Virtual Reality Scene Generator (VRSG) Version 7 User's Guide

Geospecific simulation with game quality graphics™

MVRsimulation®

MVRsimulation Inc.
57 Union Avenue, Sudbury, MA 01776 USA
Phone: 617-739-2667 Fax: 617-249-0151
Website: www.mvrsimulation.com Email: sales@mvrsimulation.com

Contents

CHAPTER 1

Introduction

MVRsimulation® Virtual Reality Scene Generator® (VRSG®) is a state-of-the-art, real-time computer image generator (IG) designed to visualize geographically expansive and detailed virtual worlds on Windows PCs. VRSG provides real-time single- or multiple-channel visualization of virtual environments, dynamic moving models, and special effects using Microsoft DirectX commercial standards.

Real-time VRSG screen capture of an F-15E entity in flight over the virtual Okinawa airport on MVRsimulations' new 3D terrain of Japan.

VRSG is a component of MVRsimulation's rapid virtual world terrain creation and visualization technology. With MVRsimulation's technology, users can build and visualize at 60 Hz frame rates virtual worlds created with real-world photographic imagery (such as aerial and satellite imagery), elevation feature data, and pattern-of-life scenarios. The goal of this technology is to deliver geospecific simulation with game quality graphics™ on displays that range from VR devices, simulated military equipment, laptop and desktop computers, to immersive training domes.

VRSG is an executable-ready Microsoft-based render engine that does not require compilation for use. It uses Windows, DirectX, and Visual C++. Configuration files provide you with the ability to control basic components of the render engine, as described in the chapters "Configuring Models and Events" and "Integrating 3D Sounds." To evolve VRSG, MVRsimulation routinely interviews its customers and determines features that are most needed and then adds them to the product. MVRsimulation attempts to aggregate the most common features into VRSG regularly, which enables all customers to benefit from the advances initiated by other and complementary users.

To supplement the render engine with low-level features not available in core VRSG, you can use the plugin interface to compile in your own features, as described in the chapter "Developing VRSG Plugins."

VRSG's core features

VRSG supports the typical features required for image generators used in flight, ground vehicle, and infantry training simulators, and many other applications. IGs are typically driven by users' simulator host model, such as a flight model. VRSG renders a virtual world as it is specified by host parameters such as location and field-of-view.

The rest of this section describes some of VRSG's core image generator features.

Lights and lighting

VRSG lighting features include:

- Full-featured, high performance light points.
- High performance light lobes with per-pixel axial and radial attenuation.
- Dynamic lighting.

Light points

Light points are small self-luminous objects that do not cast illumination on the surrounding scene. They are generally used to model airfield lighting systems (such as VASI, PAPI, and taxiway lights). VRSG supports full-featured light points; processing runs entirely in vertex shader programs downloaded to the graphics chipset, providing exceptional performance. VRSG light points were developed with input from subject matter experts, such as commercial and military pilots. Light point features include:

- Per-light color and intensity; no performance degradation for varying color and intensity within a light string.
- Per-light phase shift with no performance degradation for varying phase within a light string.
- Period, duty cycle, rotation rate.
- Real-world size rendered perspective-correct.
- Automatic luminance compensation for size-clamped lights; prevents volume clamped lights from appearing brighter at further ranges.
- Automatic ground clamping and elevation placement at load time.
- Ability to disable directional attenuation on a per-edge basis to support sharp transitions for realistic VASI and PAPI lights.
- Visibility range as a function of light type and weather conditions; separate from terrain visibility.
- Direction specified in azimuth/elevation.
- Independent horizontal and vertical beam angles.
- ASCII description file for light customization apart from the terrain database.
- Minimum size.

All light points, including directional light points with unique per FOV edge attenuation behavior, run entirely in the vertex shader, providing exceptional performance.

Light lobes

Light lobes cast light, illuminating anything 3D in front of them (like vehicle headlights) or around them (like street lights, flashlights, or searchlights).VRSG provides realistic light lobes that yield per-pixel radial attenuation and axial attenuation. VRSG light lobes are flexible enough to support landing lights, taxi lights, headlights, and searchlights. VRSG light lobes do not require multiple database render passes or hardware that can store alpha information in the frame buffer. Instead, VRSG light lobes are rendered single-pass, which affords minimal performance degradation when enabling a light lobe. You can configure multiple concurrent, independent light lobes. You can also attach a light lobe to a model as discussed in the chapter “Configuring Models and Effects”. No drastic impact on fill rate or geometry processing penalties is incurred when enabling light lobes. VRSG supports concurrent, steerable light lobes for video cards that support Pixel Shader Model 5.0.

Dynamic lighting

VRSG supports an optimized dynamic lighting pipeline, which uses per-vertex color blended with per-polygon material, combined with ambient lighting conditions and directional light sources for efficient and convincing dynamic lighting effects. Per-pixel lighting provides true Phong shading and interpolates and normalizes both the surface normal vector and the reflectance vector for per-pixel ambient, diffuse, and specular contributions to lighting.

Normal maps can also be used to provide per-pixel perturbation of a polygon surface normal vector. Specular maps can be used to create reflective surfaces that reflect the sky model. These techniques are useful for building windows or aircraft canopies.

VRSG supports advanced multi-texture techniques such as shadow maps and light maps. Models can encode a shadow map and a light map together in a single second texture applied to faces. In daytime mode, a shadow map modulates the base texture providing more variability in intensity. In nighttime mode, the light map creates a diffuse local light source, simulating the effects of illuminated window of a building, or a local spotlight or floodlights.

Environment and weather

VRSG supports advanced environment and weather effects such as:

- Multiple atmospheric layers, ground fog, and haze.
- 3D ocean sea states, wave motion and wakes, vessel surface motion, accurate environment reflections.
- Sun angle-dependent haze color and density.
- Rain and snow.
- Volumetric ray-traced clouds, which are procedurally generated or user defined and controlled. Clouds can cast shadows on the terrain.

VRSG real-time littoral view of Costa Verde, Spain, showing the 3D ocean wave fall-off and transparency at the shoreline.

VRSG uses an ephemeris model to calculate sun position, moon position, star position, and moon phase from date, time, and geographic location. Lighting conditions can also be automatically calculated from date, time, and geographic location.

Effects

You can associate events with animation effects. VRSG supports:

- Billboard-based effects for rotating 2D textures such as trees, or for a flame or detonation effect.
- Particle-based effects for smoke plumes, tactical smoke, blowing sand, blowing dust, dust trails, rotor wash, explosions, and so on. These effects provide a greater degree of realism as they are volumetric and they interact with wind. VRSG is delivered with several solid particle ballistic effects, which model projectiles with dust trails that are cast from detonation events.

VRSG is delivered with 480 effects and configuration files to map the effects to DIS events. These effects range from smoke in various colors, different types of contrails and wakes, to explosions with a variety of characteristics, to a particle-based volumetric flame effect for burning vehicles. You can configure animated effects for one-time animations or for events with longer duration.

VRSG supports real-time dynamic flare effects, which, in addition to providing a convincing animation of a flare, also dynamically illuminate nearby terrain and model geometry. The animation sequence and scene illumination properties of each flare effect are user configurable.

2D overlays

VRSG supports multiple mechanisms for adding 2D overlays to the 3D display. VRSG includes built-in overlays for many popular Unmanned Aerial Systems (UAS)and targeting pods. For CIGI integrators, VRSG supports a large portion of the Common Image Generator Interface (CIGI) 4.0 symbology opcodes. More advanced integrations can develop overlays using VRSG's Plugin API and the DirectX Graphics API. VRSG has an OpenGL interoperability plugin which enables users to code their HUDs or other overlay graphics in machine-native OpenGL.

Viewports

A viewport is a single scene rendered in a VRSG channel. A single VRSG channel can be divided up into multiple viewports, or concurrent scenes, with each viewport assigned to a different view of the scene. Multiple viewports can overlap as in a picture-in picture arrangement or to be spatially disjoint as in picture-by-picture arrangement. Some simulation applications require multiple VRSG viewports; for example, UAS simulations often need dedicated viewports for both the nose camera as well as the articulated sensor camera. Many VR HMD devices require at least two viewports (one for each eye) or four viewports (two for each eye), depending on the manufacturer. Viewports can also service two or more projectors on a large dome display. Viewport limits are associated with the license for a VRSG channel. Multiple viewports are supported by applications using CIGI, not DIS-based interfaces.

Mission functions

VRSG supports basic mission functions requirements to meet needs ranging from ground–based vehicle simulators to fast moving fixed-wing aircraft. The VRSG environment shares the responsibility of mission functions with the host simulator. Terrain elevation, point-to-point intervisibility, collision detection, and intersection with dynamic models are all supported through CIGI version 4.0. Supported CIGI mission functions include Height Above Terrain (HAT), Height Of Terrain (HOT), line-of-sight vector requests, line-of-sight segment requests, and collision segments.

A library that can be integrated into the simulation host provides point-to-point intervisibility, terrain height lookup, and general coordinate conversion utilities to work with VRSG's round-earth coordinate system.

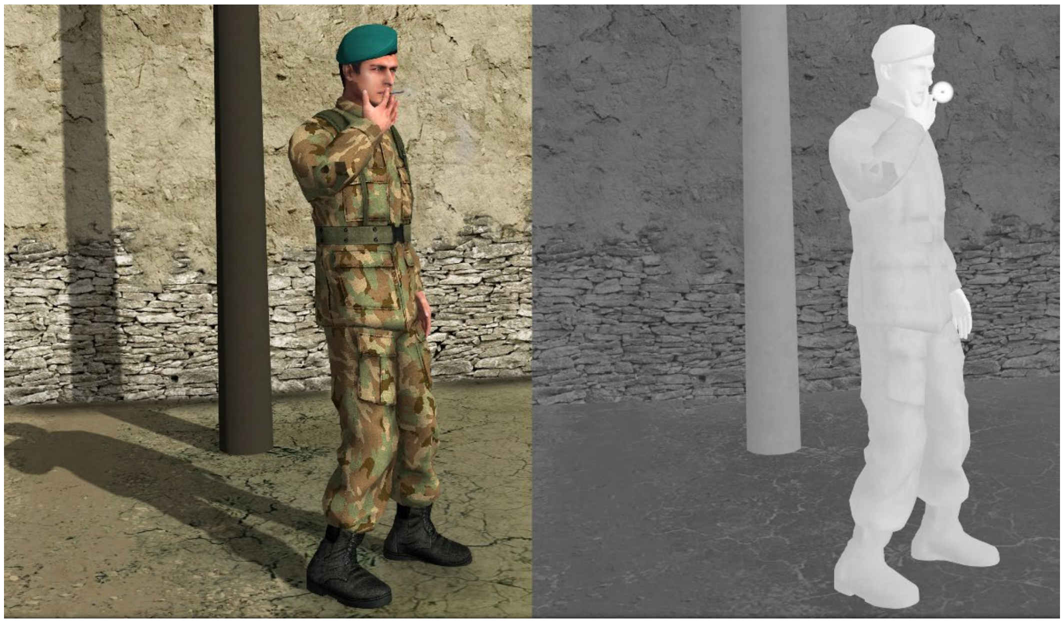
3D animated characters

VRSG can display hundreds of characters within the field of view while maintaining a high frame rate. Delivered with VRSG is a substantial model library of characters and weapons in MVRsimulation's 3D runtime model format which you can immediately configure and use in VRSG.

Real-time VRSG scene of a close air support team on MVRsimulation's geospecific terrain of Hajin, Syria.

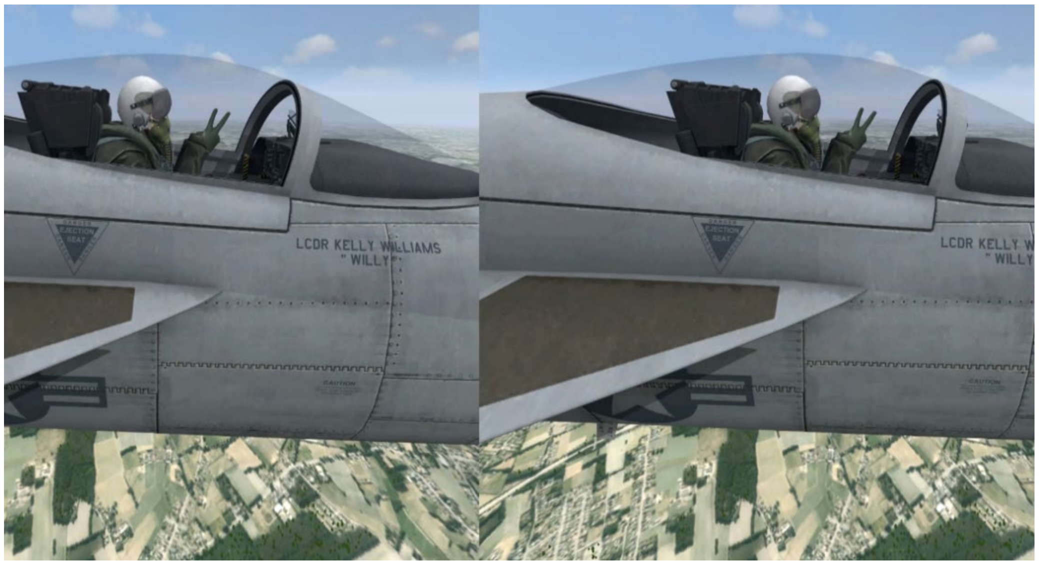
In addition you can use your own custom characters, weapons, and animations. Like a vehicle entity, a 3D character can be configured as an entity in VRSG through the ModelMap.ini configuration file. You can assign any character model to a DIS lifeform entity using its DIS enumeration.

VRSG also includes a wide range of character animations, which can be used in Scenario Editor for creating pattern of life scenarios. The animations portray all commonly used appearances required by the DIS protocol; VRSG automatically blends animation transitions smoothly.



VRSG real-time screenshot of a character animation of a character smoking a cigarette. Notice the smoke bloom in the IR sensor view.

You can animate a character’s individual fingers using 16-bone hand models. These hand models with articulated fingers can be used with our character models to simulate the gestures of non-verbal commands and communication, such as tactical combat hand signals. (The FBX rig for these hand models is available upon request.) Supporting this feature are 30 BVH animations for the skinned hand models, which were created using Autodesk Motion Builder animation software.

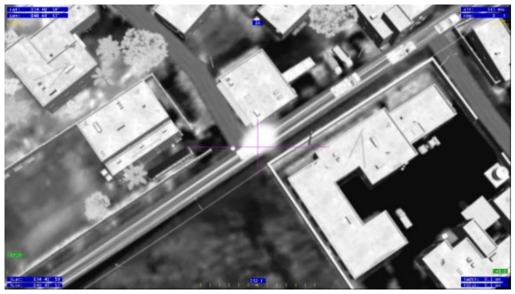


VRSG real-time screenshot taken in an HTC-VIVE Pro headset of character finger-based signal animation. The pilot character model is giving a hand signal for the Naval Air HEFOE code indicating the aircraft is having trouble with the electrical system.

Physics-based or material-based infra-red rendering

VRSG's physics-based IR uses a physics-based model licensed from TSC, in conjunction with IR rendering technology developed by MVRsimulation, featuring real-time computation of the IR sensor image directly from the visual database, without the need to store a sensor-specific database. (*Available upon ITAR approval for international customers.)*

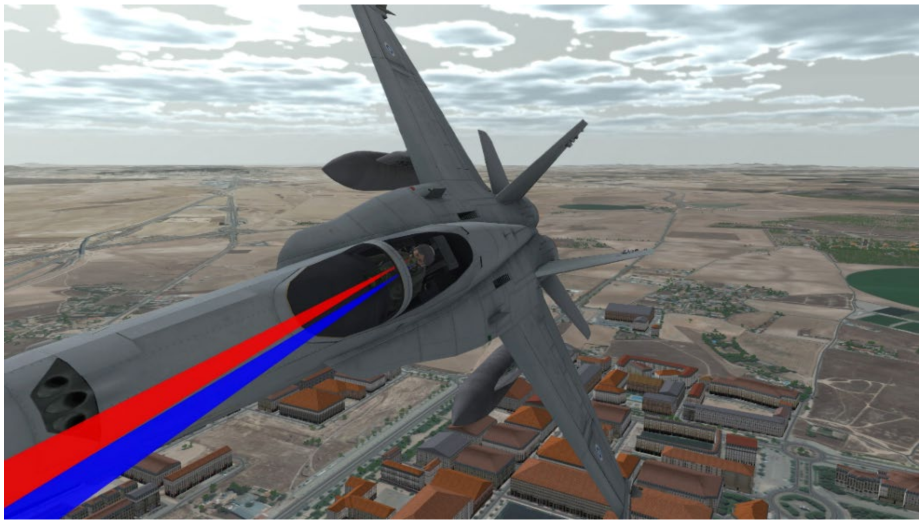
On a per-material basis, you can provide thermal radiance profile data as a function of time-of-day. The fidelity of the radiance profile data is under user control. Notional data may be supplied for installations requiring ITAR export compliance. You can also provide radiance profile data that was derived from a third-party physics-based model.



For simulating UAS video feed, VRSG includes a plugin to stream H.264/H.265 video, with MISB compliant 0601.9 KLV metadata. VRSG video and metadata has received full compliance from the MISB CMITT test suite.

Eye tracking

To take advantage of the eye-tracking technology in Varjo XR headsets, VRSG can visualize the gaze of a pilot (or ground personnel). During a simulated flight (or ground) training mission, VRSG can track the pilot's head position and orientation within the cockpit simulator, track the gaze vector using the Varjo device's pupil tracking functionality, and then depict the gaze of each eye independently as a color-coded 3D cone. Head position is depicted with a head model and pupil gaze direction is depicted with red and blue cones, as shown below.



At the end of the training mission, VRSG can export this data via DIS as a PDU log. During after-action playback, VRSG visualizes the pilot’s head position, orientation, and gaze vector over the events of the training session. From a tactical perspective, the eye-tracking playback can help identify missed moments of attention to instrumentation or the direction of an important activity. And it could be useful for beyond visual range systems management review or for critiquing basic fighter maneuvers. This setup can be used in any type of mixed-reality application.

Standalone features

The following additional capabilities are available in VRSG, independent of a simulation host model (using a 6DOF game controller for navigation as described next):

- Terrain paging algorithms to visualize expansive terrain limited only by users’ system storage capacity.
- Culling and continuous morphing levels of detail (LOD) techniques.
- Absolute terrain correlation with SAF formats.
- Teleportation to arbitrary locations.
- Real-time selection of sensor modes.
- Robust libraries of texture-mapped 3D entities and cultural feature models.
- Ability to save session configuration settings for use in subsequent sessions.
- Choice of coordinate systems and database projections.
- Support for 6DOF game devices that comply with the Human Interface Device (HID) standard.
- Support for multiple monitors and EDID emulators on headless systems.
- Synchronized multiple viewpoints for users to depict a contiguous virtual world across multiple display systems.
- Ability to render eye-tracking data captured by Varjo headsets.
- Dynamic shadows for all entities and cultural features.
- An extensive set of attachment modes.
- Ability to save and recall viewpoints.
- Editable 3D sounds associated with virtual world events and vehicle types.
- Ability to capture still images of the VRSG scene.
- Support for UDP multicast, broadcast, or point-to-point communication.
- Support for VR and mixed reality headsets that are compatible with OpenVR / SteamVR.
- Tools for converting FBX models and OpenFlight models and terrain to MVRsimulation’s model and terrain formats.

Navigation

You navigate through the virtual world by using a 6DOF game controller such as Logitech 3Dconnexion’s SpaceMouse Pro or SpaceMouse Compact device and the VRSG user interface. For fixed wing and UAV modes and for manipulating a character in First Person Simulator (FPS) mode, VRSG supports the Logitech F310 gamepad or an 8-button 4-axis joystick-type device that is compliant with the USB Human Interface Device (HID) specification.

These devices are supported by Windows and VRSG without the need to install any additional software.

Network capabilities

Using the DIS protocol VRSG can visualize up to several thousand entities simultaneously on a high-fidelity terrain database.

VRSG in one of the F-16C Block 30 simulators at Kelly Field (KSKF) at Lackland Air Force Base. (Photo courtesy of the U.S. Air Force.)

VRSG has no explicit limit on how many entities it can handle in a scene while maintaining real-time performance. Some of MVRsimulation's customers run exercises with upwards of 100,000 entities while maintaining real-time performance. The only limitations might be performance considerations such as the bandwidth of the communications channel, model complexity, database complexity, viewing range, and how near the entities are in your field of view.

VRSG is fully interoperable with semi-automated forces (SAF) applications and other DIS-compliant applications.

Interaction with SAF applications

As a DIS-based simulation application, VRSG is compatible with SAF applications such as OneSAF, JointSAF, XCITE, and commercial systems such as Battlespace Simulations' (BSI's) Modern Air Combat Environment (MACE®). Systems such as MACE and XCITE, which can utilize MVRsimulation's round-earth VRSG terrain tiles for their elevation data source, can use fully-correlated databases without the need for ground-clamping. Databases derived from third-party products, such as Presagis Terra Vista, can also achieve full correlation provided the VRSG terrain tiles were produced from the OpenFlight output of the same Terra Vista project that rendered the SAF representation of the database. As a last resort, VRSG offers a ground-clamping feature for cases where perfect correlation between VRSG and the SAF databases are not possible.

Aerial view of a portion of MVRsimulation's Afghanistan database rendered in BSI MACE, with VRSG providing the 3D view.

Correlated aerial view of the same area rendered in BSI MACE's 2D view. Images courtesy of BSI.

MVRsimulation round-earth 3D terrain database format

MVRsimulation's VRSG (MDS) terrain format is a round-earth terrain representation. This format is ideal for aerial applications, which require vast areas of terrain to use with VRSG. MVRsimulation Terrain Tools is currently used to build terrain for many manned and unmanned aircraft simulators. A round-earth terrain format has many benefits; most importantly, the terrain models the earth to a high degree of accuracy over its entire surface in contrast to a local approximation that is only valid over a relatively small range. This level of accuracy is vital for targeting applications and determining intervisibility. MVRsimulation initially designed this terrain format for aviation applications that require expansive – and possibly full world – coverage, a round-earth terrain model, and incremental database building.

The VRSG round-earth virtual terrain architecture represents the earth's surface in a geocentric coordinate system which accurately represents the curvature of the earth and handles ordinate axis convergence at the poles. This architecture solves a variety of problems associated with projection-based, monolithic visual databases, leading to improvements in database production, distribution, storage, and update, as well as in many run-time and mission function benefits.

A key component of the VRSG terrain format is the use of an atomic geometric primitive as the database tile: the triangle. The geographic coverage of the entire database is compiled as a mosaic of tiles. Database tiles typically represent a unit of database compiler output, a load module or unit of paging by the runtime system, and/or a unit of culling by the runtime system. Each tile is represented on disk by a single file. You can store tiles on multiple drives and directories.

Because the terrain is comprised of a collection of triangular tiles, you can visualize any set of tiles that you want to traverse; you can place a subset of generated tiles in a different directory from a larger set, and point VRSG to the location of only the subset.

Real-time VRSG screen capture from within the cockpit of an A-10 entity in flight over virtual Hajin, Syria.

You can use MVRsimulation's Terrain Tools to generate 3D terrain in VRSG (MDS) format from within your GIS software. You simply create a basic map in ArcGIS Pro and use Terrain Tools to generate 3D terrain in the round-earth VRSG format. The resulting terrain can be rendered in real time within VRSG. For complete information about generating VRSG-formatted terrain, see the *MVRsimulation Terrain Tools User's Guide*.

The VRSG software installation provides an MVRsimulation utility that converts OpenFlight-formatted databases to the VRSG terrain format. See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats" for information about using this utility.

Available from MVRsimulation in VRSG round-earth terrain format are datasets of Continental USA and AK and HI (CONUS++), the rest of North America, Africa, Asia, Australia and Oceania, Europe, and South America 3D terrain, with several high-resolution areas.

Scenario creation

Scenario Editor enables you to create and edit real-time 3D scenarios to play back in VRSG. Within its game-level editor type interface you can add culture and moving models directly to your 3D terrain and build dense 3D scenes with pattern-of-life scenarios.

Experienced VRSG users and novices alike can work in a flexible manner with the tools and content libraries to increase the realism of terrain easily with rich culture and scripted movements of vehicles and characters. The 3D terrain that you work with in Scenario Editor is the same 3D terrain you visualize in VRSG, and the scenarios you create can be run in both Scenario Editor and VRSG. Like VRSG, Scenario Editor requires the 64-bit Windows 10 operating system and supports terrain in VRSG round-earth terrain format.

