let you:

- Manually control the observer's position and orientation.
- Follow an entity, maintaining a consistent positional offset, with a matching heading.
- Mimic a vehicle's position and orientation. You can place the observer inside the vehicle for a driver's-eye view, or fly outside of it, moving in tandem with the vehicle.
- Tether the observer to an entity, maintaining a consistent positional offset without changing the observer's heading.
- Automatically track a vehicle's movement from a fixed viewpoint, as if watching from a control tower.

Overlays

Tactical graphics and other visual data is displayed on overlays. You can enable and disable the display of graphics individually and by overlay.

Performance Features

VR-Vantage has features that help you understand its performance. It also has features that help you improve performance.

Performance Statistics Viewer

VR-Vantage can display two kinds of performance statistics — OSG statistics (Figure 5), and VR-Vantage statistics (Figure 6).

The StatsHandler class in the osgViewer library can gather and display the following rendering performance information.

- Frame rate. osgViewer displays the number of frames rendered per second (FPS).
- Traversal time. osgViewer displays the amount of time spent in each of the event, update, cull, and draw traversals, including a graphical display.
- Geometry information. osgViewer displays the number of rendered osg::Drawable objects, as well as the total number of vertices and primitives processed per frame.

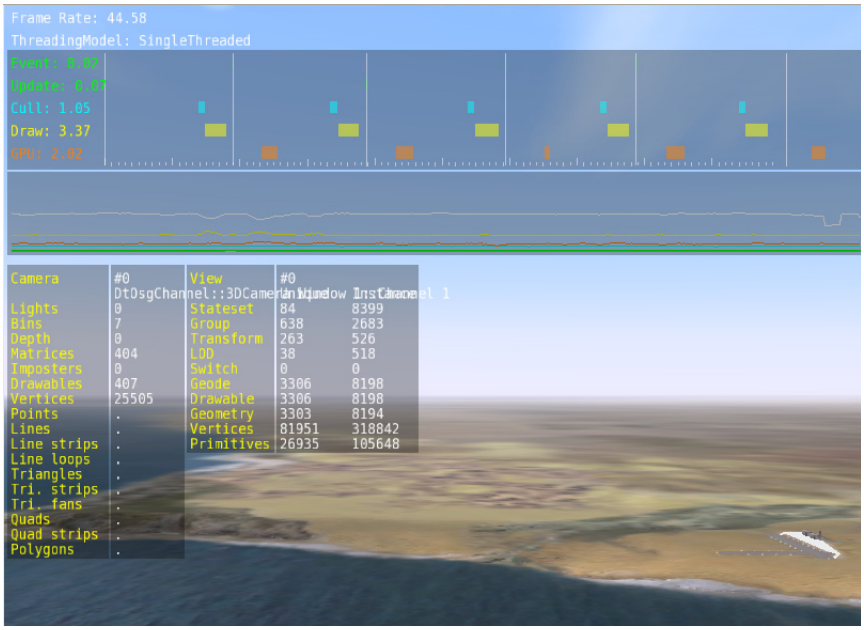

Figure 5. OSG performance statistics

The OSG statistics view can degrade performance. The VR-Vantage statistics view provides a limited set of statistics without affecting performance.

Figure 6. VR-Vantage performance statistics

File Caching

VR-Vantage will save loaded files into a fast loading format so that they are quickly loaded from disk the next time.

Instancing

VR-Vantage will keep instances of objects in memory and clone or reference them instead of loading new copies of the same data from disk again.

Remote Control

You can control VR-Vantage remotely through view control messages, which can be generated by MÄK applications such as the Logger and VR-Forces, and by programs that use the Stealth Control Toolkit or the VR-Vantage Control Toolkit.

Saved Views

VR-Vantage supports saving an observer's view at any moment. Saved views can be loaded at startup or during runtime.

Scenes

A scene contains an environment, the time of day, and possibly, a terrain. You can save and load scenes. If particular environment settings are important to the simulations you are viewing, use of scenes can be an important convenience feature. A scene does not have to have a terrain. For example, when you start VR-Vantage, a default scene is loaded that does not have a terrain.

Scene Graph

VR-Vantage provides a widget for inspecting the scene graph at any time. This is a valuable tool for debugging any VR-Vantage application.

Startup Options

On Windows, you can start VR-Vantage applications using any of the usual methods, including the Start menu, desktop icons, and taskbar icons. On both Windows and Linux, you can start VR-Vantage applications from the command line and using batch files or shell scripts. VR-Vantage has many command-line arguments that you can use to customize startup and quickly load preferred configurations.

Support for HLA, DIS, and CIGI

VR-Vantage can receive data from external simulations and be controlled by them using standard simulation protocols. You can connect and disconnect from simulations during runtime and can configure the connections in the graphical user interface (GUI). VR-Vantage supports the following simulation network protocols:

- HLA 1.3 and 1516. VR-Vantage supports the HLA 1.3 specification and the current draft of the IEEE 1516 C++ API maintained by the SISO Dynamic Link Compatible RTI API Product Development Group. It has built-in support for the HLA RPR FOM and can support other FOMs through the FOM Mapping feature.
- Distributed Interactive Simulation (DIS) protocols DIS 4, 5, and 6.
- Common Image Generator Interface (CIGI) 3.2 and 3.3. CIGI is an interface that allows an image generator (IG) to receive messages from a host to control what is drawn by the IG.

Tactical Graphics

VR-Vantage displays tactical graphics (routes, points, areas and so on) published by VR-Forces.

Terrain Display Features

VR-Vantage lets you visualize the terrain in ways that helps you understand it better:

- Elevation coloring. Color a terrain by elevation for height cues.
- Contour lines. Display contour lines for viewing slope.
- Wireframe mode. View the scene in wireframe mode to help understand its structure.

VR-Vantage 3D Display Features

VR-Vantage XR, Stealth, and IG deliver a realistic, three-dimensional view of terrain, vehicles, and other objects. VR-Vantage's navigation controls give you complete freedom to travel anywhere as an unobtrusive observer. You can place the observer at a specific location, or link it to a vehicle for automatic tracking of battlefield activities. While an exercise is underway, you can switch attach modes quickly to see the scene from any perspective. VR-Vantage can be visible to other applications or invisible, at your discretion.

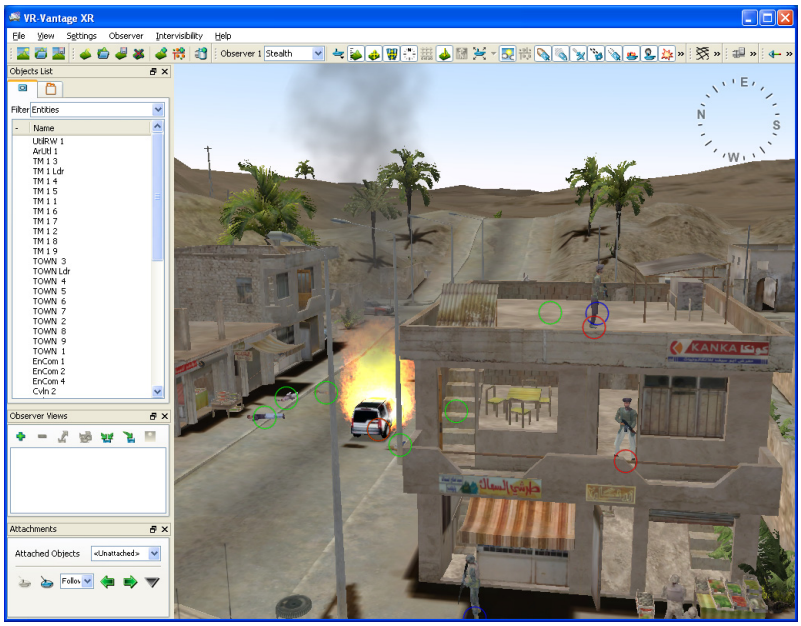

Figure 7. VR-Vantage XR 3D

Realistic 3D Display of Vehicles and Terrain

The VR-Vantage 3D projection provides a realistic display using terrain and moving-model geometry in the OpenFlight format. It includes many 3D vehicle models, some of which display movement of articulated parts, such as turrets and rotors. These models can change to show a damaged or destroyed state. You can provide your own models and map them to entity types.

VR-Vantage also displays:

- Smoking and flaming effects
- Trailing effects, such as footprints, wakes, missile trails, and dust clouds
- Muzzle flashes
- Track histories
- Explosions.

VR-Vantage presents terrain realistically, and lets you control visibility, time-of-day, and other effects.

VR-Vantage can render real time shadows for entities and lifeforms based on the position of the sun.

Ground Clamping

The ground clamping feature can ensure that all ground entities are placed correctly on the terrain surface.

Support of Stereoscopic Displays

Support for anaglyphic stereo and polarized stereo.

Ephemeris Model

Sun, moon, stars and time of day accurately modeled. Clouds, precipitation and visibility as well.

2D Display Features

Visualization of Feature Data

VR-Vantage can display the raw feature data of shape files (or VMAP) based on color coded mappings.