Refining culture placement in Scenario Editor.

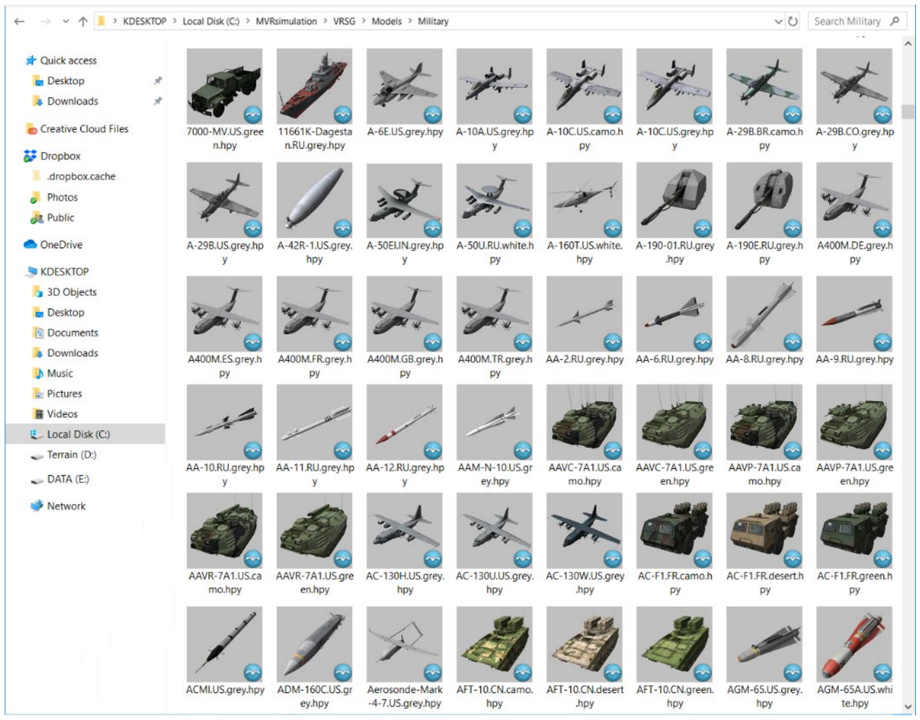
Scripting vehicle waypoints in Scenario Editor.

MVRsimulation 3D model libraries

VRSG is delivered with substantial libraries of 3D content. All models in the libraries are in MVRsimulation's HPY model format, which is described in the chapter, "MVRsimulation 3D Model Format." You can use these models to populate your virtual worlds with cultural features and to interact with networked character and vehicle entities in real time to carry out scenarios. Periodically MVRsimulation adds more models to the libraries; and makes them available to current customers on maintenance at no additional charge. Updates to the model libraries index can be found on the MVRsimulation website at: www.mvrsimulation.com.

Note that HPY models can be used in exactly the same manner as HPX format models from earlier VRSG releases. VRSG and MVRsimulation's Terrain Tools continue to fully support the HPX model format. MVRsimulation's FBX and OpenFlight conversion utilities, which are installed with VRSG, convert models to HPX format.

The model libraries for buildings, commercial and military vehicles, characters, and so on are installed in subdirectories within the \MVRsimulation\VRSG\Models directory. Although the 3D Content section of MVRsimulation's website is ideal for previewing our model libraries, without Internet access you can preview the models by viewing the directories in Windows Explorer in Icon mode.



Previewing MVRsimulation VRSG content in Windows Explorer in Icon mode.

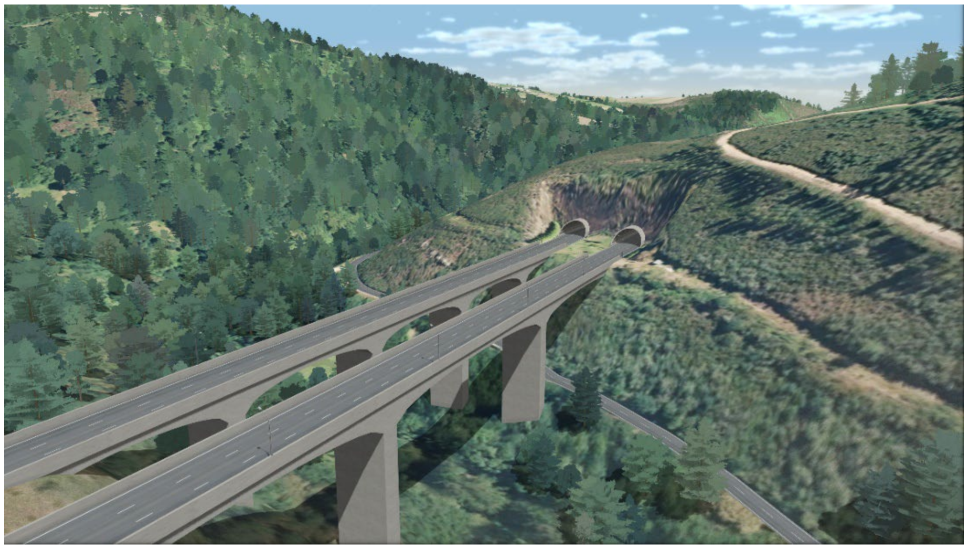
Many models were constructed with modeling tools and then textured with photo textures that were modified in an image editor such as Adobe Photoshop. Other models were converted to MVRsimulation's format from other standard 3D model formats.

Most of the military entity models include articulated parts, damage states, and advanced animations such as wheels or tank tracks that move at a rate corresponding to the vehicle velocity. Most are also ready to support real-time, physics-based thermal sensor viewing within VRSG. Many models have normal-map textures. Some models also have variants. A *variant* is a model that is derived from a basic model but differs in some way, such as an aircraft model that carries a missile or micro-UAV, as in the case of an AH-64 and its variants.

If you need an entity that is not available in VRSG's current model libraries, you can order commercially available 3D models in OpenFlight format and convert them to MVRsimulation's runtime model format. If you need to edit an MVRsimulation model, you can obtain a site license for the model's source data from MVRsimulation. For more information, contact support@mvrsimulation.com.

Model conversion utilities

With VRSG, MVRsimulation provides utilities for converting FBX and OpenFlight-formatted models to MVRsimulation's model format. These conversion utilities, described in the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats," enable you to reuse your existing FBX and OpenFlight entities and static models in VRSG.



VRSG real-time close-up scene of the lush forest and one of the many bridges and twin tunnels along the A6 highway of geospecific virtual Lugo, Spain. The bridge and twin tunnel models were built in a modeling tool, exported in FBX format, and then converted to MVRsimulation's model format.

MVRsimulation and several customers have adopted a workflow to populate 3D terrain with hundreds or thousands of geotypical 3D urban building models, initially generated in ArcGIS® CityEngine®.

VRSG real-time scene of geospecific 3D terrain of Tripoli, Libya. The terrain features 12,456 geotypical building models generated in CityEngine, exported in FBX format, and converted to MVRsimulation's model format.

The models are extruded and textured from CGA (Computer Generated Architecture) rule files that define how the actual building geometry is created and textured using attributes in the OSM data, and exported as a large culture model in the popular FBX file format. Such models can be converted to MVRsimulation's model format with the FBX conversion utility and then rendered in VRSG. A similar workflow exists for FLT formatted models.

What’s new in version 7

The January 2025 release of VRSG version 7 contains the following new features, enhancements, and 3D content:

- Ability to control model articulation by encoding animations in a model’s JSON metadata file that coordinates movement of multiple articulated parts.
- Improved atmospheric model featuring improved light, haze, and cloud interactions. The underlying atmospheric model consists of 16 distinct layers each with unique visibility ranges and wavelength-dependent absorption and scattering properties.
- Ability to teleport to a location specified by a Google Maps URL.
- The H264 plugin has two new options: an option to select CAF DMO Transmitter PDU compliance level, and an option for "leap second" offset in generated timestamps in the KLV stream. The H264 plugin will now return the encoded KLV metadata via the plugin API, enabling a subsequent executing plugin the ability to access the encoded KLV bitstream.
- VRSG's CIGI support has three new component controls: CIGI_SymbolSurfaceEntityAttached which allows a symbol surface to be attached to a DIS entity, CIGI_ComponentIdDepthOfField to control the depth-of-field sensor effect feature, and CIGI_ComponentIdOwnshipForViewport to allow each viewport to have a unique ownship entity.
- MVRsimulation’s 3D model library has surpassed 10,000 models. All models are included in the VRSG installer. You can also download new and updated models from downloads.mvrsimulation.com.
- MVRsimulation's video player has a new plugin API that allows a user-developed DLL to process decoded KLV metadata and draw graphics over the decoded video frames. The video player also has a new command-line option to change where settings are saved in the registry. This enables multiple shortcuts to remember different saved settings.
- Ability to control the degree of recoil effect for a gun model in response to a FirePDU via a JSON variable.
- Ability to specify textures to replace with VRSG 2D water during OpenFlight conversion.
- Ability to assign a wind-sensitivity metric to a texture used in a vegetation model to control how much the model will respond to wind direction and velocity.
- Ability to control the diameter of the real-world pass-through of the Enable Skirt option on the Dashboard’s Varjo tab. Enable Skirt renders a circular pass-through directly below the viewpoint of the headset, enabling the wearer to look down and always see the real-world for spatial awareness.
- Binocular Zoom Mode added to the Varjo viewpoint. The trainee can control three levels of magnification including a simulated M22 reticle.
- Ability for a role player in VRSG, wearing a head-mounted display (HMD), to be attached to a dynamic 3D model. The viewpoint follows an animation and aligns to the assigned position within the 3D model.
- An updated EME plugin for the SOFLAM, which enables users to dynamically change the FOV to match what they see in the real world.
- Integration with the Special Warfare Assault Kit (SWAK) supporting DACAS. VRSG provides the simulated ROVER feed with KLV metadata that correlates and embeds the video stream into the SWAK terrain imagery.
- Simulated NVG improvements for enhancing realism.

- Scalable plugin support for changes in calibration files produced by Scalable version 8.
- Legacy water surface can now optionally be positioned at the nominal regional geoid height instead of at the WGS84 ellipsoid height.
- Ability to use JSON model metadata to remap a model’s articulated part codes.
- Improvements to CIGI symbol rendering: multi-line text now supported and improved mitering of line edges.
- Add support for covert light lobes, visible only in NVG mode.
- New orthographic rendering mode for visualizing models in ModelViewer.
- ModelViewer now supports SGI RGB image capture format.
- CIGI screen capture requests can now request image sizes larger than the framebuffer.
- Support for terrain microtextures using normal maps.
- Instanced models may now support normal mapping textures.
- New CIGI component control that allows the user to select the sun elevation angle where light lobes are triggered to the on state.

VRSG real-time scene featuring a Khordad 15 model, one of the many new models available in VRSG 7.

New 3D terrain datasets built since the previous VRSG release include:

- Holloman Air Force Base (KHMN), New Mexico and Mayport Naval Station, Florida terrains include modeled runways complete with lighting, navigational signs, taxiways, windsocks, and tarmacs with geotypical hangars and control tower.
- Ishigaki and Miyako Islands, Okinawa Prefecture, Japan with over 70 geospecific building models. The database includes hundreds of detail models of boats, containers, vegetation, and poles.
- Washington, DC Metro Area with a total of 133,644 Cyber City 3D buildings along with geospecific artist rendered models of the Capitol Building, White House and Lincoln Memorial.
- Tampa Bay, Downtown Tampa Bay with 3D models of the city.
- MVRsimulation’s CONUS terrain has been entirely rebuilt using National Agriculture Imagery Program (NAIP) source imagery from 2014-2018. (Does not include Alaska and Hawaii.)

- Latvia, with 25 cm per-pixel imagery resolution and five cultural areas of interest (AOIs): Keguma, Riga, Segulda, Senite, and Zilupe.
- Tripoli, Libya, built using 60 cm resolution imagery and 12,456 building models generated in CityEngine.
- Port city of Aden, Yemen, built with a 50cm orthoimagery mosaic, several thousand geotypical CityEngine building models, 2D model of the runway at Aden International Airport, and bridges connecting mainland Aden to Little Aden,
- Okinawa, Mount Fuji, and Tokyo, Japan, with 30-60cm imagery.
- A region in Abu Dhabi, UAE, that includes Dalma Island, Sir Baniyas Island, Higher Yasat, Lower Yasat, Al Ruways Industrial City, Al Dhannah City, Al Hamra, Shuweihat Island, and Barakah.

Previous release

- New features in the Varjo HMD plugin (controlled in the Dashboard's Varjo tab) include using Varjo eye-tracking capabilities for visualizing in VRSG, for real-time monitoring (on a separate VRSG channel in stealth mode) and in training mission playback for after action review. Such features include Enable Gaze Tracking which turns on tracking and pupil calibration, and Eye-Track Action Review, which inserts the headset wearer's head position and pupil gaze data into the DIS stream. Other features include foveated rendering options for Varjo headset and the ability to render the VRSG scene in an external display in addition to within the Varjo headset.
- The Eye Track Visualization option on the More Graphics Options dialog box turns on the rendering of the eye-tracking data (head position and pupil gaze) from a Varjo headset that has been captured in the DIS stream in a recorded VRSG session, as described above. Head position is depicted with a head model and pupil gaze direction is depicted with red and blue cones.
- New features in sensor simulation include:
  - Built-in sensor fusion, enabling you to select two new possible sensor mode options: Visual Electro-Optic fused with IR White Hot and Visual Electro-Optic fused with IR Black Hot. You can control the fusion blend ratio by CIGI component controls or the slider on the Sensor tab of the VRSG Dashboard.
  - Use of a JSON file for creating a radiance profile of materials for sensor simulation. This means you can now generate your own radiance profile either from scratch or with IRSetup from notional data or another sensor model, and edit the file in Notepad. The IRSetup (under ITAR control) now produces an editable .json file.
- A new name convention for cultural feature files, viewpoint files, PDUlogs, sensor profiles, and microtextures: vrsg.clt, vrsg.viewpoint, vrsg.pdulog, vrsg,irm, vrsg.json, and vrsg_microtexture_ x_y_z.tex. Previously created files with the older convention of "metadesic" in the filename are respected by VRSG 7.
- Ability to improve the appearance of dynamic cast shadows when viewing the scene through a narrow field-of-view (FOV), such as those typically used for UAV sensors. A narrow FOV induces high magnification and greater standoff distances, which can cause shadows to become washed out or to disappear entirely. Use the Shadow Quality checkbox on the Shadows tab to direct VRSG to render dynamic shadows at a higher resolution, for FOV angles below a given threshold. (You can find the FOV in use on the Dashboard's Graphics tab.)

- New support for multiple sizes of pageable textures: 512 x 512, 1024 x 1024, and 2048 x 2048 in 3D terrain built with newly released Terrain Tools v2.0. Terrain databases converted from OpenFlight source also benefit from multiple sizes of pageable textures with MVRsimulation's updated OpenFlight to VRSG terrain format conversion utility.
- The Remote Regeneration options Generator/Receiver on the Attach Offsets dialog box tab configure a given VRSG machine in UAV mode to transmit telemetry via DIS packets to a remote VRSG client machine.
- Ability to attach a light lobe to a dynamic moving model via a JSON file.
- The 2D sensor overlay AN/DAS-1 has been updated to AN/DAS-4.
- New CIGI component control, CIGI_ComponentIdRotorWashParams, controls particle dispersion of a rotor wash effect: AGL height at which the effect begins, particle speed, and particle speed variance.
- A new SetPDU interface ATTRV_LOAD_MODEL_MAP to supply the directory of a specific ModelMap.ini to load.
- New CurvedDisplay plugin provides horizontal distortion correction for curved monitors.
- New PixelShift plugin supports projectors that require a pixel shift for emulated 4K resolution.
- VRSG has been recompiled in VS 2019. User-developed plugins built with VS 2015 and 2017 are compatible with VRSG v7.0.

Software maintenance updates

A purchase of a new VRSG license is delivered with software maintenance for one full year. Maintenance includes technical support and VRSG software upgrades within the one-year period. New and updated models and terrain are available to customers on active software maintenance. If you are a current VRSG customer, you can purchase a year of software maintenance directly from the order form on the MVRsimulation website, where complete pricing and order information is available.

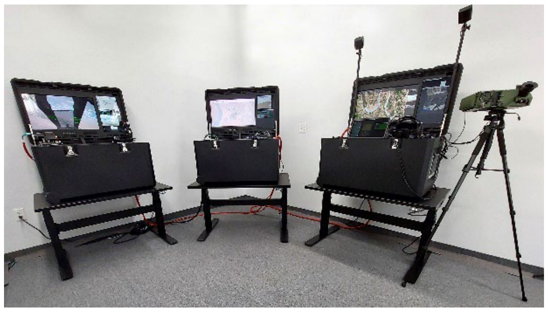
Other MVRsimulation products

In addition to VRSG, MVRsimulation delivers the following products that provide simulation hardware and tools to generate and render 3D geospecific virtual worlds. For more information about other MVRsimulation products, visit www.mvrsimulation.com.

Deployable Joint Fires Trainer

MVRsimulation’s deployable joint fires training solution, the Deployable Joint Fires Trainer (DJFT) provides a quick deploy capability for JTACs and forward observers to train alongside fixed- and rotary-wing aircrew within a fully immersive, joint training environment.

At the DJFT Observer station with the Varjo XR-3 mixed-reality headset and emulated SOFLAM.


The DJFT with VRSG and BSI MACE includes the Role Player, Instructor, and Observer stations.

The modular plug-and-play system, designed by a former JTAC is comprised of three or more stations fully contained within two-person portable ruggedized cases. The DJFT contains all the hardware required to run dynamic, full-spectrum JTAC/joint fires training scenarios, including tactile IZLID and LTD, simulated GPS receiver, mixed-reality HMD system, and communication systems. Scenarios are run on VRSG and BSI's MACE.

The DJFT is fully accredited by the US Joint Fire Support Executive Steering Committee (JFS ESC) for Type, 1, 2, and 3 Terminal Attack Control (TAC), Bomb on Coordinate (BOC), Fixed-Wing (FW), Rotary-Wing (RW), Remote Observer (RO), Video Down-Link (VDL), Suppression of Enemy Air Defenses (SEAD), Urban, Forward Air Controller (Airborne) (FAC (A)), Night, IR, and Laser controls.

Portable Joint Fires Trainer

The Portable Joint Fires Trainer (PJFT) brings mixed-reality Joint Fires simulation training to the tactical edge. Users train at the point of need using real-world Android Team Awareness Kit (ATAK) end user devices, MVRsimulation's Virtual Reality Scene Generator (VRSG), Battlespace Simulations' MACE, and the Varjo mixed-reality headset.

A complete system consists of two portable training stations: one Instructor Operator Station (IOS) and one JTAC Backpack (OBS), each fully contained in a commercial airline carry-on backpack. Systems can be configured with multiple IOS or JTAC backpacks to match specific training goals. Additional stations can be added to expand the training capabilities such as tacitle EMEs, Role Player, Sand Table, and FPV UAV Drone stations.

Two backpack version of the ultra-portable PJFT.

JTAC student and instructor using the PJFT.

The PJFT provides full-spectrum mission training as a stand-alone capability or as a portable component of the fully-accredited Deployable Joint Fires Trainer (DJFT) or similar JFS ESC accredited simulators. The PJFT is directly interoperable with the current U.S. Air Force

JTAC training program of record system, the Joint Terminal Control Training and Rehearsal System (JTC TRS).

First Person View UAV Drone Simulator

The First Person View (FPV) UAV Simulator provides a highly-realistic training solution for the operation of racing-style quadcopter attack drones on the contested battlefield or reconnaissance drones with controllable camera. The internally developed simulator combines VRSG with a high-fidelity flight model from Bihrle Applied Research, to replicate the tactile, visual and cognitive demands of operating agile UAVs in combat to successfully defeat enemy targets.

The FPV UAV can be used as an ultra-low footprint stand-alone training device for tactical operations or networked with other in-use air and ground simulators that operate on the VSRG infrastructure, enabling Large Scale Combat Operations (LSCO) training. The FPV UAV can integrate with GOTS and commercial semi-automated forces (SAF) software to see all entities in the simulated environment.

The FPV UAV drone simulator being used with FPV video goggles.

VRSG streams full motion video (FMV) including KLV metadata to stimulate tactical communication systems. The high-fidelity model is hosted in Bihrle's DSix simulation environment. The simulation employs a modular physics-based blade-element framework that has been used for full-scale rotorcraft training applications in Full Flight Simulators (FFS) and Flight Training Devices (FTD).

Fixed-Wing Part Task Mission Trainer

MVRsimulation's new portable fixed-wing Part Task Mission Trainer (PTMT), designed and built under an internal development program, provides a low-cost, quick-deploy cockpit training solution to fill the gap in current in-use mission tactics training toolkits for military fixed-wing pilots. The system aims to maximize suspension of disbelief for trainee pilots as they practice mission tactics and coordination as part of joint training operations in networked environments. It can also operate as a standalone training solution.

Flying in the PTMT, with VRSG rendering the out-the-window (OTW) view in the Varjo XR-3 mixed reality headset and on the curved display.

The PTMT with VRSG's OTW view on the curved display and simulated sensor view on the cockpit control panel.

Made in the USA with a welded aluminum enclosure, the PTMT uses operational representative aircraft hardware to conduct air-to-air or air-to-ground training scenarios. The system can be configured for training for 3rd and 4th generation combat aircraft currently used by NATO nations by easily changing the position of the specially-designed, patented, flight control stick between side-stick and center-stick positions.

Scenarios are run on VRSG and BSI's MACE. VRSG provides the real-time 3D out-the-window and sensor views. BSI's full suite of tools enables a multi-mission virtual role playing in the air-to-air arena, to include tactical displays that are integrated with HOTAS controls and emulate real world tactical systems. This coupling of MACE with VRSG provides the degree of immersion ideally suited to training from solo part-task mission objectives to large-scale, distributed live-virtual-constructive (LVC) rehearsal of major combat operations.

Terrain Tools extension to ArcGIS Pro

MVRsimulation Terrain Tools enables you to turn geospatial data into real-time 3D terrain from within your GIS software. Building on the industry standard ArcGIS® Pro platform, the Terrain Tools extension combines powerful 3D terrain creation with an accessible interface that can be easily understood by anyone with a comprehension of geospatial data concepts and experience with ArcGIS Pro.

Create real-time terrain in round-earth VRSG terrain architecture for rendering in VRSG with these key features that integrate seamlessly with ArcGIS Pro:

- Live compositing display of raster imagery and elevation data in a WYSIWYG interface.
- Support for any format of source data supported by ArcGIS Pro.
- Ability to supply vector data to define linear and areal features. Generate road networks, fine tune elevation with polygon or point data, create extruded buildings, walls and fences, create large coverage areas of cultural lights, or specify cut-in areas with blended geotypical ground textures.
- Raster display capabilities: pansharpening, custom band order, multiple resampling techniques, histogram stretching, contrast and brightness control, masking, and edge lending.

- Support for building underwater geometry (bathymetry) to increase the terrain fidelity of the ocean floor for use in littoral scenarios with VRSG's 3D ocean sea states.
- Ability to compile buildings and fences from 2D polygon footprint features using an ArcGIS Pro® CityEngine® rule package (.rpk): optimized with geometry LODs, instancing, and spatial clustering.
- Ability to compile 3D and 2D external terrain geometry (such as inset models and runways) directly into the terrain for a seamless integration between the terrain and cut-in model geometry.
- Distributed build system with a browser user interface.

High-resolution sub-inch per-pixel terrain imagery

MVRsimulation's small UAS (SUAS) collects high-resolution still-frame images at sub-inch (2 cm) per-pixel resolution, which after orthorectification, can be used as source imagery to compile 3D terrain with MVRsimulation's Terrain Tools. The resulting geospecific terrain can then be rendered in VRSG.

Real-time VRSG rendering of 2 cm terrain built with imagery collected by MVRsimulation's SUAS of the Naval Air Station Fallon Range Training Complex. On the left: visual view, on the right: the sensor view.

Aerial imagery collected by the SUAS can be ordered directly from MVRsimulation by specifying the area of interest, such as an airfield or a training site. An order of aerial imagery consists of the following deliverables:

- Raw collected photographic imagery data at sub-inch resolution.
- Orthorectified GEOTIFF imagery (processed by MVRsimulation) to be used with your Terrain Tools software license.
- Dataset of terrain tiles in VRSG (round-earth) terrain format at 2 cm resolution to be used with your VRSG software license.
- Aerial imagery orders must be for an area of a minimum of 20 sq km. Access to the area of interest for aerial photography *must* be in accordance with FAA regulations. Customers are responsible for obtaining the necessary authorization and certified access to operate within the particular area of interest for aerial photography. Customers have unlimited unrestricted use rights to the imagery. MVRsimulation retains the right to use the collected data and resulting terrain tiles and to redistribute the terrain (typically as part of a larger set of terrain tiles of a region).

Examples of terrain built from sub-inch resolution imagery can be seen in VRSG-recorded flyover videos, created with the H.264 plugin, which are located on MVRsimulation's website.

Complete systems

MVRsimulation can deliver its software on state-of-the-art personal computers. A complete VRSG system with MVRsimulation software and terrain databases preinstalled, includes a Windows PC with the latest in high-end PC gaming components. These complete systems can be built as a desktop system, notebook computer, or rackmount multi-computer system.

MVRsimulation provides Terrain Network Attached Storage (NAS) systems and Terrain Servers for storing the entire collection of MVRsimulation's 3D terrain. A specific terrain database generation system is available for use with ArcGIS Pro, Terrain Tools and VRSG.

Information about these products can be found on MVRsimulation's website at www.mvrsimulation.com.

About MVRsimulation

MVRsimulation Inc., founded in 1997, is dedicated to providing image generator software that runs on commercial off-the-shelf personal computers. MVRsimulation software is designed around the Microsoft family of products to maximize both the affordability and accessibility of high-speed virtual world creation and visualization software. Since the company's inception, MVRsimulation's visualization software has used the Direct X standard to leverage the rapid advancement of 3D graphics of game technology.

MVRsimulation's goal is to provide affordable simulation solutions to users who need compact, portable, virtual reality software to produce imagery at high throughput rates on state-of-the art personal computers.

For more information about the company and its products, visit the MVRsimulation website at www.mvrsimulation.com.

If you have specific questions about installing or using the VRSG system, or about installing or using an MVRsimulation product, you can contact MVRsimulation support services at:

Email: support@mvrsimulation.com
Phone: 617-739-2667
Fax: 617-249-0151

If you need technical assistance…

MVRsimulation is committed to providing you with a high quality product experience. If you encounter a problem with this product, first try to determine whether the problem stems from an underlying problem with your system:

- Make sure that your machine meets the system requirements listed on the MVRsimulation website at www.mvrsimulation.com/products/vrsg/vrsgsystemrequirements.html.
- Run 3D benchmarking tests to identify any subsystem problems as described in the *MVRsimulation Product Installation Guide*.

Completing these steps might solve the problem and eliminate the need for further assistance.

If you need further assistance, contact MVRsimulation via email at support@mvrsimulation.com with details about the sequence of actions that led to the problem and any resulting error messages. Attach any screenshots that help to illustrate the problem. In addition, supply the following information:

- The product version number.
- The operating system and version number.
- Any special hardware or software configuration that might affect the problem.
- The contents of the VrsgError.txt error log file.

Most email to MVRsimulation support services is answered within 24 hours, Monday-Friday 9:00 am to 5:00 pm US Eastern Time.

Returning a damaged dongle

If you have a damaged VRSG dongle, MVRsimulation will replace it upon request, as described in the Return Merchandise Authorization (RMA) instructions on the website at: www.mvrsimulation.com/howtobuy/returns.html.

To return a damaged dongle for replacement:

1. Notice the dongle ID number located on the sticker that is affixed to the dongle. This dongle ID number contains information about what the dongle is licensed for (product and product maintenance). You must supply this dongle ID number in any communication you have with MVRsimulation regarding your damaged dongle.
2. Email rma@mvrsimulation.com with a request to obtain an RMA number for the return of your damaged dongle.
3. Send to MVRsimulation the damaged dongle or, at a minimum, the metallic unit (which houses the electronic circuit board and memory chips) along with the RMA number. MVRsimulation will *not* replace a dongle for which you only return the purple plastic holder without the corresponding metallic unit.

For more information about MVRsimulation and its products, visit the MVRsimulation website at: www.mvrsimulation.com.

About this manual

This manual describes how to use and customize VRSG.

All terrain database scene images within this manual are real-time screen captures taken by MVRsimulation's VRSG except where noted, and are unedited except to format for inclusion in this manual.

MVRsimulation values any feedback you have on this manual. Email any comments or suggestions to: support@mvrsimulation.com.

CHAPTER 2

Exploring the VRSG System

MVRsimulation offers two ways to get you started with VRSG:

- Run the demo scenarios that are delivered with VRSG. During installation, one or more real-time recorded VRSG demo scenarios are copied to the VRSG installation directory on your computer. For more information, see the section, “Playing VRSG scenarios” later in this chapter.
- Run VRSG with one of the 3D terrain datasets that are installed with VRSG in \MVRsimulation\VRSG\Terrain. You can use these terrains to try out all the options described in this chapter.

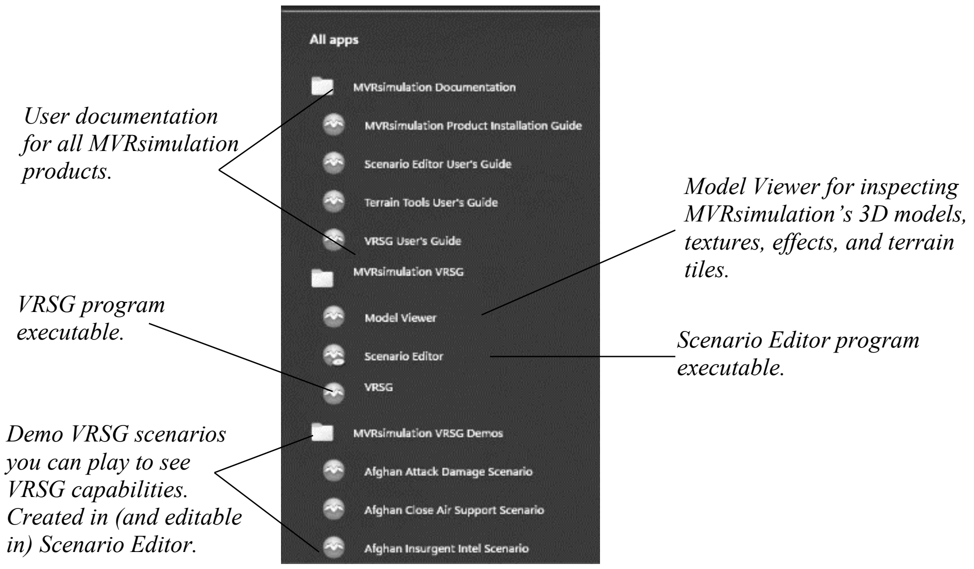
Before you operate VRSG, make sure that:

- The System Properties dialog box for the Windows Device Manager shows that all installed components are functioning properly. Any device that is not installed properly will be shown in the list of devices marked with a yellow exclamation point. You must resolve all problems with devices marked in this manner before you operate VRSG.
- A VRSG license is active on the machine on which you intend to run VRSG. VRSG requires the presence of a hardware dongle plugged into a USB port or software activation on the intended machine.
- Your user account does not need to have administrator or power user privileges. The only user account requirement for running VRSG is permission to write to the VRSG installation directory, that is, the \MVRsimulation\VRSG directory.
- *(Optional)* A 6DOF navigation device is plugged into the computer on which VRSG will be running, if user control of VRSG is required on the machine.

Note: If User Access Control (UAC) is activated on the system that is running VRSG AND the \MVRsimulation\VRSG directory is located under C:\Program Files, then any changes to VRSG files will not be saved or updated in the VRSG installation directory. Instead the updated file will appear in the directory: C:\Users\<username>\AppData\Local\VirtualStore\Program Files\MVRsimulation\VRSG.

Starting VRSG

To start VRSG, choose the VRSG7 executable from the MVRsimulation VRSG7 program folder on the Windows Start menu. The Start menu contains shortcuts to VRSG, VRSG Scenario Editor, VRSG demos, and documentation, as shown in the following example:


User documentation for all MVRsimulation products.
VRSG program executable.
Demo VRSG scenarios you can play to see VRSG capabilities. Created in (and editable in) Scenario Editor.
All apps
MVRsimulation Documentation
MVRsimulation Product Installation Guide
Scenario Editor User's Guide
Terrain Tools User's Guide
VRSG User's Guide
MVRsimulation VRSG
Model Viewer
Scenario Editor
VRSG
MVRsimulation VRSG Demos
Afghan Attack Damage Scenario
Afghan Close Air Support Scenario
Afghan Insurgent Intel Scenario
Model Viewer for inspecting MVRsimulation's 3D models, textures, effects, and terrain tiles.
Scenario Editor program executable.

Alternatively, you can start VRSG by double-clicking the VRSG 7 executable directly in the \MVRsimulation\VRSG\Bin directory.

You can also start VRSG from the command line. For example:

```
Run C:\MVRsimulation\VRSG\Bin\Vrsg7.exe
```

Information about how to automate starting VRSG is presented later in this chapter, in the section, "Running VRSG from the command line."

About enabling certain VRSG options

The status of a VRSG license, its maintenance plan renewal, and access to certain options are all managed by the authentication software associated with the product's license. You update the VRSG license by using an unlock code obtained from MVRsimulation. Some VRSG options or MVRsimulation 3D terrain are not enabled in the core product, but instead require an MVRsimulation-supplied unlock code to gain access to them. For example, the physics-based IR feature requires an unlock code to activate it.

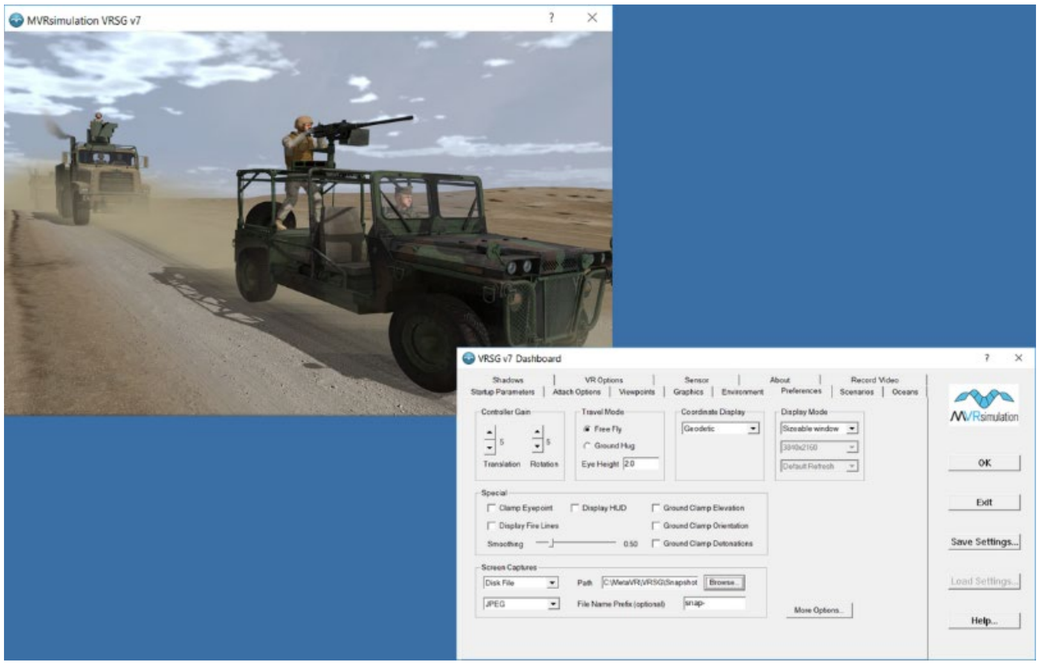
To use VRSG's physics-based IR, you must first enter an MVRsimulation-supplied unlock code to enable this option for your VRSG license. (International customers must obtain ITAR approval for this feature.) For more information about software maintenance renewal and unlock codes, see the *MVRsimulation Product Installation Guide*.

About making a temporary license permanent or renewing product maintenance

VRSG is initially delivered with a temporary license, which has an expiration date. This date is based on the payment terms of the purchase and an additional grace period. A few days before the temporary license is due to expire, a warning message will appear when you start VRSG, reminding you to email MVRsimulation to obtain a permanent license. When your VRSG license maintenance plan nears expiration, a reminder message will also appear. For information about obtaining an unlock code to update your VRSG license in either case, see the *MVRsimulation Product Installation Guide*.

Setting VRSG session options and preferences in the Dashboard

The VRSG user interface consists of the Dashboard where you specify session settings, and a visualization window where VRSG renders the 3D virtual terrain.



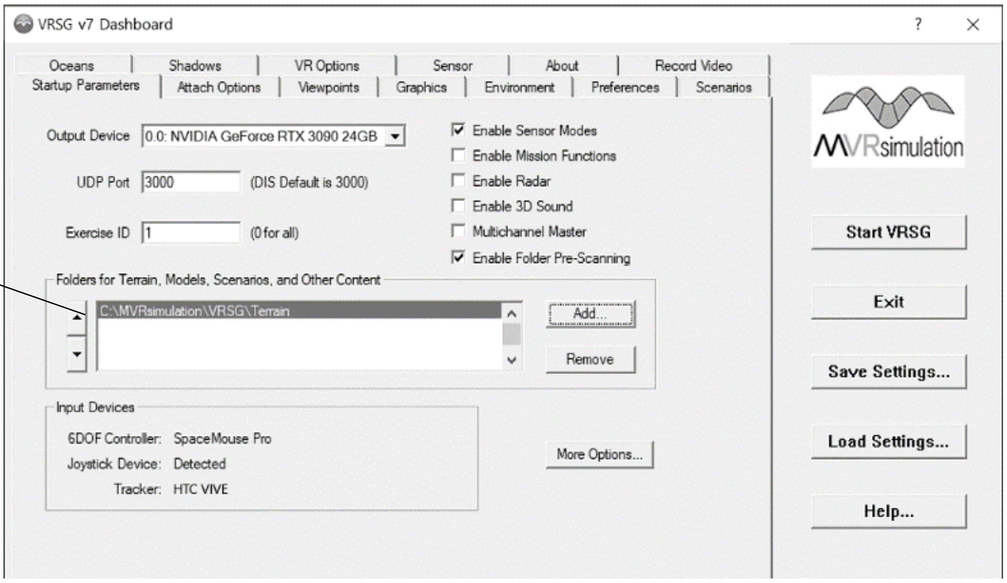
Controller Gain: Translation 5, Rotation 5
Travel Mode: Free Fly [✓], Ground Hug; Eye Height 2.0
Coordinate Display: Geodetic
Display Mode: Sizeable window; 3840x2160; Default Refresh
Special: Clamp Eyepoint, Display HUD, Ground Clamp Elevation, Display Fire Lines, Ground Clamp Orientation, Smoothing 0.50, Ground Clamp Detonations
Screen Captures: Disk File; JPEG; Path C:\MetaVR\VRSG\Snapshot; File Name Prefix (optional) snap-

The Dashboard shown here in front of the VRSG visualization window.

Each tab in the VRSG Dashboard contains various settings and preferences for the virtual world session, such as network options, display parameters, paths to the 3D terrain, models, and scenarios to render, environment conditions, entities to attach to, and more.

The following example shows the VRSG Dashboard:

Initially set to visualize MVRsimulation's Hajin, Syria, terrain, which is installed with VRSG.



Output Device: 0.0: NVIDIA GeForce RTX 3090 24GB
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

☑ Enable Sensor Modes
☐ Enable Mission Functions
☐ Enable Radar
☐ Enable 3D Sound
☐ Multichannel Master
☑ Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content: C:\MVRsimulation\VRSG\Terrain

Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: HTC VIVE

The action buttons on the right side of the Dashboard apply to all settings.

- Click the Start VRSG button to switch to the visualization window. The label on the button reads "Start VRSG" when you first start a VRSG session. That label changes to read "OK" after you have launched VRSG and 3D terrain (and other content) has been loaded in the visualization window.
- To switch focus between the visualization window and the Dashboard, press the Esc key on the keyboard. If the Dashboard is hidden, this displays it in front of the visualization window. Press Esc again (or click OK) to dismiss it.
- Click the Exit button to end the VRSG session.
- When you exit from VRSG, most Dashboard settings are saved in the file DefaultConfigMds_<machineName>.json, located in the directory \MVRsimulation\VRSG\Settings. You can however save settings specific to a particular configuration in a unique settings (*.json) file.
- Click the Save Settings button to save the VRSG startup parameters to a unique settings (*.json) file that you define. If you load a variety of different terrain regularly in VRSG, saving settings is a quick way to save the paths to different directories that contain terrain, models, and scenarios, as well as environmental settings like fog color, cloud coverage, and so on. That way you can easily restore those paths and settings at a later time by loading the .json file.

- Click the Load Settings button to load the settings that are stored in a JSON settings file (.json). By default, VRSG starts with the file DefaultConfigMds_<machineName>.json located in the \MVRsimulation\VRSG\Settings subdirectory. You can specify a different settings file to load that contains the startup parameters that you previously saved separately from a particular session. Upon startup, if no settings file is specified VRSG loads whatever was loaded for the previous VRSG session.
- Click the Help button to display the Dashboard's online Help, which provides information about the fields and controls in each tab.

Navigating the virtual world display

You can navigate in VRSG's visualization window and attach to entities in a virtual world scenario by using the keyboard and a game controller.

Using the keyboard, you navigate by pressing the arrow keys and key sequences that include the arrow keys and the Shift key.

For unconstrained database navigation, a 6DOF device is preferred. Although you do not have to use a 6 DOF controller with VRSG, most users prefer to use one, because it provides optimal 360-degree navigation in any direction, thus a more intuitive virtual world experience. For this reason, most complete VRSG systems are sold with a 3dConnexion 6 DOF game controller.

As a secondary device, you can use a gamepad or joystick for fixed-wing mode, manipulating a First Person Simulator character, or controlling a simulated UAV camera view.

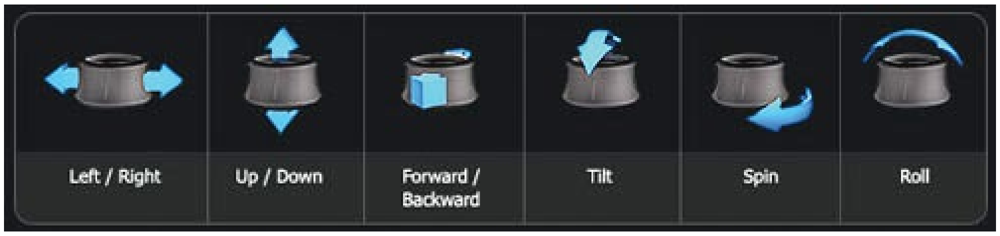
Using a 6DOF controller

VRSG functions map to buttons located on supported 6DOF controllers. You can adjust the sensitivity of the controller on the Preference tab of the Dashboard. In addition, you can slow down the gain dynamically as you move through the virtual world by pressing the Y key on the keyboard to turn on Nudge mode. Press Y again to turn it off.

VRSG supports the Logitech 3Dconnexion SpaceMouse Pro, and SpaceMouse Compact 6DOF controllers.

SpaceMouse Pro and SpaceMouse Compact 6DOF controllers.

These controllers use a knob called a *controller cap* which provides the 3D navigation by flexing in all directions. When you use the device with VRSG, moving the cap left or right moves in those directions, pulling and pushing moves up and down, pushing the cap away from you moves forward, pulling it toward you moves backward, and rotating or tilting the cap rotates the view about the respective axis.

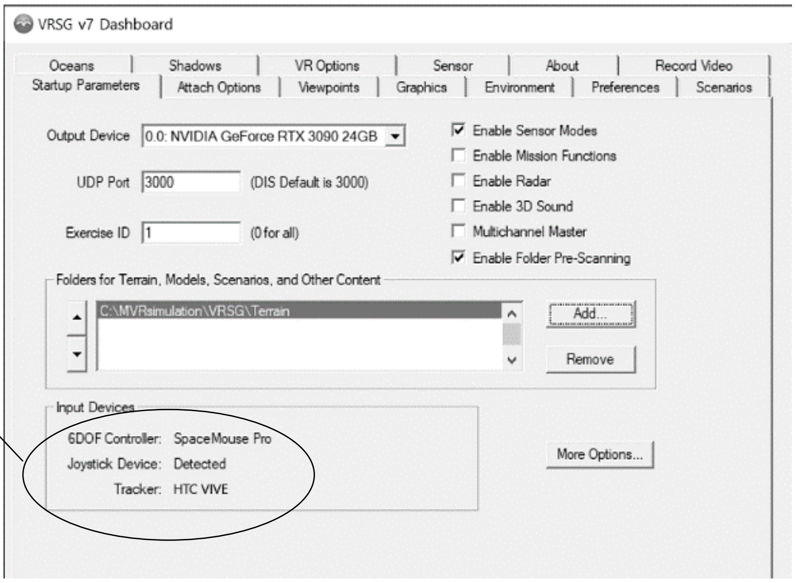


Buttons 1, 2, T, L, R, and F on the SpaceMouse Pro correspond to VRSG functions in the virtual world. For example, by pressing the R button on the SpaceMouse Pro you can cycle through the attachment mode options, which correspond to the Mode options listed in the Attach tab of the VRSG Dashboard.

If you are short on work space, you can also use the SpaceMouse Compact, which is a small 6 DOF controller without a wrist rest and only two buttons. You can access VRSG functions (mapped to buttons on the two controllers) with keyboard shortcuts and on the Dashboard.

When you first start VRSG, the status of detecting an attached controller is displayed on the Startup Parameters tab as shown:

Indicates VRSG successfully found the attached controller (in this case a SpaceMouse Pro) and gamepad. In this example, VRSG has also detected the presence of an HTC VIVE device.



Output Device 0.0: NVIDIA GeForce RTX 3090 24GB

UDP Port 3000 (DIS Default is 3000)

Exercise ID 1 (0 for all)

[✓] Enable Sensor Modes
[ ] Enable Mission Functions
[ ] Enable Radar
[ ] Enable 3D Sound
[ ] Multichannel Master
[✓] Enable Folder Pre-Scanning

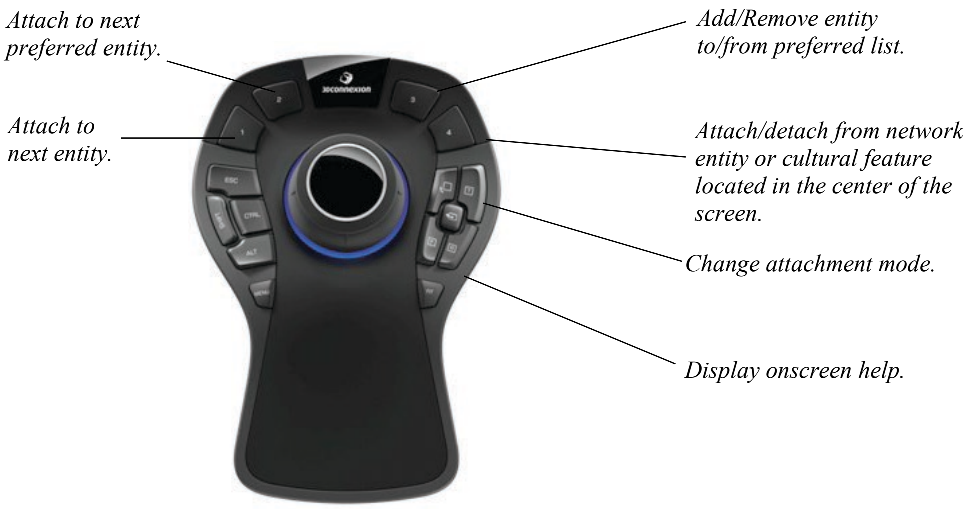
Folders for Terrain, Models, Scenarios, and Other Content

C:\MVRsimulation\VRSG\Terrain

Input Devices

6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: HTC VIVE

The following example describes the button mapping of a 6DOF controller to VRSG features:


Attach to next preferred entity.
Attach to next entity.
Add/Remove entity to/from preferred list.
Attach/detach from network entity or cultural feature located in the center of the screen.
Change attachment mode.
Display onscreen help.

3Dconnexion SpaceMouse Pro.

You can adjust the sensitivity of the controller on the Dashboard's Preferences tab, as previously described. In addition, you can slow down the gain dynamically as you move through the virtual world, by pressing the Y key on the keyboard to turn on Nudge mode. Press Y again to turn it off.

Using a gamepad or joystick

The Logitech F310 gamepad is MVRsimulation's joystick-type device of choice.

Logitech F310 gamepad.