Application-Specific Features

The next few sections describe features that are specific to the different VR-Vantage applications.

VR-Vantage IG

VR-Vantage IG supports the common set of VR-Vantage terrain, object modeling, and display features. VR-Vantage IG's default observer settings are customized for an out-the-window IG user rather than the information-station focus of VR-Vantage Stealth.

VR-Vantage Stealth

VR-Vantage Stealth supports the common set of VR-Vantage terrain, object modeling, and display features. It also supports the 3D display features previously described. The Stealth's default observer settings are oriented towards providing situational awareness and simulation information.

Intervisibility

You can display intervisibility (line-of-sight or LOS) between points, within areas, and between a point and all entities in a defined area. Intervisibility displays can be transient or permanent. Permanent intervisibility displays are objects. You can resize them and move them around the terrain. Intervisibility objects can be tied to a particular entity.

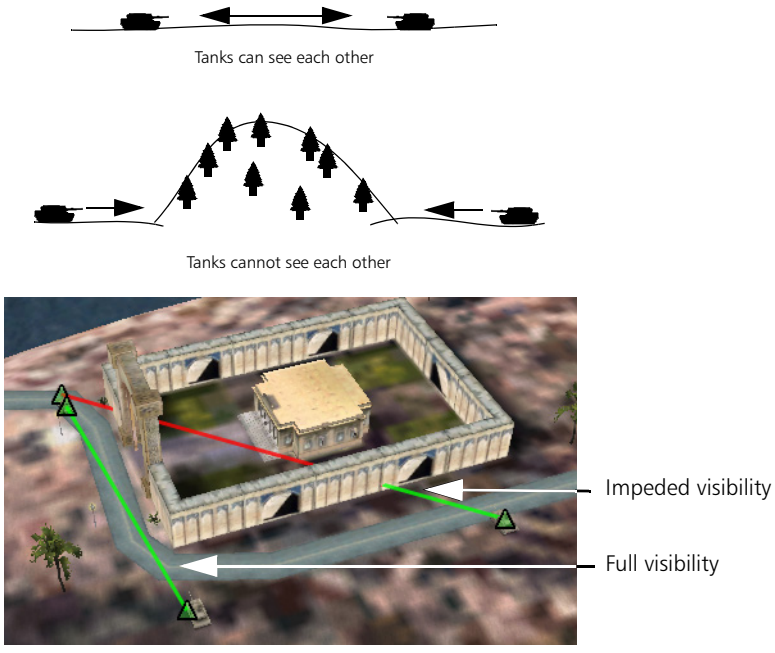

Figure 8. Intervisibility

VR-Vantage PVD

VR-Vantage PVD supports the common set of VR-Vantage terrain, object modeling, and display features. It displays entities using the MILSTD 2525B icon set. Entity icons have heading indicators and can display text labels adjacent to the icon that show several types of information about entities as, including name, orientation, velocity, and acceleration.

VR-Vantage PVD can display a grid line overlay. You can configure the coordinate system used for grid lines, the precision, and spacing. Spacing can be fixed, in meters, or determined by VR-Vantage using a specified number of screen pixels.

VR-Vantage PVD supports intervisibility, as described in the previous section.

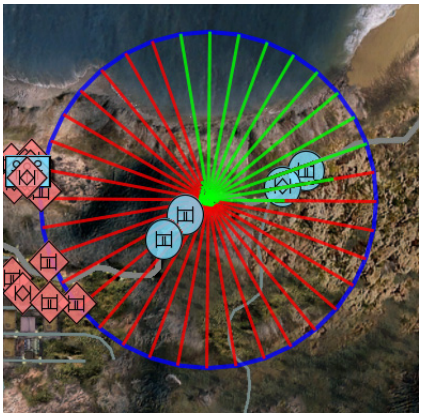

Figure 9. Intervisibility fan

VR-Vantage XR

VR-Vantage XR supports the common set of VR-Vantage terrain, object modeling, and display features, all of the 3D features of VR-Vantage Stealth, and all of the 2D features of VR-Vantage PVD. Then it adds the XR model set to enhance your ability to understand the layout of forces in the 3D projection. The notional icons are scaled to be visible regardless of the observer's distance from the entities. You can resize the icons to ensure that you have the right view for your needs.

VR-Vantage XR also supports scaling of the realistic 3D model set.