Gamepads and joysticks are used in VRSG in the following cases:

- Flying the terrain database in fixed-wing mode. Press F11 on your keyboard to activate fixed wing mode; press F11 again to deactivate it. Fixed wing mode flies a straight path. You can change direction and orientation of eyepoint with the left thumbstick on the gamepad. The right thumbstick acts as a throttle, which you move in the right-left direction to adjust the speed.
- Controlling a simulated UAV camera in UAV View mode, which enables you to attach to a DIS entity that supplies the UAV airframe telemetry. In this mode, VRSG uses an

internal sensor payload model, which requires an attached gamepad to pan and zoom the camera view. For more information about using a gamepad with UAV mode, see the chapter "Configuring VRSG for Simulating UAVs."

- Creating a dismounted character to perform activities on the database in VRSG First Person Simulator (FPS); press F9 on your keyboard to instantiate an FPS character. For ease of use in manipulating a character in FPS, MVRsimulation strongly recommends the Logitech F310 gamepad. With two analog thumbsticks, digital buttons, precise D-pad, and smooth, precise action, this gamepad is the ideal input device for manipulating the character's movements and various functions such as firing weapons, laser ranging or designating in JTAC/CAS mode, and so on. For more information about using a gamepad with FPS and with UAV simulation, see the chapters "Using 3D Characters in VRSG" and "Configuring VRSG for Simulating UAVs."

Specifying terrain and other startup parameters

On the Startup Parameters tab, you specify information about the VRSG session, such as the 3D terrain and other content to load, network settings, DIS protocols, and whether to run VRSG in physics-based sensor mode. These settings can be used for a single session or can be saved to a settings file to use again in subsequent sessions, as mentioned earlier.

VRSG can render any 3D terrain that is in MVRsimulation's VRSG round-earth terrain format (MDS). For example, after you install VRSG, you can immediately load one of the demo terrain datasets that were also installed. By default, the Hajin, Syria, terrain will load when you first start VRSG after installation unless you specify other terrain to load.

Other 3D terrains you can use include:

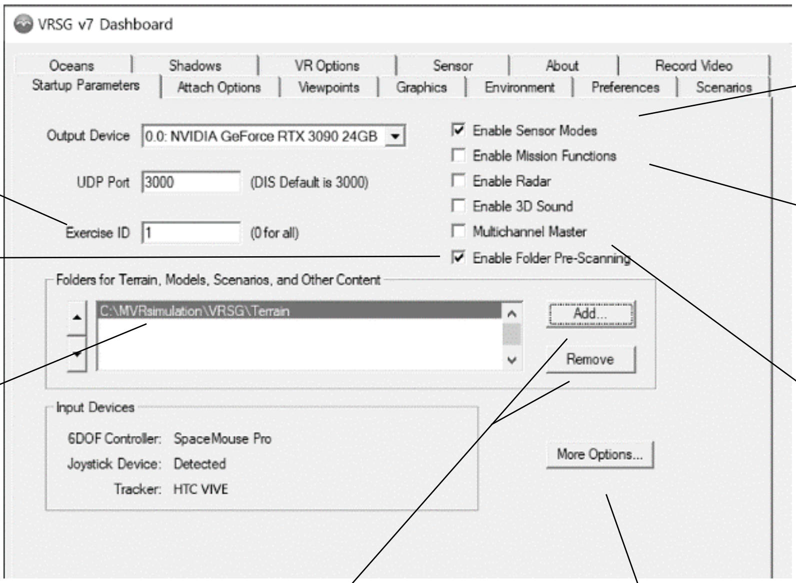
- Terrain drives you received from MVRsimulation containing 3D terrain of CONUS++ or other region of the world.
- Terrain tiles created with MVRsimulation's Terrain Tools.
- OpenFlight terrain databases you convert to MDS terrain tile format using MVRsimulation's OpenFlight conversion utilities that are delivered with VRSG. (See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats" for more information.)
- Databases obtained from a third party, such as a government agency or contractor.

When you start VRSG, the Dashboard appears with the Startup Parameters tab displayed:

The number of the exercise active on DIS network. (Use 0 to receive all exercises.)

Instructs VRSG to pre-scan all folders in its search path prior to runtime. Turned on by default.

The search path: a list of all the directories of all content that VRSG should load for a visualization session.



VRSG v7 Dashboard

Output Device: 0.0: NVIDIA GeForce RTX 3090 24GB

UDP Port: 3000 (DIS Default is 3000)

Exercise ID: 1 (0 for all)

- [✓] Enable Sensor Modes
- [ ] Enable Mission Functions
- [ ] Enable Radar
- [ ] Enable 3D Sound
- [ ] Multichannel Master
- [✓] Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content: C:\MVRsimulation\VRSG\Terrain

Input Devices

6DOF Controller: SpaceMouse Pro

Joystick Device: Detected

Tracker: HTC VIVE

Activates VRSG's sensor view modes, and the Sensor tab.

Enables VRSG to listen to CIGI messages for elevation lookup, laser range find, collision detection, and intersection with dynamic models.

Use VRSG as a multi-channel stealth or to test a multichannel configuration without a simulation host model.

Add or remove directories for terrain, models, and any other files you want VRSG to load.

Click to set network options.

- Output Device – The monitor on which VRSG will render the 3D scene. VRSG supports the use of multiple monitors in one system, as described in the section "Using VRSG on multiple monitors" later in this chapter. In the example shown above:
  `0.0: NVIDIA GeForce RTX 3090 24 GB`
  the first number is the video card index, and the second is monitor index. Following the name of the video card is the amount of memory on the card.

- Port – VRSG expects all DIS participants/traffic to be on a single port; default is 3000.

- Exercise ID – The number of the exercise active on DIS network. (Use 0 to receive all exercises.) All participants in an exercise must use the same exercise number in this field in order for entities to be visible on the network. If you are running scenarios exported from VRSG Scenario Editor, the exercise ID (and port number) must match what was set for the scenarios. If you run VRSG with Battlespace Simulations' Modern Air Combat Environment (MACE), be sure the same exercise number is listed in MACE as well.

- Enable Sensor Modes – Activates VRSG's sensor view modes and the Sensor tab, as described later in this chapter, and more fully in the chapter, "Working with Sensor-View Modes and Physics-Based IR."

- Folders for Terrain, Models, Scenarios, and Other Content – VRSG's search path of directories of all the content VRSG should search and load in the visualization session. For more information about loading terrain and content into VRSG see the section "Setting up search paths to terrain and other content" later in this chapter.

- Enables Mission Functions – Activates processing of CIGI mission functions (e.g. HAT/HOT, Line of Sight Requests, Collision Segment Notifications). In a multichannel environment, only one VRSG channel should have this option enabled, otherwise the host will receive mission functions responses from multiple channels. See the appendix "CIGI Version 4.0 Support" for more information.

- Enable Radar – Supports the running of a radar simulation on top of VRSG. See the chapter "VRSG Radar Simulation" for more information.
- Enable 3D Sound – Activates support for sounds associated with events, as described in the chapter "Integrating 3D Sounds."
- Enable Folder Pre-Scanning – Instructs VRSG to pre-scan all directories in its search path prior to runtime and maintain a local map of where files are located. This option, enabled by default, speeds up load times in most configurations in which VRSG is run.

Note: Once you have launched the VRSG visualization window, the settings on the Startup Parameters tab are no longer editable. The tab is displayed in read-only mode. (Settings on most other Dashboard tabs remain editable throughout the VRSG session, except for specifying the RTSP mode and port on the Record Video tab.)

Setting up search paths to terrain and other content

Upon startup, VRSG loads by default the terrain, model, scenario, and texture directories that were installed in its installation path. When you first start VRSG after it is installed, MVRsimulation's Hajin, Syria, terrain will load by default, unless you specify other terrain to load. The set of directory paths of 3D content you instruct VRSG to load is called its *search path*. You can instruct VRSG to load other terrain, models, textures, scenarios, and so on, by specifying a different search path.

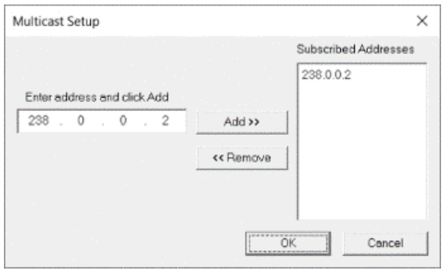
In the Folders for Terrain, Models, Scenarios, and Other Content section of the Startup Parameters tab, list all paths to terrain, models, and any other files (such as cultural feature files, scenarios) you want VRSG to load that were not installed with VRSG. For performance reasons, VRSG's search is not recursive; you must specify each directory you want to be searched, either explicitly in this field or in a text file called VRSGTerrainSearchPath.txt, which is described in the chapter "Loading Content into VRSG."

The order in which directories are listed in the Folders for Terrain, Models, Scenarios, and Other Content section is critical, as VRSG searches the list from top to bottom. For example, if you have a small area of high resolution terrain of an airfield that you want to render along with lower resolution terrain of the larger region that includes the airfield and much more, you would list the directory that contains terrain tiles of the higher resolution first (that is, higher in the search path list). For more information about configuring content for VRSG, see the chapter "Loading Content into VRSG."

Setting additional startup parameters

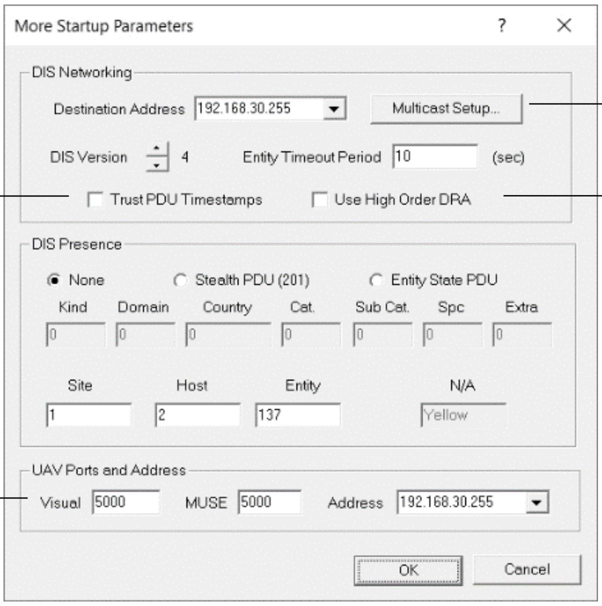
On the Startup Parameters tab, click the More Options button to display a dialog box in which you specify networking settings and other information:

- DIS network interactions. For entities to appear in a network exercise, each VRSG participant must have unique Site Host and Entity values set in this dialog box, and a common Exercise ID specified on the Startup Parameters tab.
- Set up multiple multicast addresses if your simulation requires it. To do so, click Multicast Setup to display the dialog box in which, for each multicast group, you enter the address to subscribe to, and then click Add.



Multicast Setup dialog box.

- Client machines to set on the UAV master machine for UAV simulation, as described in the chapter "Configuring VRSG for Simulating UAVs."



Destination Address: 192.168.30.255
DIS Version: 4
Entity Timeout Period: 10 (sec)
☐ Trust PDU Timestamps
☐ Use High Order DRA
DIS Presence: ● None ○ Stealth PDU (201) ○ Entity State PDU
Kind 0, Domain 0, Country 0, Cat. 0, Sub Cat. 0, Spc 0, Extra 0
Site 1, Host 2, Entity 137
UAV Ports and Address: Visual 5000, MUSE 5000, Address 192.168.30.255

Click here to display a dialog box in which you can specify the address for each multicast group.

Instructs VRSG to rely on the timestamp in the DIS PDU header provided by the simulator to optimize dead-reckoning. If this option is not selected (the default), VRSG assigns its local clock time to a PDU when the PDU is received.

Support for higher–order dead reckoning algorithms such as those that include acceleration and angle rates.

Settings for UAV simulation.

Setting entity attachment options

On the Attach Options tab you specify which entities to add or attach to your virtual world display, which mode of attachment to use (related to the viewpoint), and which entities to detach from or remove from the session. The All Entities list is populated with the entities mapped in the ModelMap.ini file which are on the network at a given moment, which is described in the chapter "Configuring Models and Events."

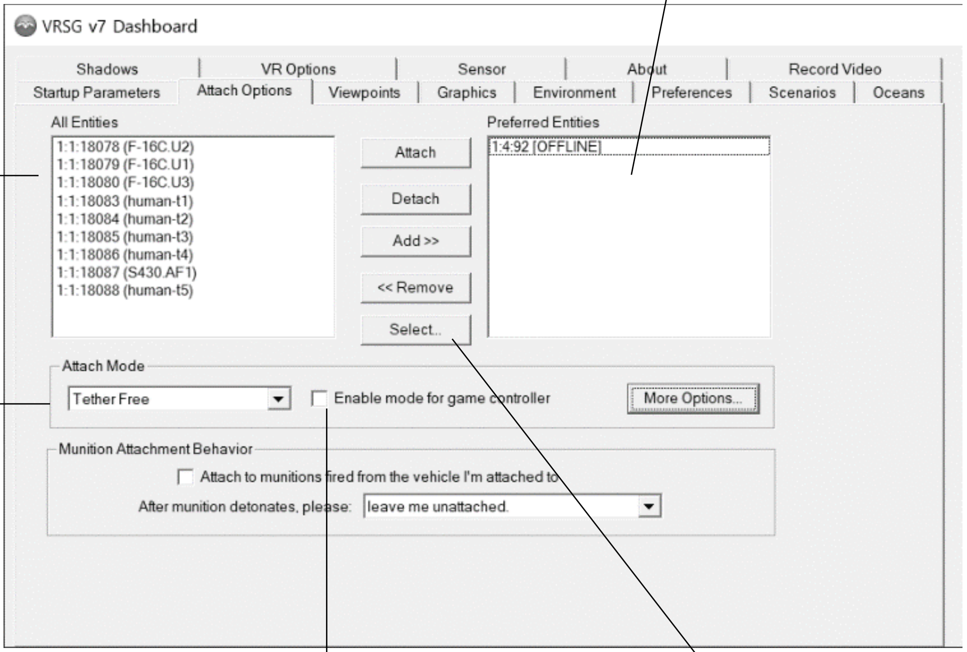
Not only can you attach to a dynamic entity; you can attach to a static model too. After attaching to a static model you can move it to refine the model's position on the terrain. (For information about adding dynamic and static models directly to a VRSG scene, see the chapter "Configuring Models and Events.")

You can select multiple entities at once for adding or removing them from the preferred list. Press and hold the Shift key to select contiguous entities in the list; press and hold the Control key to select noncontiguous entities. Double-click on an entry from either list for a quick attachment.

A subset of the entity list; shows the entities that you frequently attach to.

The entities VRSG knows about, which you can attach to during the visualization session.

Current mode of attachment to an entity or cultural feature.

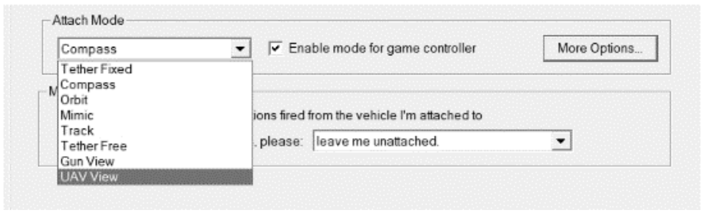


Makes the selected attachment mode available to select with a control on the 6DOF controller.

Click to attach to a particular entity (or cultural feature) by its site/host/ID or by marking.

Select the Enable Mode for Game Controller checkbox to attach to and detach from entities and cycle through the attachment modes using a 6DOF game controller. All features available through the game controller are described later in this chapter in the section "Navigating the virtual world display."

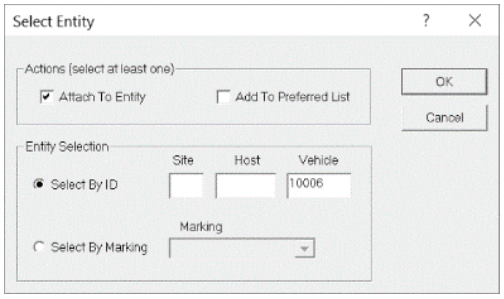
The Attach modes are for attaching to an entity:



Attaching to an entity

To attach to an entity, select the entity and click Attach (or press the appropriate button on your game controller). You can also attach to an entity by simply positioning the cursor directly on the entity and pressing the middle mouse button/wheel. When you do so, you become attached to the entity in the attachment mode selected in the tab. To detach from the entity, press the D key on the keyboard.

Click the Select button on the Attach Options tab to attach to a particular entity or cultural feature by its site/host/ID or its marking. When the Select dialog box appears, select whether to attach to the entity or cultural feature or add it to your preferred list of entities you will attach to during the session. Then provide the means of identifying it.


Select Entity

Actions (select at least one)

[✓] Attach To Entity

[ ] Add To Preferred List

Entity Selection

(●) Select By ID — Site: ___ Host: ___ Vehicle: 10006

( ) Select By Marking — Marking: ___

When you select an entity by marking, a dropdown list appears with a list of the entities that match the characters you have typed so far, which means you only need to type as many characters as required to get a unique match.

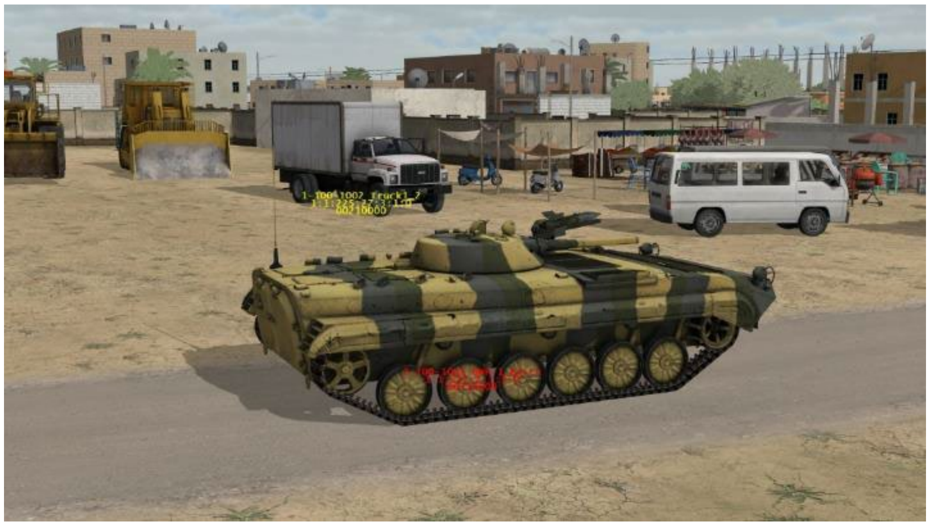
Displaying dynamic entity IDs and culture IDs directly on the scene

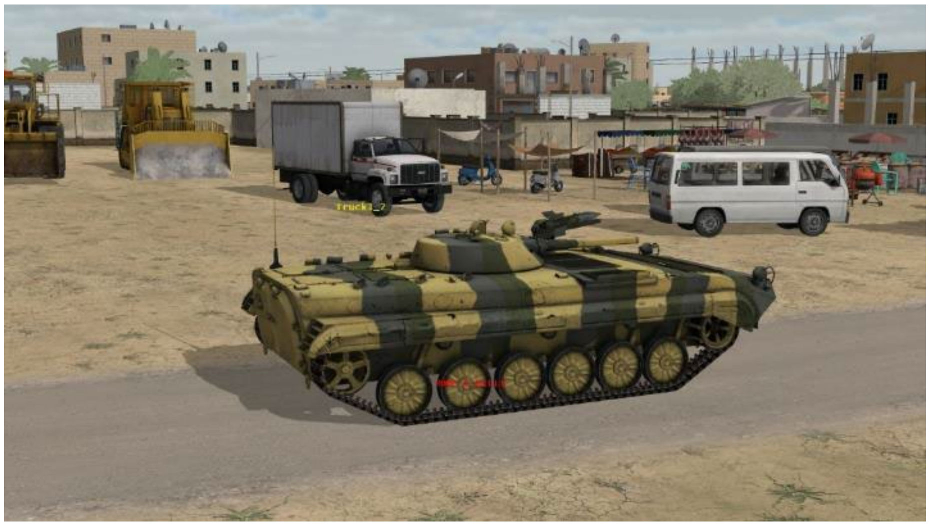
You can display entity ID numbers in the virtual world display by pressing F12 key on the keyboard, as shown on the next page, or by locating the entity's entry in the ModelMap.ini as described in the chapter "Configuring Models and Events." Pressing the F12 key on the keyboard displays the ID information of entities present in the scene in three ways. Each subsequent press of F12 cycles the manner in which the entity ID is displayed. In a multiple viewport setup, this information is displayed in one viewport. (For information about using more than one viewport, see the section "Using multiple viewports" later in this chapter.)

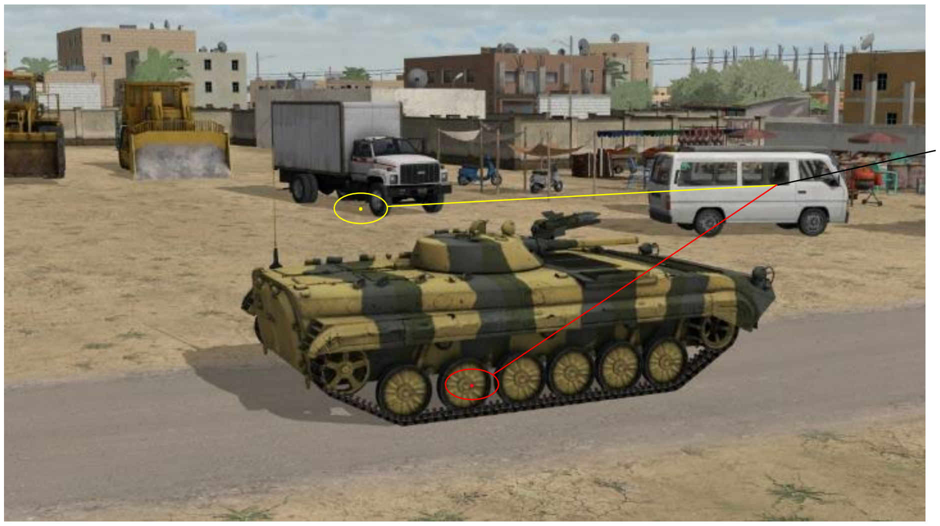
The VRSG screen captures on the following page show the three options for displaying the IDs of the convoy entities visible in the scene.

- The first image shows the complete entity information including site-host-entity, callsign, DIS enumeration, and appearance state.
- The second image shows only the callsign.
- The third image shows a dot (**.**) representation shown at the bottom of the entity. This model representation is useful in situations where there are many entities grouped together in a scene and you want to see the location and movement of the group.

If your entities have been assigned as friendly or opposing forces, the entity ID information will display in the corresponding friendly/opposing color as configured in the More Preferences dialog box described later in this chapter.

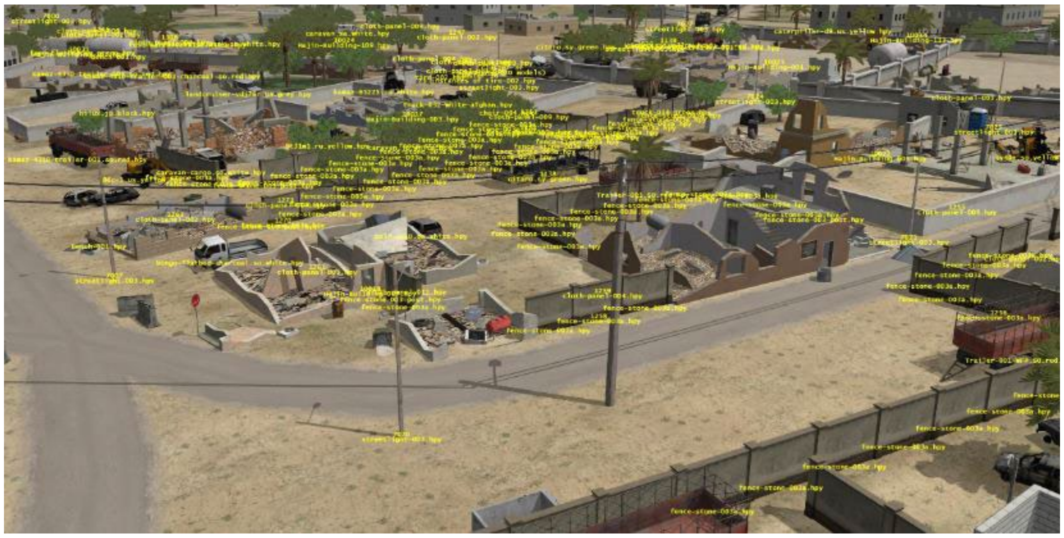






The dot denotes the presence of a dynamic entity.

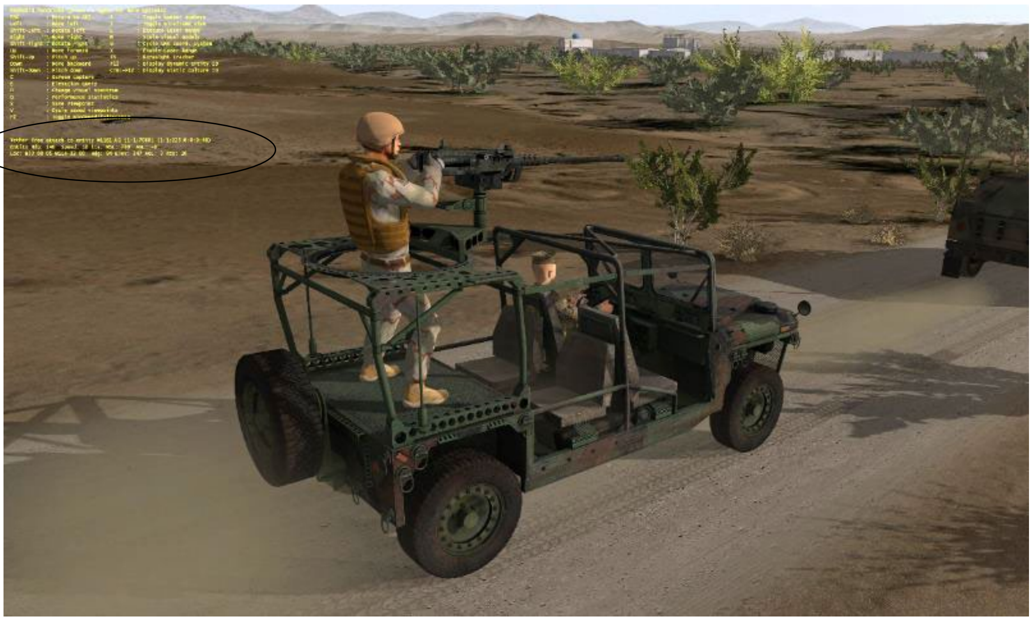
To display information about cultural features in the scene, press Ctrl-F12. The following example shows the resulting names of static models displayed from pressing Ctrl-F12:



Notice the example also shows VRSG's runtime grouping of identical models and neighboring models into clusters and aggregates for performance efficiency.

You can obtain more information about a specific entity by attaching to it (clicking on it with middle mouse button/wheel) and then pressing F1. The entity information is displayed below the onscreen help as shown in this next image:

Information about the attached entity.

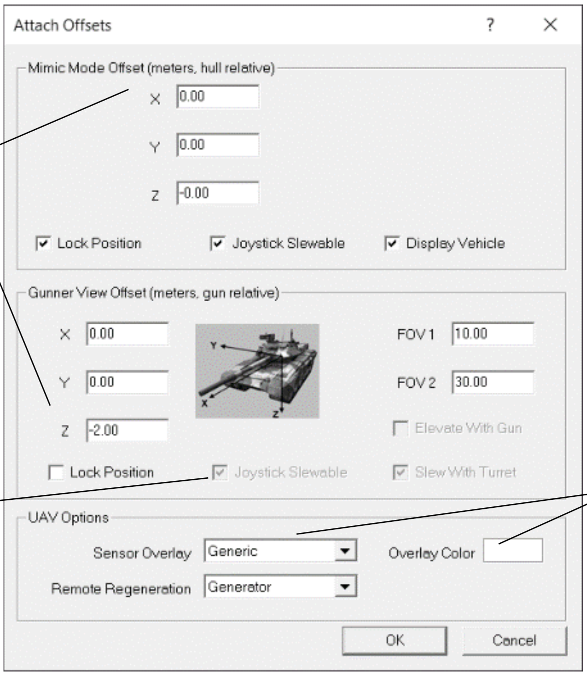


To cease or pause entity propagation, press Ctrl-F. This action ceases dead reckoning, timing out, and processing PDUs. Press Ctrl-F again to restore normal processing. (For information about all of VRSG's keyboard functions, see the appendix "VRSG Keyboard Functions.")

The Attach Mode options attach an entity to the viewpoint in the following ways:

- Tether Fixed – Attaches the viewpoint to the entity at a fixed distance; when the entity changes orientation, the viewpoint also changes orientation.
- Compass – Attaches the viewpoint to the entity at a fixed distance from the entity in a world coordinate system; when the entity changes orientation, the viewpoint's orientation (such as north-south) remains unaffected.
- Orbit – Rotates the viewpoint around the entity, at fixed a distance. Shift-left click to define the point of rotation. Click again (anywhere) to remove the point of rotation.
- Mimic – Attaches the viewpoint to the crew compartment of the entity.
- Track – Attaches the viewpoint at a fixed point with the entity centered in the field of view.
- Tether Free – Attaches the viewpoint at a variable distance (which you can modify with a game controller).
- Gun View – Attaches to the articulated gun of an entity.
- UAV View – Enables you to attach to a DIS entity that supplies the UAV airframe telemetry. In this mode, VRSG uses an internal sensor payload model, which requires an attached gamepad or other joystick device to pan and zoom the simulated camera view.

Click the More Options button to display the Attach Offsets dialog box, where you specify the viewpoint to associate with a particular entity and other options. The Joystick Slewable option, selected by default, is required for using a gamepad with VRSG (in UAV camera, fixed-wing, or FPS mode).


Attach Offsets
?
Mimic Mode Offset (meters, hull relative)
Gunner View Offset (meters, gun relative)
UAV Options
OK
Cancel
Offsets relative to the DIS body coordinate system.
Required for using a gamepad with VRSG (in UAV camera, fixed-wing, or FPS mode).
2D HUD display for the UAV camera view.

Mimic Mode Offset (meters, hull relative)
X 0.00
Y 0.00
Z -0.00
[✓ Lock Position] [✓ Joystick Slewable] [✓ Display Vehicle]

Gunner View Offset (meters, gun relative)
X 0.00
Y 0.00
Z -2.00
FOV 1 10.00
FOV 2 30.00
Elevate With Gun
Lock Position
[✓ Joystick Slewable]
[✓ Slew With Turret]

UAV Options
Sensor Overlay: Generic
Overlay Color
Remote Regeneration: Generator

For the UAV options Sensor Overlay and Overlay Color fields, select the 2D overlay display appropriate for your camera view. VRSG includes built-in 2D “HUD” overlays for many popular UAS systems and targeting pods. Use the Remote Regeneration options Generator/Receiver when a given VRSG machine in UAV mode transmits telemetry packets via DIS packets to a remote VRSG client machine. See the chapter “Configuring VRSG for Simulating UAVs” for more information.

Instead of attaching to an entity, you can fly through the virtual world unconstrained as a *stealth*. A stealth in the context of distributed simulation is a 6 degree-of-freedom (6DOF) massless entity that can fly anywhere in the virtual world and observe the 3D world from vantage points not normally possible when using a conventional ground or winged vehicle.

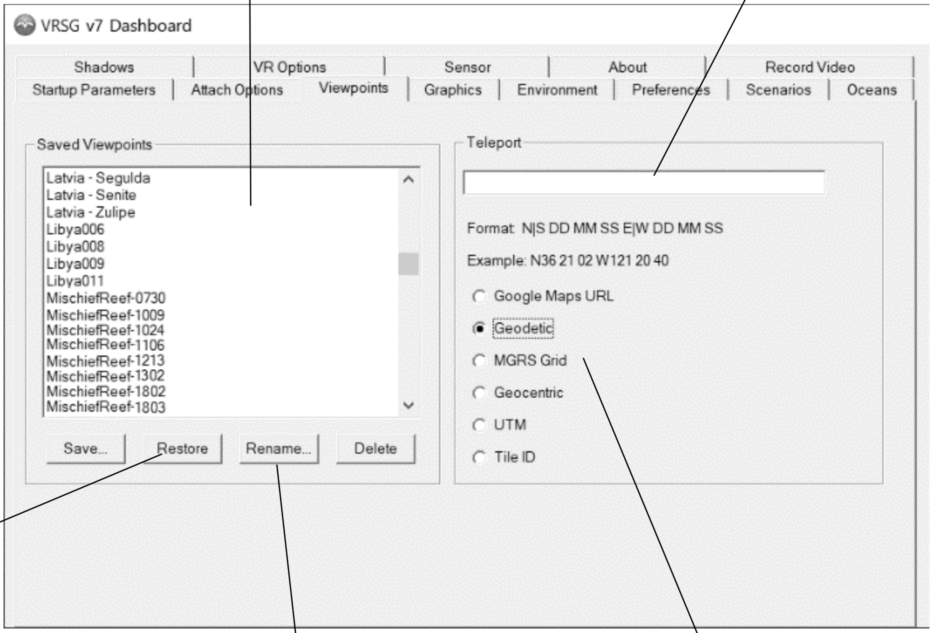
You can also attach to a point on the terrain to orbit around that location. Press Shift and click the middle mouse button/wheel to orbit about the terrain where you clicked.

Setting viewpoints

On the Viewpoints tab of the VRSG Dashboard, you specify the viewpoints to save or return to while visualizing a set of VRSG terrain tiles. A *viewpoint* contains information about a view, or eyepoint: its location, orientation, near clipping and field-of-view settings, the attachment mode in effect, and some environment settings. Saving viewpoints is an easy way to keep track of points of interest on the 3D terrain that you want to return to in a visualization session.

List of saved viewpoints. If you name a viewpoint “default” VRSG will display that viewpoint when it first loads the virtual terrain.

Enter the coordinates of a location you want VRSG to move the eyepoint to in the rendered scene. Use this field to specify a precise location as an alternative to navigating to it with a game controller.



Latvia - Segulda
Latvia - Senite
Latvia - Zulipe
Libya006
Libya008
Libya009
Libya011
MischiefReef-0730
MischiefReef-1009
MischiefReef-1024
MischiefReef-1106
MischiefReef-1213
MischiefReef-1302
MischiefReef-1802
MischiefReef-1803

Format: N|S DD MM SS E|W DD MM SS

Example: N36 21 02 W121 20 40

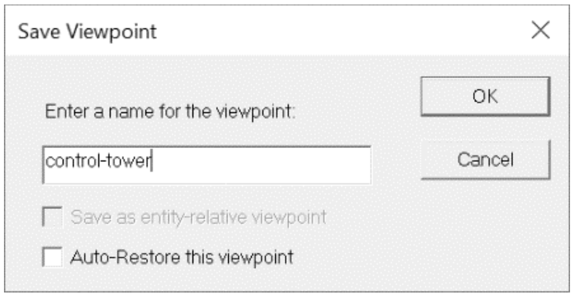
( ) Google Maps URL
(●) Geodetic
( ) MGRS Grid
( ) Geocentric
( ) UTM
( ) Tile ID

Select a viewpoint in the list and click Restore to go to that viewpoint.

Edit the viewpoint’s name.

The coordinate system for the location you enter in the Teleport field.

Click Save to save your current view of the database as a viewpoint. When you do so, a dialog box appears requesting a name for the new viewpoint:


Save Viewpoint
Enter a name for the viewpoint:
control-tower
OK
Cancel
Save as entity-relative viewpoint
Auto-Restore this viewpoint

You can also just press S to save a viewpoint. This saves a viewpoint with a default name "Viewpoint *n*" where *n* is the sequential number of saved viewpoints. You can rename the viewpoint to a meaningful name later.

Viewpoints are usually relative to the terrain. However, if you save a viewpoint while attached to an entity, you can select the checkbox "Save as entity-relative viewpoint" instead.

The Auto-Restore option allows for the viewpoint to be automatically restored when VRSG sees a PDU with the same timestamp as the time associated with the viewpoint when it was saved. This is useful for after-action review (AAR) purposes when the DIS PDUs are being replayed from a DIS playback tool.

By default, viewpoints are named Viewpoint 1, Viewpoint 2 and so on, but you can give viewpoints more meaningful names, as shown in the example above. Rename a viewpoint by selecting the viewpoint in the list and clicking Rename. When the Rename Viewpoint dialog box appears, change the name of the viewpoint and click OK. You can explicitly assign a default viewpoint for VRSG to display when it loads the database, by naming the viewpoint "default." Bear in mind that a viewpoint includes saved near clipping and field-of-view (FOV) settings, thus the "default" viewpoint will restore such saved settings each time VRSG is started.

You can select multiple viewpoints at once, for example to delete them, by holding the Shift key down for selecting contiguous viewpoints or by holding the Control key for selecting noncontiguous viewpoints.

Viewpoints are stored in an editable ASCII .viewpoint file located in the \MVRsimulation\VRSG\Viewpoints directory. The viewpoint file is always named "vrsg.viewpoint" as in \MVRsimulation\VRSG\Viewpoints\vrsg.viewpoint.

Although VRSG always writes new viewpoints to the file \MVRsimulation\VRSG\Viewpoints\vrsg.viewpoint, VRSG supports the presence of multiple viewpoint files located in different subdirectories, including, for example, directories storing terrain tiles organized by geographic area. During visualization, VRSG loads any relevant vrsg.viewpoint files it finds in its search path (that is, the directories listed in the Folders for Terrain, Models, Scenarios, and Other Content section of the Startup Parameters tab) and merges their content. The viewpoint file is formatted as an editable ASCII text file, which means you could copy/paste any number of new viewpoints from the default vrsg.viewpoint file to a terrain-specific viewpoint file.

An example of terrain-specific viewpoints can be found in any of the terrain datasets installed with VRSG, such as \MVRsimulation\VRSG\Terrain\Somalia\Kismayo\vrsg.viewpoint.

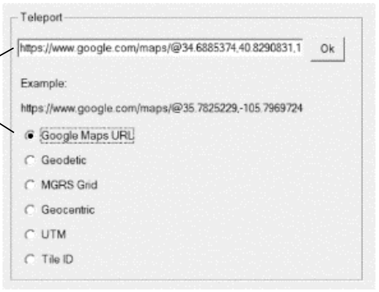
VRSG saves a viewpoint file each time a screen capture is saved to an image file. This saved viewpoint file contains the viewpoint of the exact location on the terrain from which the screen capture was taken. To return to the location of the screen capture, drag the .viewpoint file from the Windows File Explorer and drop it on the VRSG visualization window; doing so moves the eyepoint to the location of the screen capture. Once you use this viewpoint, it becomes persistent; VRSG saves it to the vrsg.viewpoint file and it appears in the list on the Viewpoints tab. See the section "Taking screen captures of the rendered scene" later in this chapter for more information about screen captures.

Note: If you are running VRSG on a system that has User Access Control (UAC) activated, the vrsg.viewpoint file will not be saved or updated in the VRSG installation directory if the \MVRsimulation\VRSG directory is located in C:\Program Files. Instead the updated vrsg.viewpoint file will be stored in the directory C:\Users\<username>\AppData\Local\VirtualStore \Program Files\MVRsimulation\VRSG. To have an updated .viewpoint file take effect the next time you start VRSG, copy the vrsg.viewpoint file from the \VirtualStore\Program Files\MVRsimulation\VRSG directory to C:\Program Files\MVRsimulation\VRSG. To avoid having VRSG files written to the \VirtualStore directory, turn off UAC from the User Accounts control panel if your site policy allows it, or install VRSG in a different directory such as C:\MVRsimulation\VRSG.

To have VRSG move the eyepoint to a precise location in the rendered scene, enter the desired coordinates in the Teleport field of the Viewpoints tab, as an alternative to navigating to it with a game controller. The teleport coordinate system defaults to the display coordinate system set on the Preferences tab, but you can change the teleport coordinate system here, independent of the display coordinates. (Changing the teleport coordinates does not change the display coordinate system.)

New in VRSG 7 is the ability to teleport to a location specified in Google Maps URL format. Select the Google Maps option, then copy/paste a Google Maps location from the browser to the Teleport field, and then click OK.

VRSG can move the eyepoint to a specified Google Maps URL pasted from the browser.



Teleport

https://www.google.com/maps/@34.6885374,40.8290831,1

Example:

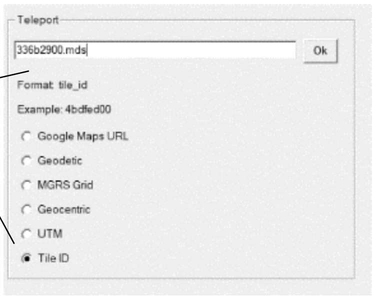
https://www.google.com/maps/@35.7825229,-105.7969724

[✓ Google Maps URL] Geodetic MGRS Grid Geocentric UTM Tile ID

The format requires the text google.com/maps/@ followed by the lat/lon coordinates. Any text in the URL before or after those values is ignored, including other text between "maps/" and "@".

Teleporting to a location based on tile ID is also supported. Select the Tile ID option and enter the name of a terrain tile to center the view on that tile and click OK. Coordinates are encoded within a tile's name. This option is useful for moving the eyepoint to a specific tile.

VRSG can use the name of a specific terrain tile to center the view on that tile.



Teleport

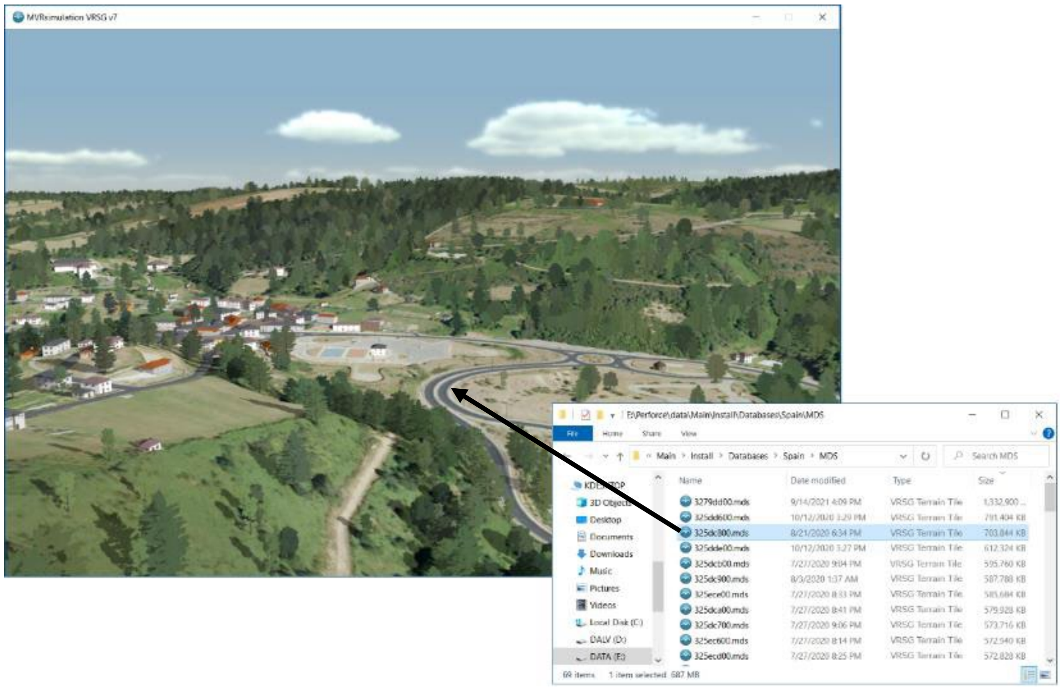
336b2900.mds

Format: tile_id

Example: 4bdfed00

( ) Google Maps URL
( ) Geodetic
( ) MGRS Grid
( ) Geocentric
( ) UTM
(●) Tile ID

Another way you can move the eyepoint directly to a terrain tile of interest in a rendered scene is to simply drag a tile in the current search path from the Windows Explorer to the VRSG visualization window as shown in the following example:



Name	Date modified	Type	Size
3279dd00.mds	9/14/2021 4:09 PM	VRSG Terrain Tile	1,332,900 ...
325dd600.mds	10/12/2020 3:29 PM	VRSG Terrain Tile	791,404 KB
325dc800.mds	8/21/2020 6:34 PM	VRSG Terrain Tile	703,844 KB
325dde00.mds	10/12/2020 3:27 PM	VRSG Terrain Tile	612,324 KB
325dcb00.mds	7/27/2020 9:04 PM	VRSG Terrain Tile	595,760 KB
325dc900.mds	8/3/2020 1:37 AM	VRSG Terrain Tile	587,788 KB
325ece00.mds	7/27/2020 8:33 PM	VRSG Terrain Tile	585,684 KB
325dca00.mds	7/27/2020 8:41 PM	VRSG Terrain Tile	579,928 KB
325dc700.mds	7/27/2020 9:06 PM	VRSG Terrain Tile	573,716 KB
325ec600.mds	7/27/2020 8:14 PM	VRSG Terrain Tile	572,940 KB
325ecd00.mds	7/27/2020 8:25 PM	VRSG Terrain Tile	572,828 KB

69 items 1 item selected 687 MB

Dragging the tile to the VRSG visualization window moves the eyepoint to the tile. This can be useful when you load a large set of tiles with no set viewpoints and the initial scene rendered on VRSG startup is of water. (To identify the tile name of a given geographic region, use the MVRsimulation Terrain Tile Utility, which is described in the chapter "Previewing Models, Effects, and Terrain.")

Setting graphic options

On the Graphics tab, you specify scene display options, such as the field-of-view (FOV) angles (magnification), levels of detail handling, volumetric cloud layer options, screen space ambient occlusion, and the clip region.

The cloud layer's altitude.

Scales LOD ranges across VRSG channels in a multichannel system.

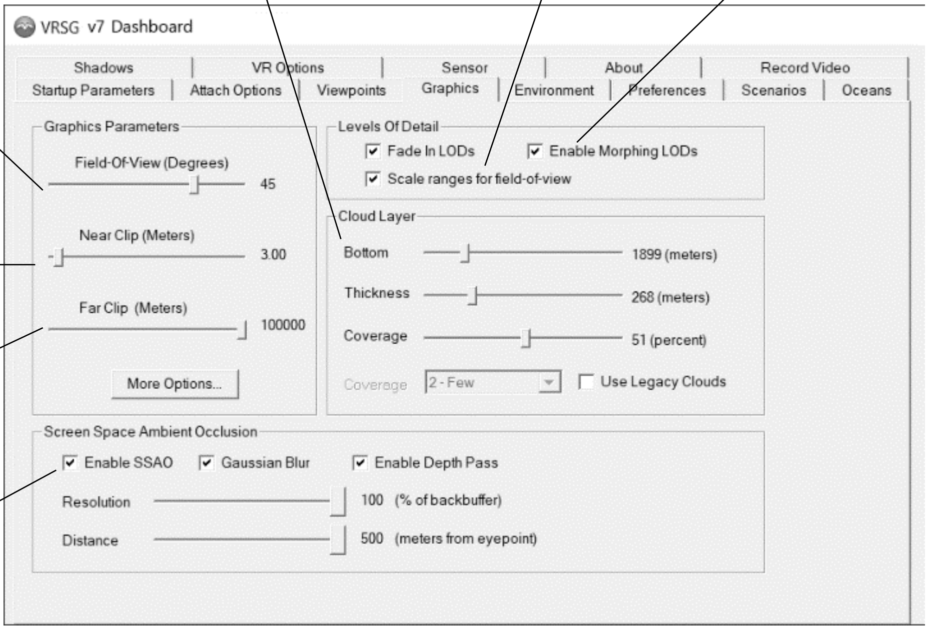
Enables (default) newer handling of transitions between terrain LODs.

The horizontal viewing angle or magnification. Use the slider to zoom in or out of the scene.

How close to the viewer objects should be rendered.

Distance to the horizon.

Simulates shadowing on models in the scene caused by blocked ambient light.



Graphics Parameters
- Field-Of-View (Degrees): 45
- Near Clip (Meters): 3.00
- Far Clip (Meters): 100000

Levels Of Detail
- [✓] Fade In LODs
- [✓] Enable Morphing LODs
- [✓] Scale ranges for field-of-view

Cloud Layer
- Bottom: 1899 (meters)
- Thickness: 268 (meters)
- Coverage: 51 (percent)
- Coverage: 2 - Few
- [ ] Use Legacy Clouds

Screen Space Ambient Occlusion
- [✓] Enable SSAO
- [✓] Gaussian Blur
- [✓] Enable Depth Pass
- Resolution: 100 (% of backbuffer)
- Distance: 500 (meters from eyepoint)

Graphics parameters

Use the Field-Of-View slider to set the horizontal viewing angle in degrees. Larger field-of-view (FOV) angles correspond to lower magnification levels. Conversely, smaller FOV angles correspond to higher magnification levels. VRSG automatically sets the vertical FOV to be aspect-correct with respect to the horizontal FOV, based on the current viewport dimensions. (For information about using more than one viewport, see the section "Using multiple viewports" later in this chapter.) If you need to enter explicit horizontal and vertical FOV half-angles, click More Options to specify them in the More Graphics Parameters dialog box.