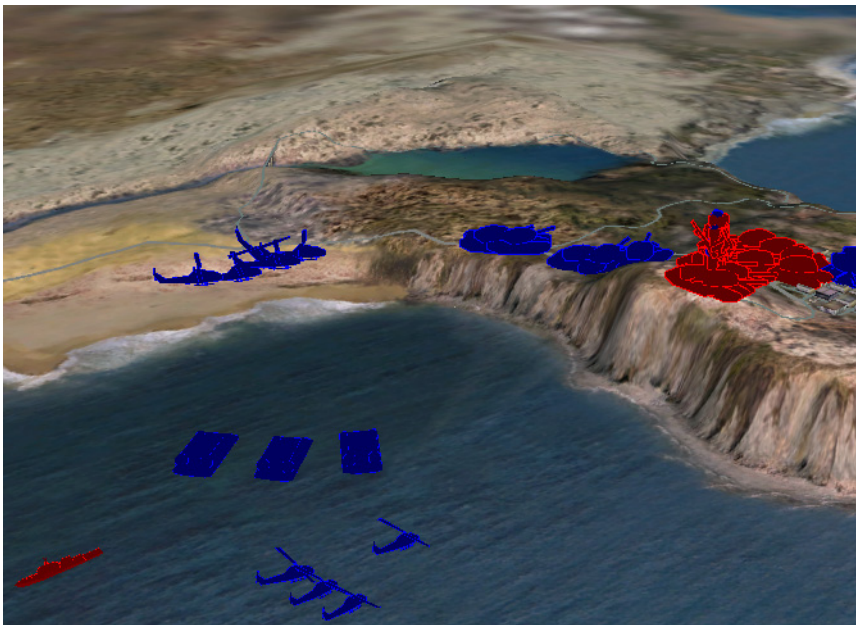

Figure 10. Model scaling and colorized models

VR-Vantage FreeView

VR-Vantage FreeView supports VR-Vantage's 3D and 2D terrain viewing and composing features. It cannot connect to simulation networks. VR-Vantage FreeView also has one feature that is unique – the model viewer. VR-Vantage FreeView lets you view 3D models to see how they will look when you load them in a VR-Vantage application. You can view models in the MEDF, OpenFlight, 3ds, and OBJ formats. (MEDF is MÄK Encrypted Data Format.)

You can use the navigation keys to rotate the model so that you can see it from all sides.

SensorFX

In addition to the VR-Vantage applications described above, MÄK offers SensorFX, a plug-in for the VR-Vantage 3D applications. SensorFX changes VR-Vantage from a visual scene generator to a sensor scene generator. SensorFX models the physics of light energy as it is reflected and emitted from surfaces in the scene and as it is transmitted through the atmosphere and into a sensing device. SensorFX also models the collection and processing properties of the sensing device to render an accurate electro-optical (EO), night vision or infrared (IR) scene.

Extensive Sensor Coverage SensorFX enables you to credibly simulate any sensor in the 0.3-16.0um band with VR-Vantage, including:

- FLIRs / Thermal Imagers: 3-5 & 8-12um
- Image Intensifiers / NVGs: 2nd & 3rd Gen
- EO Cameras: Color CCD, LLTV, BW, SWIR

VR-Vantage Toolkit

The VR-Vantage Toolkit allows developers to design and deploy realtime 2D and 3D visual applications for the simulation and training domain. It is a cross-platform C++ API based on OpenSceneGraph (OSG). Through the VR-Vantage Toolkit, developers can extend existing VR-Vantage applications, create their own applications or embed visuals into existing applications.

Application Framework

The VR-Vantage Toolkit provides an application framework for users to easily build custom applications using pre-built or even custom components.

Agents

Distributed rendering is made easy through the use of agents… classes created by our code generator that cause objects to be created and added to the scene graph of each display engine.

Design Patterns

Heavy use of design patterns makes understanding a complicated system easy. Architecture specific design patterns make adding new features quick.

Drivers

Adding simulation code or connecting to an existing networked simulation is done via the simulation driver pattern.

Forests

SpeedTrees are handled efficiently through use of forest classes.

Geometry Manipulation

Many 3D applications require modification of loaded geometry to achieve desired performance or rendering results. The osgUtil library contains classes to provide several types of common geometrical operations.

GUI Components

Menus, toolbars, dialogs, pages and doc widgets are all easily added to VR-Vantage applications.

Intersection

Typically, 3D applications need to support user interaction or selection, such as picking.

VR-Vantage has a variety of intersection utilities for this purpose. Additionally the osgUtil library efficiently supports picking with a variety of classes that test the scene graph for intersection.

Lighting

OSG fully supports OpenGL lighting, including material properties, light properties, and lighting models. Like OpenGL, light sources aren't visible-OSG doesn't render a bulb or other physical manifestation. Also, light sources create shading effects, but don't create shadows.