Use the Near Clip and Far Clip sliders to define the clip region, the near and far distance at which VRSG should render the scene. If you experience flickering or flashing in the scene caused by z-fighting, adjusting these options, plus the FOV slider described above, are simple ways to remedy the problem.

The greater the far clipping value (distance to the horizon), the more system memory is needed to store and render the scene to that far clip distance. A problem with insufficient memory can occur if you try to run a larger far clip plane than the complexity of the terrain allows. If VRSG issues an "Out of memory" message in the VrsgError.txt file, try changing the far clipping value to a smaller value. If you encounter this problem repeatedly, consider adding more memory to your system.

Keep in mind that Near Clip and Far Clip settings are saved in viewpoints, as described earlier. If your simulation starts, or keeps reverting to, a clipping value that you do not want, check whether you have a viewpoint named "default" (which could be setting the unwanted

Near Clip or Far Clip), you can resave the viewpoint with the intended clipping value, or just delete the default viewpoint.

Levels of detail

Normally, the Fade In LODs option should be selected to provide a smoother and more continuous change between an object's various levels-of-detail. Although enabling this option has a small impact on performance, it improves visual quality by reducing popping of scene elements during level-of-detail transitions. Only unselect this option to improve a performance issue. When this option is not selected, VRSG performs a discrete switch between an object's levels-of-detail; this can result in popping artifacts when an object switches levels-of-detail.

VRSG version 6.5 (and MVRsimulation Terrain Tools 1.6) introduced a new method for rendering the transition between terrain LODs. In this technique, called *LOD morphing*, terrain vertices are blended from their true elevation towards the elevation of the next lower LOD before the lower LOD is reached. This technique (which also referred to as *vertex blending*) replaces the legacy LOD fade-blending method specified by previous versions of Terrain Tools and rendered in previous versions of VRSG.

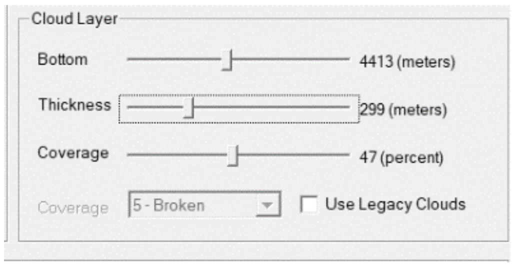
By default, the “Enable LOD morphing” option is selected. If you need to render terrain in an older version of VRSG or a mix of older terrain (created in Terrain Tools 1.5 or older), unselect the Enable Morphing LODs checkbox. This action will force VRSG to render all terrain with the legacy LOD fade-blending method.

The “Scale ranges for field-of-view” option globally scales LOD ranges across VRSG channels in a multichannel system. This option is turned on by default. Turn it off in situations where LOD switch points across multiple channels need to be consistent, as in a dome display, where different channels projecting onto the dome may have different fields-of-view.

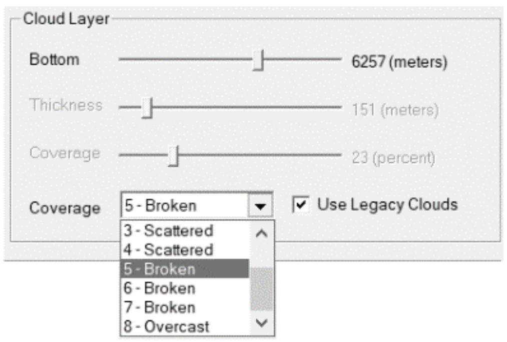
Cloud layer

You control the display of clouds by specifying the altitude of the bottom of the cloud layer, the thickness, and the percentage of cloud coverage.

The volumetric ray-traced cloud system enables you to set the cloud coverage from 0% to 100% (fully overcast).



Cloud Layer
Bottom 4413 (meters)
Thickness 299 (meters)
Coverage 47 (percent)
Coverage 5 - Broken
☐ Use Legacy Clouds



Cloud Layer
Bottom 6257 (meters)
Thickness 151 (meters)
Coverage 23 (percent)
Coverage 5 - Broken
☑ Use Legacy Clouds

3 - Scattered
4 - Scattered
5 - Broken
6 - Broken
7 - Broken
8 - Overcast

Select the Use Legacy Clouds checkbox to use the older volumetric cloud system, where coverage and thickness are combined in predetermined settings from sparse cloud coverage to solid overcast, as shown in the drop-down list in the legacy options shown above on the right.

VRSG scene with the default cloud system.

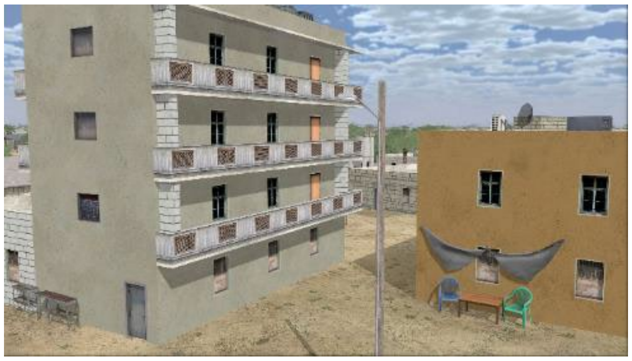
VRSG with legacy clouds enabled.

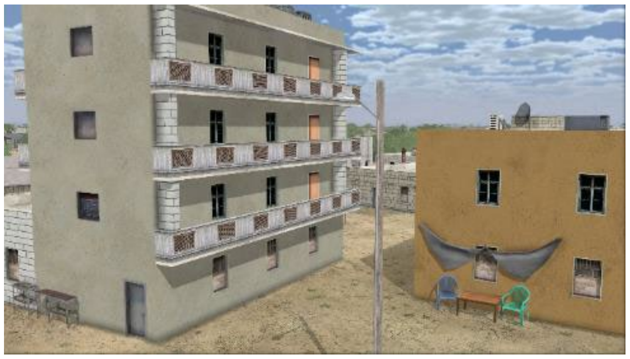
You can also have clouds cast shadows on the terrain by using controls on the Shadows tab, described in the section "Controlling the display of shadows."

Screen space ambient occlusion

Screen space ambient occlusion (SSAO) is a shading effect that enhances the 3D perception of the shape of all models (static and dynamic) rendered in a scene. SSAO simulates the shadowing caused by the blocking of ambient light. By adding darkness to corners, crevices, and other angular transitions, SSAO decreases the ambient lighting that occurs at the intersection of planes on the surface of a 3D model or at the intersection of models and the terrain.

When Enable SSAO is selected, use the Resolution slider to control the size of the framebuffer to render (how much SSAO detail VRSG should render). The default resolution is 50%. The Distance slider controls the distance from the eyepoint at which VRSG should render the SSAO effect. The further the distance, the more graphically realistic the effect appears, but at a higher the cost in frame rate.


VRSG scene without SSAO enabled.


Same scene in VRSG with SSAO enabled.

Click the More Options button on the Graphics tab to display the More Graphics Options dialog box, where you can set field-of-view and rendering options.

The Asymmetrical Field of View options are for creating independent half-angles for top, bottom, left and right.

LOD range scale

LOD Range Scale is a factor for changing LOD ranges globally for the geometry and terrain textures. The number scales the computed range to objects, so smaller values make objects persist longer, adding more geometry to the scene. Load management (set on the Graphics tab) modulates these values dynamically, unless No FOV Scaling is selected. These options are useful to tune issues such as images switching out too quickly, rendering performance needing improvement, and so on. If VRSG displays a message about a "thrashing texture," it means more textures are being requested from the scene than can fit into video memory. Adjust both the Geometry and Texture sliders to a higher value to alleviate the video memory problem. (Default value for both Geometry and Texture is 1.0.)

Rendering options

The availability and effect of the rendering options depends on the 3D graphics card in use.

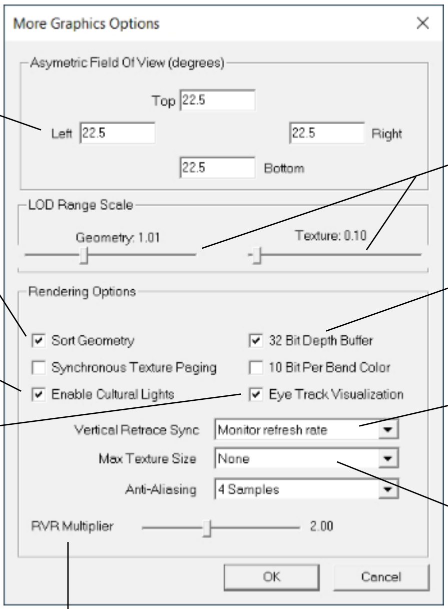
- Sort Geometry performs an object-level sort of transparent objects from back to front.
- Synchronous Texture Paging instructs VRSG to synchronously obtain required terrain textures from disk in the frame that they are requested. If this option is not selected, VRSG uses any valid memory-resident level of detail while I/O is pending, and will transition to the requested level-of-detail when the I/O has completed. If selected, the rendering pipeline may stall to complete pending I/O operations.

For creating independent half-angles.

Performs object-level sort of transparent objects from back to front.

Toggles the display of light points built into the terrain.

Turns on stealth view rendering of the head position and pupil data captured in the DIS stream.


More Graphics Options

Asymetric Field Of View (degrees)
Top 22.5
Left 22.5
Right 22.5
Bottom 22.5

LOD Range Scale
Geometry: 1.01
Texture: 0.10

Rendering Options
[✓] Sort Geometry
[ ] Synchronous Texture Paging
[✓] Enable Cultural Lights
[✓] 32 Bit Depth Buffer
[ ] 10 Bit Per Band Color
[✓] Eye Track Visualization

Vertical Retrace Sync: Monitor refresh rate
Max Texture Size: None
Anti-Aliasing: 4 Samples
RVR Multiplier 2.00

Factor for dynamically changing level-of-detail ranges for the geometry and terrain textures. Load management (set on the Graphics tab) modulates these values dynamically.

Display support for use with monitors, projectors, and graphics cards that are high dynamic range (HDR) compliant.

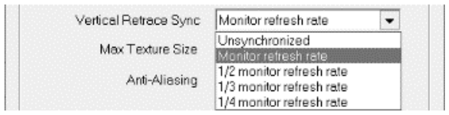
Synchs the frame rate to the refresh rate of the monitor. Selected by default and required for multichannel systems.

Instructs VRSG to downsize all loaded textures to 2048 x 2048, 1024 x 1024, or 512 x 512 pixels. (VRSG can accept a maximum texture size of 4096 x 4096 pixels without needing to downsize.)

Runway lighting visibility range scale factor to apply to the terrain haze settings on the Graphics tab.

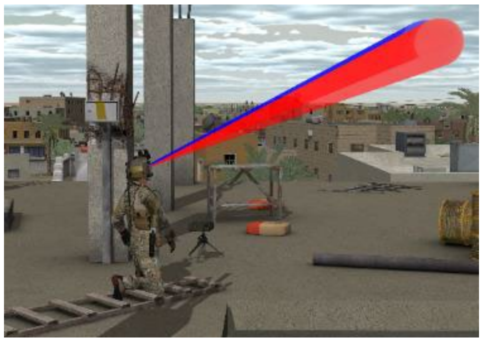
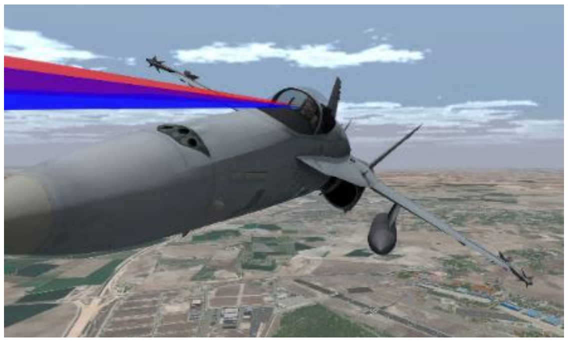
- Enable Cultural Lights (turned on by default) toggles the display of light points built into the terrain with MVRsimulation Terrain Tools (with a shapefile of Cultural Lights feature type) typically used for illuminating road networks. Disabling such light points is useful for ground-based simulations. This option does not disable light point models that are placed on the terrain via a cultural feature file (vrsg.clt), such as runway lights.

- 10-Bit Per Band Color is intended for use with monitors, projectors, and graphics cards that are high dynamic range (HDR) compliant. This option displays many more colors in the scene than the default 8-bit color, resulting in nuanced, fine grain color changes with minimized banding. For example, this option creates finer granularity in the FLIR sensor scene. This option also affects screen captures and H.264 video. To check whether your monitor is capable of handling 10-bit per band color (bpc), open the Nvidia control panel and choose Display > Change Resolution. If 10 bpc is supported by your display, it will appear as an option in the Output Color Depth drop-down list. To use the 10-bit color option in a VRSG mode other than full-screen mode, set that10 bpc option in the Nvidia control panel.
- Vertical Retrace Sync synchronizes the frame rate to the monitor's refresh rate by default. This option is required for multichannel systems that need to stay in phase. Choose Unsynchronized to have VRSG render as fast as possible. However, this option may result in tearing of the visual scene, so it is most appropriate in situations where the visual scene is not important, such as when using VRSG as a dedicated Radar server, or for performance evaluation.



To have the frame rate match the monitor refresh rate, choose the default option and then choose the monitor refresh rate on Preferences tab as described later in this chapter.

- Eye Track Visualization turns on the rendering of the eye-tracking data (head position and pupil gaze) from a Varjo headset that was captured in the DIS stream in a recorded VRSG session, as described in the chapter "Using VRSG with VR Systems, Trackers, and Simulated Military Devices." Head position is depicted with a head model and pupil gaze direction is depicted with red and blue cones, as shown in these two examples:

- Max Texture Size forces VRSG to downsize all loaded textures to 2048 x 2048, 1024 x 1024, or 512 x 512 pixels. The None option (the default) means that VRSG will not downsize any textures. VRSG can accept a maximum texture size of 4096 x 4096 pixels without needing to downsize. Limiting the maximum texture size may help if your content is overloading video or system memory. You can override the selected Max Texture Size option on a per-model basis with a file \Models\max_texture_sizes.txt, in which you list one model per line, followed by the maximum texture size for that model as described in the chapter "Configuring Models and Events."
- Anti-Aliasing performs 2-sample, 4-sample (the default), or 8-sample anti-aliasing on the rendered scene; an option also disables anti-aliasing. Anti-aliasing is a method of smoothing out jagged pixel edges in 3D objects and scenes to improve the visual quality. Where an image's curves and line edges appear jagged, anti-aliasing creates the illusion of blending by placing similarly colored pixels next to one another. The higher the anti-aliasing level, the better the rendered scene appears, but higher levels use more video memory and might cause a performance slow-down on your system. Modern game-level graphics cards support up to 16 subpixel anti-aliasing. VRSG automatically enables 4-sample full-scene anti-aliasing and 8-sample anisotropic filtering. This way, you do not need to set anti-aliasing explicitly for the application in your graphic card's control panel.
- The RVR Multiplier option sets the maximum visibility of a runway light point as a function of terrain visibility (haze settings on the Graphics tab). For example, a setting of 2.0 means that runway lights will be seen two times further than other scene objects.

Setting environmental options

Use the options on the Environment tab to specify various environmental characteristics of the VRSG scene. You can make these changes before or after you start rendering the scene in VRSG; changes you make during a visualization session are reflected in the rendered scene immediately.

VRSG version 7 includes improved light, haze, and cloud interactions. The underlying atmospheric model consists of 16 distinct layers each with unique visibility ranges and wavelength-dependent absorption and scattering properties.

Environmental conditions that you can control in this tab are:

- Sky models
- Haze visibility range and colors
- Ground fog height and range
- Water textures (for 2D water)
- Sun and moon angles
- Time-of-day clock and light source
- Ambient and diffuse lighting levels
- Ephemeris calculated star positions (augmented by a notional 5,000 light point star field for night scenes)

The Visibility settings define the overall atmosphere visibility range of the rendered scene.

The Haze End slider indicates the range at which the terrain completely fades into the haze color. Two haze colors must be provided.

- The Color option is the primary haze color used when the view is not aligned with the sun. Color In Sun is the color of haze used when the view is aligned with the sun. The actual haze color used for a pixel is a blend between the primary haze color and the Color In Sun haze color. The angle between the line-of-sight vector to the object and to the sun determines the ratio between these two colors.
- The Vertical Scale slider changes the rate of visibility improvement as a function of altitude, when Ground Fog is not turned on.

The Ground Fog settings define an atmospheric layer that is close to the ground and ends at a specified height. The density of the ground fog is described by the visibility range in effect in that layer. VRSG calculates the depth into the ground fog layer that light must pass through before it reaches the eye. This effect, which adds further attenuation beyond the atmospheric haze, can be used to achieve the conditions of foggy or smoggy valleys with hilltops that extend up through the fog or smog layer.

The Time of Day/Lighting Conditions options enable you to select a sky model, and create lighting conditions for a day or night scene for simulating the light cast from the sun or moon for shading purposes. You can also direct VRSG to simulate lighting conditions automatically calculated from date, time, and geographic location.

VRSG includes an ephemeris model that can automatically calculate sun position, moon position, star positions, and moon phase from date, time, and geographic location. This means you can direct VRSG to simulate lighting conditions automatically calculated from a specific date, time, and geographic location. Enter the year, month, and day, and drag the time-of-day slider to the desired time. Various environmental settings automatically change in response to the slider movement. You can optionally override the defaults by editing the specific individual setting after you set the time-of-day. Override the size of the stars with the starSizeScalePercent DWORD registry variable, as described in the Appendix “VRSG Registry Variables.”

You can specify an explicit year, month, day, hour, and minute in Zulu/Universal Time (UTC) or local time, or have VRSG obtain the date/time information from your system clock.

You can set specific custom lighting conditions manually in two ways:

- “Diffuse” light illuminates the scene from the direction of the sun (or the moon at night). Diffuse light can cause shadows to be cast. Click the Select Diffuse color rectangle to display a color palette from which you can select a color for the diffuse light source.
- “Ambient” light applies general illumination to the scene, and does not cast shadows. Click the Select Ambient color rectangle to display a color palette from which you can select a color for the illumination.

The options in the Sky Model drop-down list specify the type of sky to use for the scene, such as Dawn, Dusk, High Cirrus, Cloudy, Night, and so on. The sky blends with the selected Fog Color near the horizon. The None option displays the entire sky as the fog color.

To set up a night scene, select the None sky model, adjust the Visibility Haze colors, the Moon light source direction, the Ambient and Diffuse colors, and optionally the moon phase.

The display of the moon requires the moon elevation angle to be above 5.7 degrees, and it be at night time. The ephemeris model predicts the moon position as a function of date, time, latitude and longitude. As long as these four inputs put the moon above 5.7 degrees, and it is night time, the moon will display. (The moon will not display in the daytime even if it is above the horizon threshold.)

Sun and Moon options set the position of the light source (sun or moon). This illumination is independent of any 3D graphic of a celestial body that may be a part of the sky model texture.

The Elevation slider specifies the vertical angle of the light source upward from the horizon.

The Azimuth slider specifies the horizontal angle of the light source relative to north. Note that the sky model is rotated about the Z-axis consistent with the azimuth, which is useful for orienting the horizon glow of the sunset sky.

To specify a 2D wave and swell patterns to appear on the water surface of rivers, streams, and ponds, choose one of seven 2D water texture options. This texture is used for VRSG's default 2D water for non-ocean bodies of water (and for oceans too *if* the 3D Oceans setting is *not turned on* in the Oceans tab). You can also use a custom water texture to override VRSG's default water texture, as described in the chapter "Loading Content into VRSG.")

In addition to using the lighting and weather settings in the Environment tab, you can model wind direction and magnitude for effects such as smoke and dust, as described in the chapter "Configuring Models and Events."

Overall atmospheric visibility range of the rendered scene.

Rate of visibility improvement as a function of altitude, when Ground Fog is turned off.

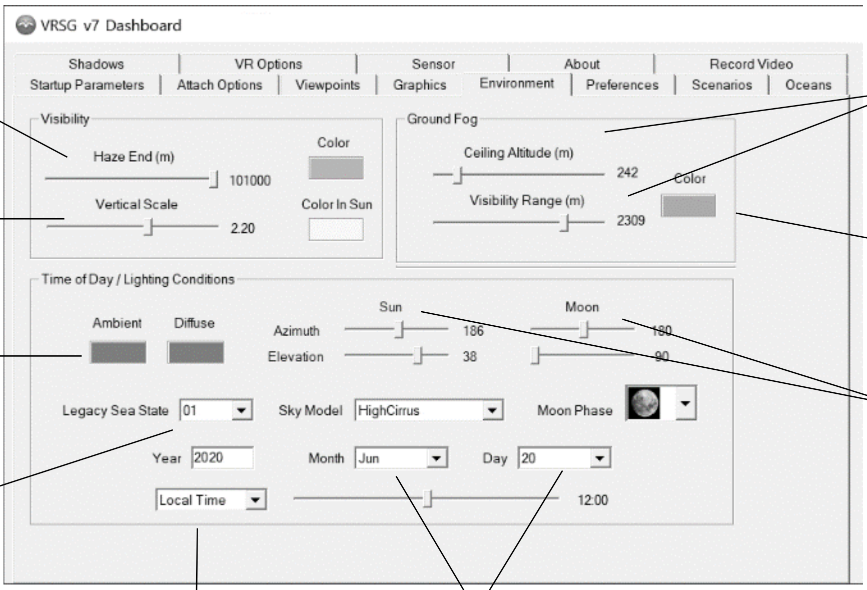
Lighting conditions for a day or night scene.

Sea state options to change the wave and swell pattern on the 2D water surface.

The height and range of the fog layer close to the ground.

Ground fog color.

Simulates the light cast from the sun or moon for shading purposes.



Haze End (m) 101000
Vertical Scale 2.20
Ceiling Altitude (m) 242
Visibility Range (m) 2309
Sun: Azimuth 186, Elevation 38
Moon: Azimuth 180, Elevation 90
Legacy Sea State 01
Sky Model HighCirrus
Year 2020
Month Jun
Day 20
Local Time 12:00

Option to set Zulu/ UTC time or local time for the scene.

Options to advance the time in a scene, like a clock, starting from a specified time.

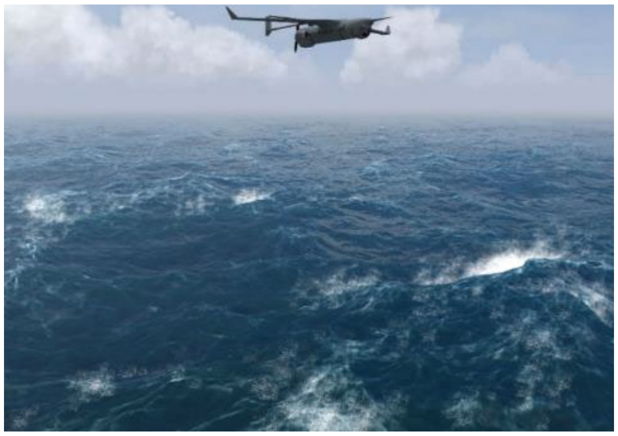
Setting 3D ocean sea states and wakes

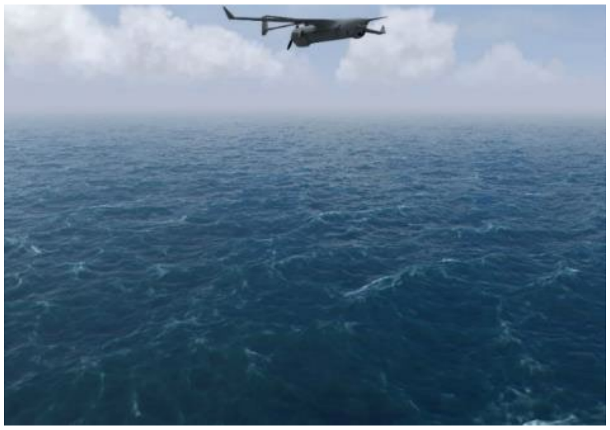
VRSG can simulate 3D ocean sea states with realistic 3D wave motion, multiple sea states, vessel surface motion, 3D wakes, accurate environment reflections, and bathymetry data in MVRsimulation's round-earth terrain format for shoreline wave shape and opacity.

Note: Currently, VRSG supports 3D water at sea-level up to 85 meters. VRSG will continue to render bodies of water (rivers, lakes, and ponds) at elevation higher than 85 meters with its legacy 2D water.

Entities with the DIS enumeration in the surface domain (domain 3) will properly clamp to the ocean surface and will generate wakes automatically. A full network interface enables DIS hosts such as Battlespace Simulations' Modern Air Combat Environment (MACE) to query the ocean surface to support their 3D boat dynamics simulation.

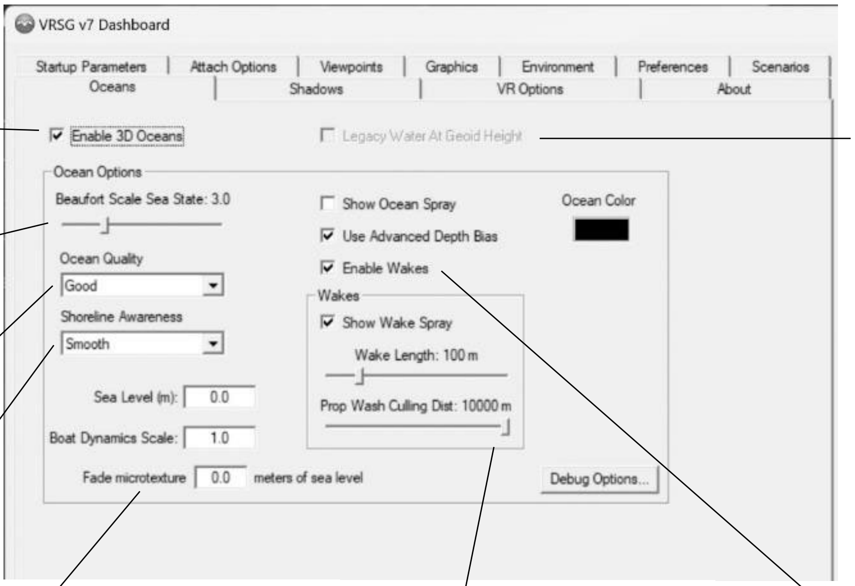
Use the options on the Oceans tab to specify various characteristics of the ocean and sea vessel motion on the ocean surface. You can make these changes before or after you start rendering the scene in VRSG; the changes are reflected in the rendered scene immediately.


3D ocean sea state 12, ocean spray turned on.


3D ocean sea state 6, with no ocean spray.

To activate 3D ocean simulation, select the Enable Oceans checkbox at the top of the Oceans tab. This option toggles rendering VRSG's 3D ocean sea states and legacy 2D water. When this option is enabled, the other options on the tab become available.

- Beaufort Sea State defines the level of impact of wind force on the sea conditions. Specify a sea state based on the Beaufort wind force scale, ranging from 0 (calm seas) to 12 (rolling waves of hurricane conditions).
- Ocean Quality (Good, Better, Best) sets the resolution level (triangle density) of the ocean surface. The Best option renders the highest level of detail, while the Good option has the least impact on VRSG performance.
- Shoreline Awareness (Smooth, None), when set to Smooth, enables a higher-fidelity smooth transition of shallow wave shape and water transparency near the shore, especially when shoreline terrain contains bathymetry data. If you want to raise the sea level and let the ocean flood over the terrain at the natural shoreline, set Shoreline Awareness to None. (Bear in mind that with Shoreline Awareness turned off with None, the wave shape and transparency shoreline features will not be in effect, and the ocean might render above the terrain.)


VRSG v7 Dashboard

Startup Parameters | Attach Options | Viewpoints | Graphics | Environment | Preferences | Scenarios
Oceans | Shadows | VR Options | About

[✓] Enable 3D Oceans
[ ] Legacy Water At Geoid Height

Ocean Options
Beaufort Scale Sea State: 3.0
Ocean Quality: Good
Shoreline Awareness: Smooth
Sea Level (m): 0.0
Boat Dynamics Scale: 1.0
Fade microtexture 0.0 meters of sea level

[ ] Show Ocean Spray
[✓] Use Advanced Depth Bias
[✓] Enable Wakes

Wakes
[✓] Show Wake Spray
Wake Length: 100 m
Prop Wash Culling Dist: 10000 m

Ocean Color

Debug Options...

Turns on 3D ocean simulation and enables the options on this tab.

Sets sea state based on the Beaufort wind force scale (0 to 12).

Sets the resolution (level of detail) of the ocean surface.

Sets a higher-fidelity smooth transition of shallow wave shape and water with transparency near the shore.

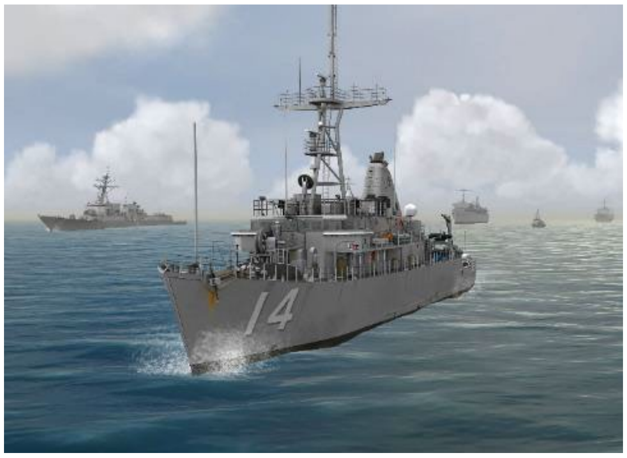
Enables legacy water at the surrounding nominal geoid height. Only if "enable 3D Oceans is unchecked.

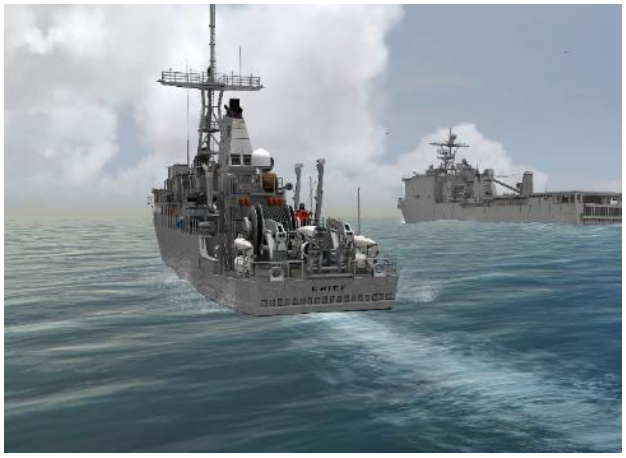
Distance at which to fade-out a shoreline/underwater microtexture.

Controls the distance from eyepoint at which VRSG stops rendering a vessel's propeller wash.

Turns on 3D vessel wake waves.

- Sea Level sets the MSL height of the ocean surface in meters.
- Boat Dynamics Scale controls the buoyancy or realistic motion of vessels, using simple boat dynamics. The smaller the value, the greater the dampening of the motion.
- Fade Microtexture Within controls the distance at which to fade-out a shoreline/underwater microtexture as the eyepoint approaches sea level.
- Show Ocean Spray renders spray particles from breaking waves in choppy sea conditions.


Bow wake: wakes and wake spray options turned on.


Stern wake: wakes and wake spray options turned on.

- Use Advanced Depth Bias calculates where a vessel is in relation to the ocean; by default it is turned on. Unselect the checkbox to turn off this feature when using narrow fields of view, like those used by UAV sensors. Turning off the feature applies a much larger depth bias, thus avoiding potential z-fighting or other visual artifacts.

- Enable Wakes toggles rendering 3D ship wake waves, including bow wave (optionally with spray), stern wave, and prop wash. When this option is enabled, you can also set whether to render the wake spray particle effect and the length of the wakes from 50 to 600 meters.
- Prop Culling Distance controls the distance from the eyepoint at which VRSG stops rendering a given vessel's engine propeller wash. (Can affect VRSG performance.)
- Ocean Color modifies the color of the ocean. Click the ocean color patch, and when the Windows color palette appears, select a color and click OK. (Note the default ocean color appears as black in the color palette, but does not color the water black.) The color you choose is mixed with the default ocean color; it does not completely replace the ocean color. VRSG will retain this color modification across sessions until the next time the ocean color is changed or a settings file is loaded that overrides the ocean color. (Ocean color is saved as part of VRSG's settings file.)

Ocean scene using a custom ocean color.

Note: To change the wind direction to affect the direction of wave motion, press the W key on the keyboard. Each key press rotates the wind direction clockwise a cardinal/intercardinal point.

See the chapter "Configuring Models and Events" for customizing an entity's behavior on the ocean surface with ModelMap.ini commands.

Note for using VRSG with BSI MACE: In addition to supporting VRSG's 3D ocean sea states, MACE has its own physics-based hydrodynamic model which can be used to depict physics-based real-time rendering of vessel behavior. In order to achieve this effect, VRSG transmits to MACE several water elevation points around each vessel entity such that the height and slope of the waves can be used by MACE to generate the resulting realistic vessel movements. In addition to supporting movements of surface vessels, MACE supports the surfacing and submerging of submarines and the wave-jumping of speed boats.

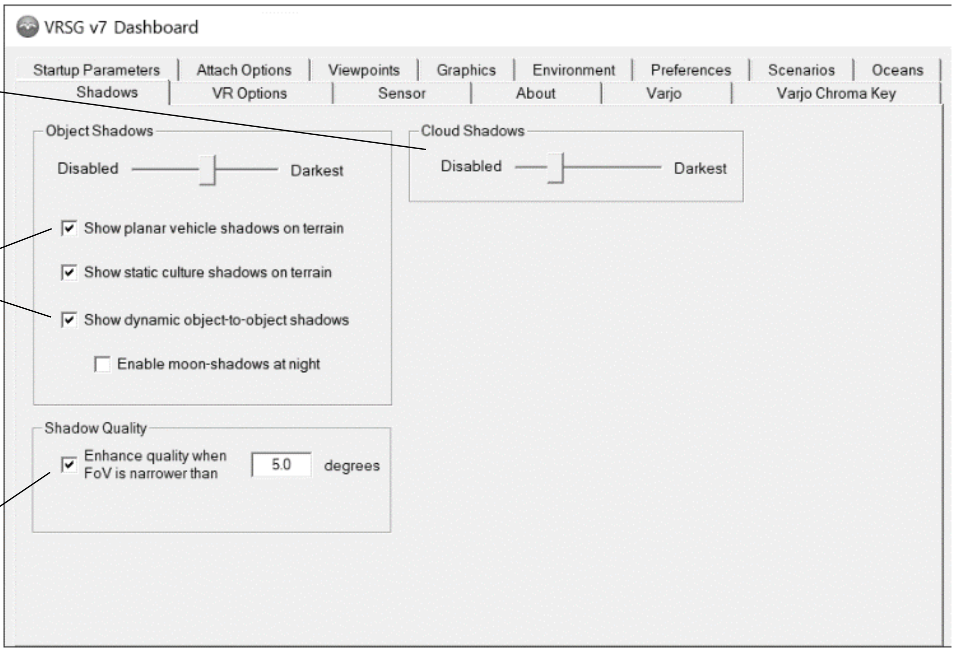
Controlling the display of shadows

Clouds, and static and dynamic models can cast shadows in the VRSG scene. All models (dynamic vehicles and characters, and static culture) can cast realistic shadows onto terrain, water, and everything around them, including self-shadowing. The shadow options are available on the Shadows tab.

Controls the display intensity of shadows cast by clouds on the ground.

Controls the display of shadows cast by all objects in the scene -- dynamic ground entities (vehicles and characters), cultural features, and object-on-object shadows.

Click this checkbox and enter a narrow FOV angle threshold below which dynamic shadow enhancement will take effect.


VRSG v7 Dashboard

Startup Parameters | Attach Options | Viewpoints | Graphics | Environment | Preferences | Scenarios | Oceans | Shadows | VR Options | Sensor | About | Varjo | Varjo Chroma Key

Object Shadows: Disabled — Darkest

Cloud Shadows: Disabled — Darkest

[✓] Show planar vehicle shadows on terrain

[✓] Show static culture shadows on terrain

[✓] Show dynamic object-to-object shadows

[ ] Enable moon-shadows at night

Shadow Quality

[✓] Enhance quality when FoV is narrower than 5.0 degrees

Turn off/on the object or cloud shadow feature with the corresponding slider. Using the checkboxes for shadows of dynamic ground entities (vehicles and characters), cultural features, and object-on-object shadows you can tune performance and retain only the shadows important for the realism of your simulation.

Enabling shadows for clouds, and static or dynamic models instructs VRSG to render a shadow based on the sun position (or moon position, if "Enable moon shadows at night" is selected).

- Use the Cloud Shadows slider to disable or control how much volumetric clouds block directional light. (Set the volumetric cloud layer on the Graphics tab.) When you drag the slider all the way to the right, the shadows cast by the cloud layer completely hide object shadows.
- Use the Object Shadows slider to disable or control the intensity (darkness) of the type of cast shadow.
- Click the checkbox of the type of object shadow to display or hide. All types are selected by default except for "Enable moon shadows at night". (The latter option depends on object-on-object shadows.) To refine performance improvement you can turn off a type of shadow.

VRSG has the ability to improve the appearance of dynamic cast shadows when viewing the scene through a narrow field-of-view (FOV), such as those typically used for UAV sensors. A narrow FOV induces high magnification and greater standoff distances, which can cause shadows to become washed out or to disappear entirely. Click the Shadow Quality checkbox to direct VRSG to render dynamic shadows at a higher resolution, for FOV angles below a given threshold. (You can find the FOV in use on the Dashboard's Graphics tab as described earlier.)

Although cast shadows significantly increase the realism of a scenario, bear in mind that rendering shadows doubles the demand on system resources for the clouds or models casting the shadows. For content-rich scenes that are dense with hundreds of culture models, you can minimize the performance impact of object shadows on the scene by controlling the display of rendered shadows. For example, if frame rate or refresh rate drops when you fly over a dense culture area, consider turning off dynamic object-on-object shadows, as these shadows are less discernable in flight simulation and turning them off will likely improve performance.

Setting client views for multi-channel synchronization

See the chapter “Running VRSG with Multi-Channel Synchronization” for information about setting the controls in the Client Views tab when you want to create a multi-channel stealth system not controlled by a simulation host. VRSG MultiChannel can coordinate multiple VRSG systems as a single synchronized multi-channel system.

Using VRSG with VR system controllers

You can use VRSG with the HTC VIVE/VIVE Pro, VIVE Tracker, Samsung Odyssey+, HP Reverb, Valve Index, Varjo VR-2 and Varjo VR-2 Pro virtual reality systems, and Varjo XR-1 and XR-3 mixed-reality systems. You control various settings for most devices in the VR Options tab.

By default, VRSG assumes the VR system is in seated tracking mode. To use the VR system in standing mode, select the Standing Mode checkbox.

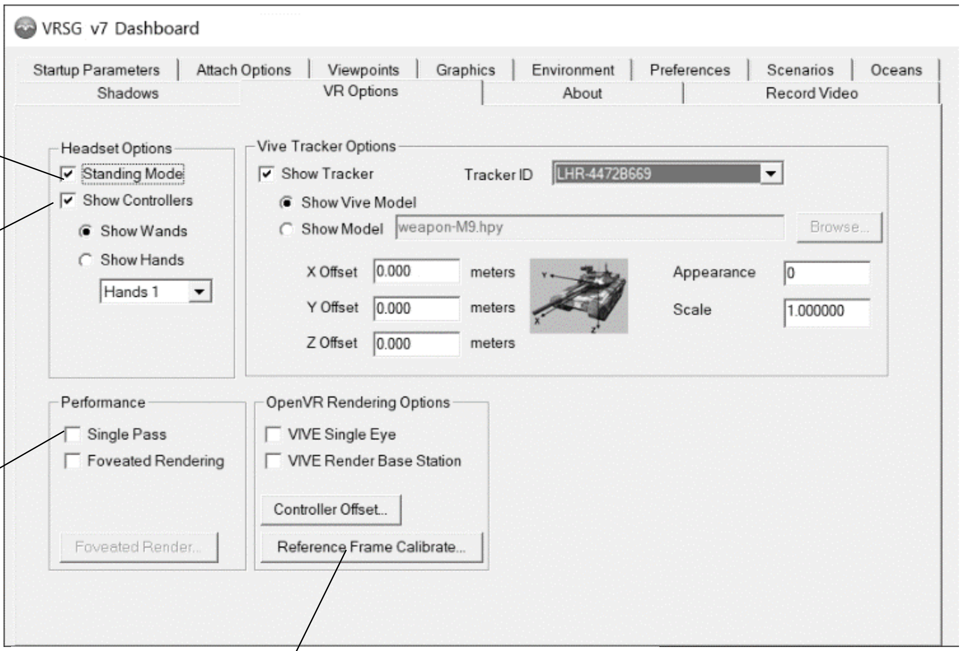
You can instruct VRSG to display the VR motion controllers in the scene by selecting Show Controllers, and then choosing to show the controllers onscreen either as controller models or as hands. Options on the tab also enable you to display a VIVE tracker, optionally display it as a weapon model from the model libraries, and set its scale and offset.

- The Performance options direct VRSG to render the scene in a single pass (rather than the default two passes – one for each eye) and to choose and specify details about foveated rendering.
- The OpenVR Rendering Options instruct VRSG to render the left eye only (“Single Eye”) on a display (rather than rendering the scene of both eyes in a split screen manner) in cases where the OTW view in the HMD is also being rendering externally. Another option renders the base station, which can be useful for orienting yourself/the HMD in relation to actual space.
- The Reference Frame Calibrate option calculates the difference between the reference frames of Windows Mixed Reality VR headsets and the Vive tracker and transforms the result in order to use them together.

Select Standing Mode to override VRSG's default seated tracking mode.

Select Show Controllers, and then choose to show the wands onscreen as wand models or to depict them as a set of hand models.

Renders the scene in a single pass (rather than the default two passes – one for each eyepiece).



Headset Options
[✓] Standing Mode
[✓] Show Controllers
(●) Show Wands
( ) Show Hands
Hands 1

Vive Tracker Options
[✓] Show Tracker
Tracker ID: LHR-4472B669
(●) Show Vive Model
( ) Show Model: weapon-M9.hpy
X Offset: 0.000 meters
Y Offset: 0.000 meters
Z Offset: 0.000 meters
Appearance: 0
Scale: 1.000000

Performance
[ ] Single Pass
[ ] Foveated Rendering

OpenVR Rendering Options
[ ] VIVE Single Eye
[ ] VIVE Render Base Station

Calculates the difference between reference frames of Windows Mixed Reality VR headsets and Vive tracker.

Use of a Varjo device with VRSG requires a plugin DLL, which is installed in the \VRSG\Plugins\HMD directory. The Varjo DLL adds two Varjo-specific tabs to the Dashboards, which you would use with a Varjo device instead of this VR Options tab.

For more information about using VRSG with a VR system, see the chapter "Using VRSG with VR Systems, Trackers, and Simulated Military Devices."

See the section "Starting VRSG from the command-line" later in this chapter for command-line options to have VRSG start immediately ready for use with a VR controller.

Setting display mode, coordinate system, and other preferences

On the Preferences tab, you specify information about the resolution and coordinate system to use for the VRSG virtual world display, the manner and format in which screen captures are saved, and various controller options.

The gain sensitivity of the controller. A value of 1 is minimum gain; 10 results in a highly sensitive controller.

The gain of rotations made by the controller. A value of 1 is the minimum gain.

Select the coordinate system for VRSG to display on screen in real-time.

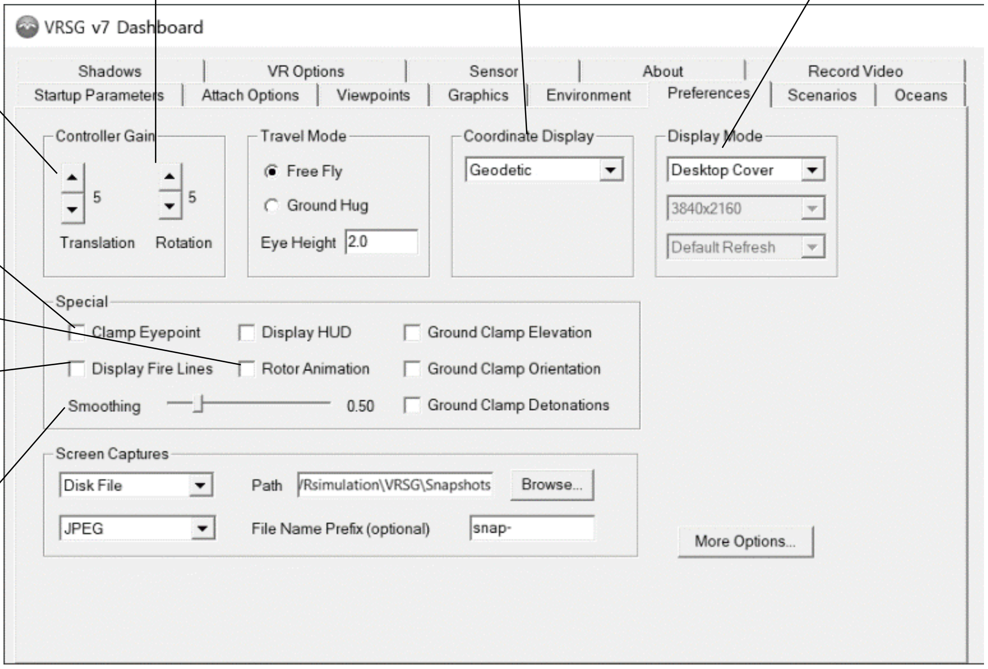
The display mode of the visualization window of the virtual world: desktop cover, or resizable window.

Prevents you from positioning the eyepoint below the terrain level. By default, this option is not selected.

Overrides rotating geometry with flipbook animation.

Draws a "fire line" from the entity/shooter to the target and draws a bounding sphere around the target.

The sensitivity (in seconds) at which to render the motion of an entity; minimizes the jitter effect of rendering updates.


VRSG v7 Dashboard

Tabs: Shadows | VR Options | Sensor | About | Record Video | Startup Parameters | Attach Options | Viewpoints | Graphics | Environment | Preferences | Scenarios | Oceans

Controller Gain: Translation 5, Rotation 5

Travel Mode: (●) Free Fly, ( ) Ground Hug; Eye Height 2.0

Coordinate Display: Geodetic

Display Mode: Desktop Cover; 3840x2160; Default Refresh

Special: [ ] Clamp Eyepoint, [ ] Display HUD, [ ] Ground Clamp Elevation, [ ] Display Fire Lines, [ ] Rotor Animation, [ ] Ground Clamp Orientation, Smoothing 0.50, [ ] Ground Clamp Detonations

Screen Captures: Disk File; JPEG; Path /Rsimulation\VRSG\Snapshots; Browse...; File Name Prefix (optional) snap-

More Options...

Display mode

Use the Display Mode options to control how VRSG presents the 3D real-time scene.

- Sizeable Window displays the 3D scene in a visualization window with Windows borders that you can reposition or resize.
- Desktop Cover displays the 3D scene across the entire monitor's display without borders.

VRSG's window will be created on the selected monitor and display adapter selected on the Startup Parameters tab under the Output Device menu. If a sizeable window is selected, the initial size of the window will be 2/3rds of the monitor. If Desktop Cover is selected, the window will be sized to cover the full selected display.

While running in Desktop Cover mode, you are able to access the VRSG Dashboard by pressing the ESC key. You are also at liberty to Alt-Tab to access other applications without disrupting VRSG's rendering. For best rendering performance however, the VRSG window should be the top-most visible application, with the Dashboard hidden.

You may use the F2 key to toggle between a sizeable window and Desktop Cover without the need to restart VRSG.

The mouse cursor is normally hidden in the VRSG window to provide an unobstructed scene into the virtual environment. The cursor becomes visible upon detecting motion of the mouse and will disappear after the mouse has been idle for 5 seconds.

In addition to these options, VRSG can be started in a fixed, borderless window, which is useful in cases where the VRSG visualization is used within a simulation application that has other onscreen controls. See the section "Running VRSG from the command-line" later in this chapter for information about this fixed-window option and other startup settings.

Fire lines

The Display Fire Lines option draws a "fire line" from the entity/shooter to the target, and draws a bounding sphere around the target. The line moves with the entity. When this option is selected, the fire lines effect is displayed in the scene for the duration specified on the More Options dialog box.