Memory Management

OSG provides an automated garbage collection system that uses reference counted memory. All OSG scene graph nodes are reference counted, and when their reference count decrements to zero, the object deletes itself. As a result, to delete the scene, your application simply deletes the pointer to the root node. This causes a cascading effect that deletes all the nodes and data in the scene graph. VR-Vantage typically handles the deletion of the scene graph for you.

Multi-Texture

VR-Vantage can use multi-textures to add detail to terrain skins without adding geometry.

Network Visualizers

Rendering simulations transmitted over DIS and HLA is made easy via the visualizer pattern.

Node Types

The VR-Vantage Toolkit provides users with a variety of nodes that can be added to the scene graph for different purposes.

- Node. The Node is the base class for all nodes in the scene graph. It contains methods to facilitate scene graph traversals, culling, application callbacks, and state management.
- Group Node. The Group node is the base class for any node that can have children. It is a key class in the spatial organization of scene graphs.
- Geode. The Geode (or Geometry Node) corresponds to the leaf node in OSG. It has no children, but contains osg::Drawable objects (see below) that contain geometry for rendering.
- DOF (Degree of Freedom). Node allows children to be articulated within a set of constraints.
- Geometry Drawable. A set of polygons for drawing.
- LOD (Level of Detail) Node. The LOD node displays its children based on their distance to the view point. This is commonly used to create a varying levels of detail for objects in a scene.
- Switch Node The Switch node contains a Boolean mask to enable or disable processing of its children.
- Channel Specific Group Node. Renders a specific child based on the channel being rendered.
- Matrix Transform. The MatrixTransform node contains a matrix that transforms the geometry of its children, allowing scene objects to be rotated, translated, scaled, skewed, projected, etc.
- Sequence
- PositionAttitudeTransform

NodeKits

NodeKits extend the concept of Node, Drawable, and StateAttribute objects, and can be thought of as extensions to the OSG library in core OSG. NodeKits must do more than derive from an OSG class. They must also provide a dot OSG wrapper (an OSG plugin to support reading from and writing to an OSG file). As a result, a NodeKit is composed of two libraries: the NodeKit itself, and a dot OSG wrapper plugin library.

OpenSceneGraph

The OpenSceneGraph can be accessed directly and easily for quick integrations. Existing or new OSG nodekits can be added this way.

Proxy Objects

Proxy objects allow the master application to control windows and channels on remote display engines.

Render Bins

A render bin is a bucket into which geometry and OpenGL commands can be stored in. These are used to ensure certain things are rendered before other things. Users can use the existing bins or create their own.

Serialization

Data and settings are easily serialized in XML via our record pattern.

State Management Classes

OSG provides a mechanism for storing the desired OpenGL rendering state in the scene graph. During the cull traversal, geometry with identical states is grouped together to minimize state changes. During the draw traversal, the state management code keeps track of the current state to eliminate redundant state changes.

Texture Mapping

VR-Vantage supports multiple types of texture. You can set the following modes: GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D, GL_TEXTURE_CUBE_MAP, GL_TEXTURE_RECTANGLE, GL_TEXTURE_GEN_Q, GL_TEXTURE_GEN_R, GL_TEXTURE_GEN_S, and GL_TEXTURE_GEN_T.

Visitors

NodeVisitor is OSG's implementation of the Visitor design pattern [Gamma95]. In essence, NodeVisitor traverses a scene graph and calls a function for each visited node. Visitors are used during culling, updating, drawing and for intersections and picking, among other things.

The VR-Vantage Control Toolkit

The VR-Vantage Control Toolkit lets developers send VR-Vantage control messages from other applications. Move the observer, change a setting or even create graphics on a channel can all be done remotely through an HLA or DIS interface.

VR-Vantage Debug Notification

Debug and error messages can be sent to the console window or within the application's error message box. Messages can be tagged with a severity level to allow users to display only certain classes of messages at any particular time.

OSG is capable of displaying a large amount of debugging information to std::cout. This is useful in developing OSG applications, because it provides insight into what OSG is doing. The OSG_NOTIFY_LEVEL environment variable controls how much debugging information OSG displays. You can set it to one of seven values for varying verbosity levels. ALWAYS (least verbose), FATAL, WARN, NOTICE, INFO, DEBUG_INFO, and finally DEBUG_FP (most verbose).

For typical OSG development, set OSG_NOTIFY_LEVEL to NOTICE, and adjust the value up or down the verbosity scale as necessary to receive more or less output.