Smoothing entity rendering updates

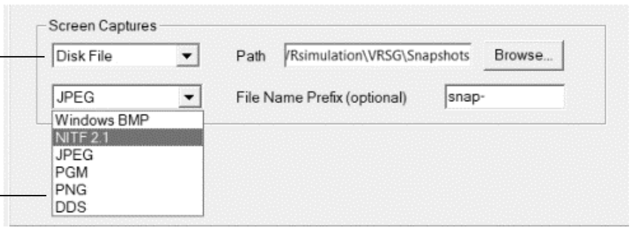
To minimize the jitter effect of rendering updates, the Smoothing slider controls the sensitivity at which to render the motion of an entity, that is, how long it takes for an entity to achieve a PDU's indicated position and orientation from the time of receipt. If smoothing disabled, the position and orientation in the PDU is used immediately in the frame the PDU is received. For DIS entities, zero smoothing is usually not desirable, as discrete jumps and changes in orientation can occur, due to variable latency and dead reckoning thresholds used by the senders. The slider's smoothing units are seconds; the first step on the slider is 0.1 second. This means that when a PDU is received, a new velocity vector and angle rate will be computed to achieve the intended position and orientation in 0.1 seconds from where the entity is currently positioned/oriented. By using the slider, you can explore the tradeoff between smoothness and latency in achieving the PDU ground truth.

Screen captures

You can save screen captures you take in VRSG (by pressing the "C" key on the keyboard) to the Clipboard or to a file in a specified directory path. The example below shows screen captures set to be saved in JPEG format to a file in the default \Snapshots directory. The other formats in which you can save screen captures are shown:

Option to save a screen capture to the Clipboard or to a file.

File formats in which VRSG can save screen captures.


Screen Captures
Disk File
Path /Rsimulation\VRSG\Snapshots
JPEG
File Name Prefix (optional) snap-

To save a screen capture to a file, in the top drop-down list, select Disk File and then select the file format in which the screen capture should be saved. Screen captures can be saved to the default \MVRsimulation\VRSG\Snapshots directory or to a different specified directory. Browse for the directory path in which the capture should be saved, and optionally specify a prefix for the screen capture's filename. Using either the default filename prefix "snap" or a specified prefix, VRSG appends sequential numbers to the names of screen capture image files as they are taken. In this case, the first one would be named snap000.jpg, snap001.jpg, snap002.jpg, and so on. More about taking screen captures in VRSG is discussed later in this chapter, in the section "Taking screen captures of the rendered scene."

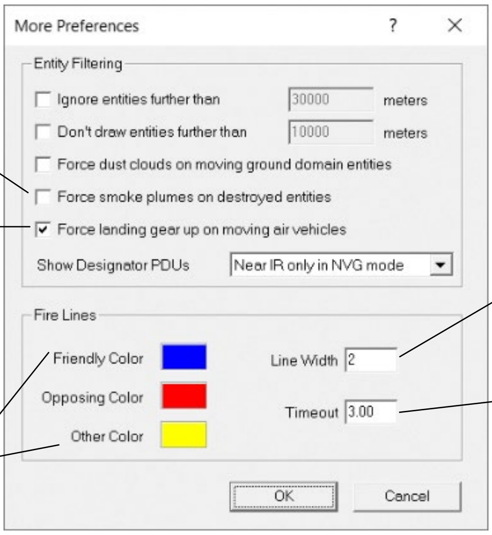
Entity rendering preferences

Click the More Options button to specify additional options affecting entity filtering and rendering of fire lines.

Displays smoke plumes rising from entities that have been destroyed.

Forces the "gear up" switch state on a moving aircraft entity to retract the undercarriage or landing gear, spin the propellers, and so on.

Specifies the colors to use for fire lines, if the Display Fire Lines checkbox is selected on the Preferences tab.



More Preferences

Entity Filtering

- [ ] Ignore entities further than 30000 meters
- [ ] Don't draw entities further than 10000 meters
- [ ] Force dust clouds on moving ground domain entities
- [ ] Force smoke plumes on destroyed entities
- [x] Force landing gear up on moving air vehicles

Show Designator PDUs: Near IR only in NVG mode

Fire Lines

Friendly Color 🟦 Line Width 2

Opposing Color 🟥 Timeout 3.00

Other Color 🟨

Specifies fire line width in meters.

Specifies the duration in seconds that fire lines will remain displayed in the VRSG window.

- The Entity Filtering settings control aspects of rendering entities in a VRSG scene.
- “Ignore entities further than (range in meters)” instructs VRSG to ignore entities further than some range from VRSG's current eyepoint location. PDUs referring to entities further than the specified range from the current VRSG eyepoint will be discarded, treated as if they did not exist. The entities beyond the range will not be visible or available for attachment until they come within the given range.
- “Don't draw entities further than (range in meters)” instructs VRSG to not draw entities that are located further than the given range from VRSG's current eyepoint location. In contrast with the option that precedes it, this option means that the entities beyond the range will be allowed into the system and will be available for attachment; they will merely not be rendered if they are further than the given range.
- “Force dust clouds on moving ground domain entities” displays dust cloud behind moving ground vehicles.
- “Force smoke plumes on destroyed entities” displays smoke plumes rising from entities that have been destroyed.
- “Force landing gear up on moving air vehicles” forces the "gear up" switch state on a moving aircraft entity to retract the undercarriage or landing gear, spin the propellers, and so on. This option is useful for direct control over raising and lowering the landing gear during an entity’s takeoff and landing.
- “Show Designator PDU” displays the designating PDU, in JTAC mode while First Person Simulator (FPS) is running. (For more information about FPS and JTAC mode, see the chapter “Using 3D Characters in VRSG.”)
- The Fire Lines settings control the color and width used for the line and sphere of fire lines. These settings are in effect when the Display Fire Lines checkbox on the

Preferences tab is selected. To change these colors click a color field and select another color. You can also change the width (in meters) of the fire line.

When you display entity information on screen (by pressing the F12 key on the keyboard) the entity information for friendly and opposing entities will display in the corresponding colors set in this dialog box. (If the entity has no designation as friendly or opposing, the information about it is displayed in the third "Other" color.)

Setting sensor-view mode properties

On the Sensor tab, you specify characteristics of the VRSG sensor mode display, such as the amount of noise or blur, the level of intensity of the terrain or vehicles, and whether to display simulated A/C banding. The Sensor tab is displayed in the Dashboard *only* when the Enable Sensor Modes checkbox is selected on the Startup Parameters tab at the start of a VRSG session. You should select the Enable Sensor Modes option only if you need sensor modes in your VRSG session. When this option is not selected, VRSG loads faster and consumes fewer system resources.

The post-processing effects are:

Noise - Adjusts the amount of artificial noise of the scene in all sensor modes.

Focus - Adjusts the intensity of optical focus (blur) of the scene in all sensor modes.

Level - Adjusts the brightness of the scene in all sensor modes.

Gain - Adjusts the contrast of the scene in all sensor modes.

Digital Zoom – Performs a pixel magnification to zoom the image.

Motion Blur - Adjusts the simulated blurred or smeared appearance of objects along the direction of relative motion.

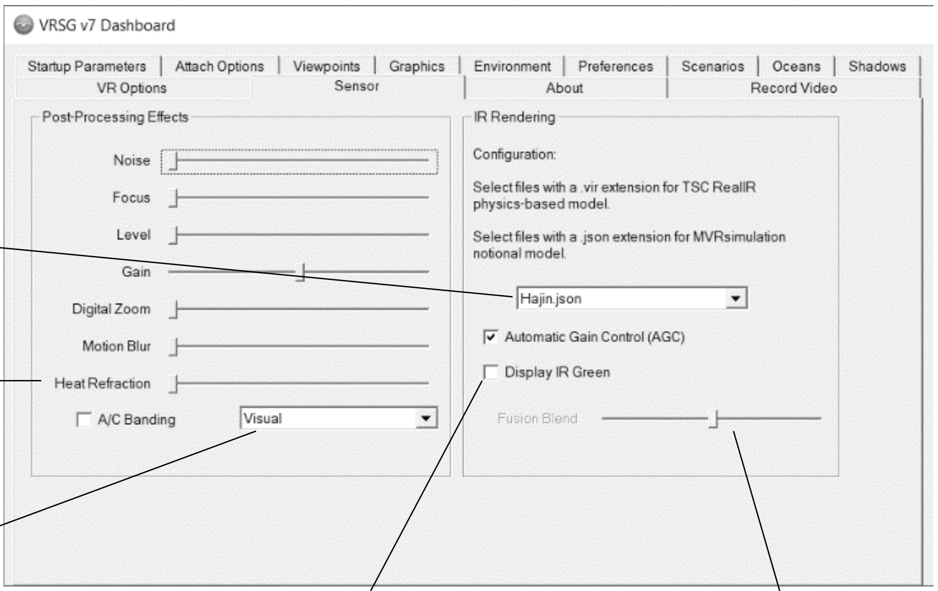
Heat Refraction - Adjusts the simulated heat haze appearance of a scene when it is viewed through a layer of heated air (produced from conditions such as jet fuel exhaust).

The A/C Banding option displays the scrolling horizontal stripe that simulates the banding produced by power supplies that have frequencies dissimilar to the vertical retrace period of the monitor. When you select this option, also select the sensor view on which you want the banding to appear.

The sensor modes drop-down list enables you to choose Electro-Optic (EO), White Hot, Black Hot, or Night Vision Goggles (NVG) sensor modes, or a fusion of EO with White Hot or Black Hot.

The IR Rendering effects control the ratio at which to blend/fuse two sensor modes, specify a physics-based or notional radiance profile, and control whether the white hot or black hot scenes are displayed in green or in black and white and whether to use automatic gain control.

The Configuration option, available as part of VRSG's physics-based IR, enables you to select an IR configuration file, or radiance profile, that was created by hand for notional sensor simulation or with VRSG's IR Setup utility for physics-based IR simulation.


VRSG v7 Dashboard
Startup Parameters
Attach Options
Viewpoints
Graphics
Environment
Preferences
Scenarios
Oceans
Shadows
VR Options
Sensor
About
Record Video
Post-Processing Effects
Noise
Focus
Level
Gain
Digital Zoom
Motion Blur
Heat Refraction
A/C Banding
Visual
IR Rendering
Configuration:
Select files with a .vir extension for TSC RealIR physics-based model.
Select files with a .json extension for MVRsimulation notional model.
Hajin.json
Automatic Gain Control (AGC)
Display IR Green
Fusion Blend
IR configuration file for physics-based IR or notional sensor simulation.
Simulates heat shimmer.
Sensor mode options, including two sensor fusion options.
The color used to depict IR scenes.
The fusion blend ratio, if Fused EO/IR White Hot or Black Hot sensor mode is selected.

The IR Setup utility, described in the chapter "Working with Sensor-View Modes and Physics-Based IR," enables you to configure your runtime IR sensor simulation based on a given set of sensor wavelength limits, and material and environment values. The resulting JSON configuration file (.json) is editable in Notepad or any ASCI editor for making adjustments. A site using a library of terrain tiles of different geographic regions will likely have multiple IR configuration files, or even multiple IR configuration files for terrain of one geographic area based on different environmental settings. (VRSG's physics-based IR capability is available in the US domestic release of VRSG with an unlock code from MVRsimulation, and to international customers with ITAR approval.)

The Automatic Gain Control option automatically selects the radiance minimum and maximum range to which the display dynamic range is mapped; based on the selected IR configuration file.

The Display IR Green option displays the White Hot and Black Hot and fusion modes in green.

Use the Fusion Blend slider to adjust the ratio at which to blend either of the two fused sensor mode options.

For more information about sensor post-processing effects, IR rendering options, and physics-based IR simulation, see the chapter "Working with Sensor-View Modes and Physics-Based IR." Sensor characteristics can also be controlled programmatically as described in the appendix "CIGI Version 4.0 Support."

Displaying version information

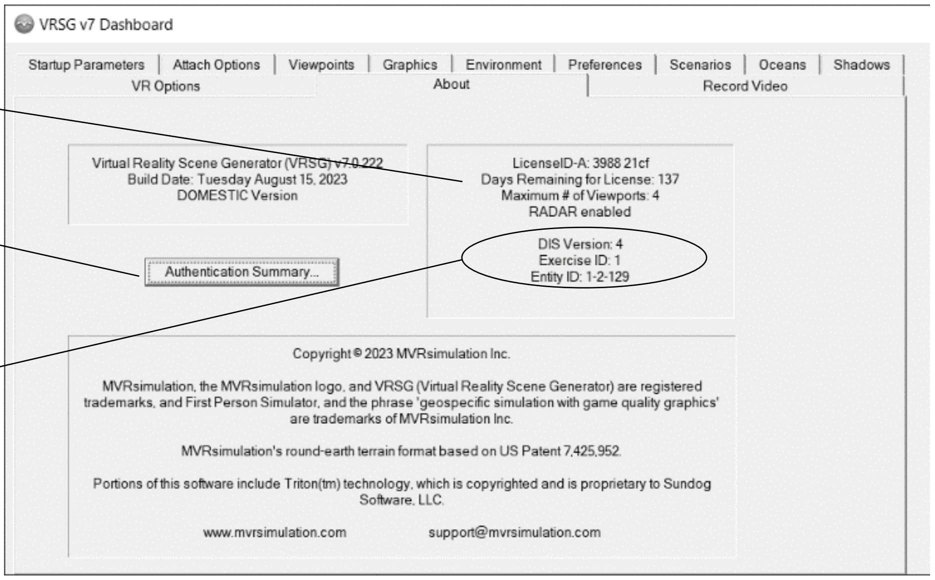
The About tab of the VRSG Dashboard displays which version of VRSG is running and the license ID number. This tab also contains the maximum number of viewports associated with this VRSG license.

When you run VRSG on a network, this tab also lists the exercise ID, and site, host and entity information, as shown in this example:

Maximum number of viewports allowed for this VRSG license.

Click to manage your VRSG license and renew software maintenance.

Exercise ID, and site, host, and entity information of the current VRSG session.



Virtual Reality Scene Generator (VRSG) v7.0.222
Build Date: Tuesday August 15, 2023
DOMESTIC Version

LicenseID-A: 3988 21cf
Days Remaining for License: 137
Maximum # of Viewports: 4
RADAR enabled

DIS Version: 4
Exercise ID: 1
Entity ID: 1-2-129

Copyright © 2023 MVRsimulation Inc.

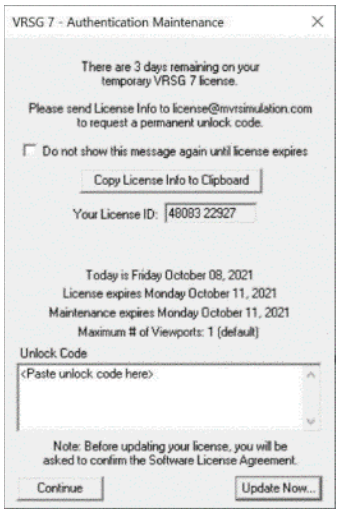
MVRsimulation, the MVRsimulation logo, and VRSG (Virtual Reality Scene Generator) are registered trademarks, and First Person Simulator, and the phrase 'geospecific simulation with game quality graphics' are trademarks of MVRsimulation Inc.

MVRsimulation's round-earth terrain format based on US Patent 7,425,952.

Portions of this software include Triton(tm) technology, which is copyrighted and is proprietary to Sundog Software, LLC.

www.mvrsimulation.com support@mvrsimulation.com

Click the Authentication Summary button to access the Authentication Summary for the VRSG license. You use the Authentication Summary dialog box to enter the MVRsimulation-supplied unlock code associated with your dongle when you first use VRSG after installing it, renew product maintenance, and so on. See the *MVRsimulation Product Installation Guide* for more information.


VRSG 7 - Authentication Maintenance
Copy License Info to Clipboard
Continue
Update Now...

There are 3 days remaining on your temporary VRSG 7 license.

Please send License Info to license@mvrsimulation.com to request a permanent unlock code.

☐ Do not show this message again until license expires

Your License ID: 48083 22927

Today is Friday October 08, 2021
License expires Monday October 11, 2021
Maintenance expires Monday October 11, 2021
Maximum # of Viewports: 1 (default)

Unlock Code
<Paste unlock code here>

Note: Before updating your license, you will be asked to confirm the Software License Agreement.

Starting VRSG from the command line

Instead of starting VRSG from the Dashboard, you can start it from the Windows command line and use one or more options to set up the VRSG session automatically. Launching VRSG this way is useful in situations where the computers driving VRSG are booted up and automatically logged in to a user account, and need to be able to start up VRSG without intervention. This approach is often used in simulators with multiple VRSG channels.

Auto-start

Use the `-autostart` command-line option to start a VRSG visualization session. Doing so is similar to clicking the Start VRSG button on the Dashboard. Starting VRSG this way is useful if you do not need to modify the search path.

For example:

```
Run C:\MVRsimulation\VRSG\Bin\Vrsg7.exe -autostart
```

This command line option starts VRSG, immediately opens the visualization window, and loads terrain tiles from the search path specified on the Startup Parameters tab.

Use a specified settings file

If you want VRSG to load a previously-saved settings file, use the `-settings` option with a specified file, as shown in this example:

```
Run C:\MVRsimulation\VRSG\Bin\Vrsg7.exe -autostart
   -settings=YUMAsettings.json
```

Suppress joystick detection

To have VRSG ignore the detection of a joystick upon startup, use the `-ignoreJoystick` option. This option is useful if you are running VRSG on the same machine as another application that needs control of the joystick.

```
Run C:\MVRsimulation\VRSG\Bin\Vrsg7.exe –ignoreJoystick
```

If you use VRSG with BattleSpace Simulations' MACE, the –ignoreJoystick option is needed when MACE and VRSG are running on the same computer, and you just want MACE to see the joystick.

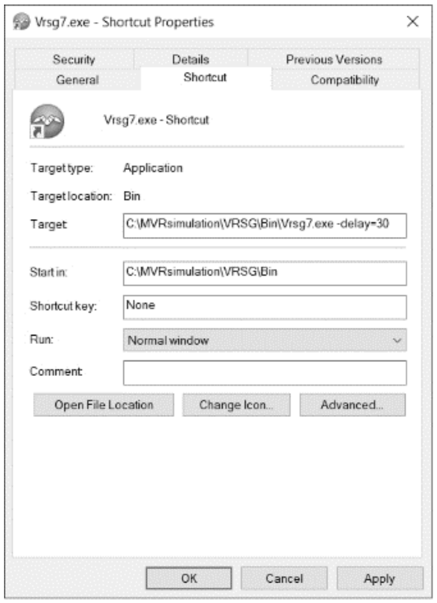
Start VRSG with specified delay

To start VRSG with a delay of a specified number of seconds, use the `-delay=` command-line option. For example:

```
Run C:\MVRsimulation\VRSG\Bin\Vrsg7.exe -delay=30
```

This example instructs VRSG to wait for 30 seconds before launching. This delay gives the system time to power up the display, detect video settings, and so on, and is useful for launching several VRSG channels at once.

You can place such an instruction within a shortcut as well:


Vrsg7.exe - Shortcut Properties

Security | Details | Previous Versions | General | Shortcut | Compatibility

Vrsg7.exe - Shortcut

Target type:	Application
Target location:	Bin
Target:	C:\MVRsimulation\VRSG\Bin\Vrsg7.exe -delay=30
Start in:	C:\MVRsimulation\VRSG\Bin
Shortcut key:	None
Run:	Normal window
Comment:	

Start VRSG in a customized visualization window

In the VRSG auto-startup command, you can have the visualization window launch with specific dimensions, location on the desktop, and optionally borderless. Such setups can be helpful in cases where VRSG is used with another application that contains other displays or simulated controls displayed on the desktop and you want to control the exact position where the VRSG virtual world will be displayed.

To specify the initial position and dimensions of VRSG's visualization window, use the `-windowRect` command-line option in the auto-startup command, with the following syntax:

```
-windowRect=x,y,width,height
```

Provide the position and dimensions in pixels, relative to the upper-left corner of the desktop. These dimensions include any window title bar, borders, and resize corners. If you need to achieve a specific size of the visualization window itself, the dimensions you give will need to be slightly larger to account for window borders and the title bar. This window can be resized and moved anywhere on the desktop.

To specify a non-resizable visualization window, use the `-noresize` command-line option. This option creates a window with a title bar that can be used to drag the window to a new location, but it will not have any resize corners, thus will not be resizable.

To create a fixed visualization window without any title bar, resize corners, or borders, use the `-borderless` option. The resulting window will not be movable or resizable. By coupling this option with `-windowRect`, you can define a specific window size needed for video recording purposes. Since the resulting window lacks any borders, the dimensions specified in `-windowRect` will be the exact dimensions of the resulting visualization window.

Using `-windowRect`, it is possible to create a rendering window that is larger than the monitor can accommodate. This is useful if VRSG is being used to stream video from multiple viewports, and the sum of the viewport dimensions are larger than the monitor.

Automatically load scenarios

To have VRSG start playing a scenario immediately when it finishes loading the terrain in visualization mode, specify the `-scenario` option with a specified file, using the following syntax:

```
Run C:\MVRsimulation\VRSG\Bin\Vrsg7.exe -autostart
-scenario="C:\MVRsimulation\VRSG\Terrain\terrain-name\Scenarios
    \scenario-name"
```

For example:

```
C:\MVRsimulation\VRSG\Bin\Vrsg7.exe" -
settings="C:\MVRsimulation\VRSG\Terrain\Somalia\Kismayo\
Scenarios\AlShabaabCharcoal1\Kismayo-Vrsg-Settings.json" -
scenario="AlShabaabCharcoal1" -autostart
```

This example starts the Al Shabaab Charcoal scenario that is delivered with VRSG as soon as VRSG loads the Kismayo terrain. The specified `scenario-name` only requires quotation marks if the path to the scenario contains spaces.

Disabling pop-up messages

Use the `-nodialogs` option to suppress VRSG from displaying pop-up messages that require a yes/no response, such as "Do you want to see the error log?" Keep in mind that this option will also suppress the reporting of errors that might need your attention. You could create a separate testing VRSG shortcut without the `-nodialogs` option, to see whether there are any issues that you need to address.

Controlling use of a VR or other tracking system

When VRSG detects a VR system or other head-mounted display (HMD) or tracking system, it normally prompts you to confirm whether to use the tracker for your VRSG session. (VRSG does *not* issue a confirmation message for Varjo devices.) To suppress this prompt and have VRSG always use the tracker, add the `-noTrackerPrompt` command-line option.

To prevent VRSG from scanning for a tracking system, use the `-ignoreTracker` command-line argument.

Often tracking sensors are mounted in head-mounted displays (HMDs) or simulated military equipment in a reverse direction. This mounting direction results in tracker azimuth and pitch angles changing in the opposite direction as intended. To account for reverse-mounted trackers, use the `-reverseTracker` command-line argument.

To force VRSG to display the crosshair reticle with simulated military equipment such as binoculars or laser range finder devices, use the `-crosshair` command-line option. This option displays the same crosshair reticle in the VRSG scene as pressing the X key activates (as described later in this chapter). Pressing the X key can still be used to toggle the crosshair display.

Use the `-viveMode=ViveStandingMode` option to use an HTC VIVE VR system in standing mode. By default VRSG assumes the VIVE's room-scale mode.

CIGI command-line option

Use the `-cigi_quiet` option to suppress VRSG from sending CIGI packets to the CIGI host. This option is useful in a multichannel environment where the host should receive CIGI responses from a single channel. See the appendix "CIGI Version 4.0 Support" for information about using VRSG with CIGI.

Displaying onscreen information

As you navigate in the virtual world, you can obtain onscreen information: Help, performance, system memory and texture memory statistics, the name of a model, texture, or terrain tile under the cursor, and a compass rose.

VRSG onscreen Help

The Help text appears as an overlay on the display in the visualization window.

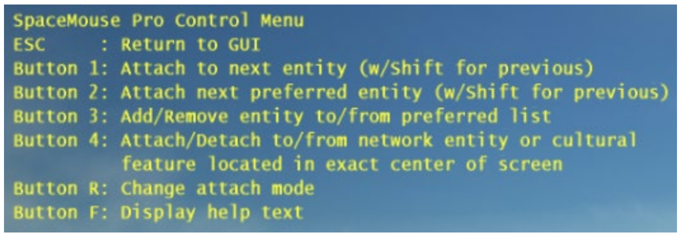
To display Help do one of the following while you are rendering a database in the visualization window:

Press the F1 key on the keyboard to obtain a quick reference to the key sequence associated with each function.

Press the F button on the SpaceMouse Pro to obtain a quick reference to the function associated with each button.

As noted in the Help text, to switch to the VRSG Dashboard, press the Esc key on your keyboard.

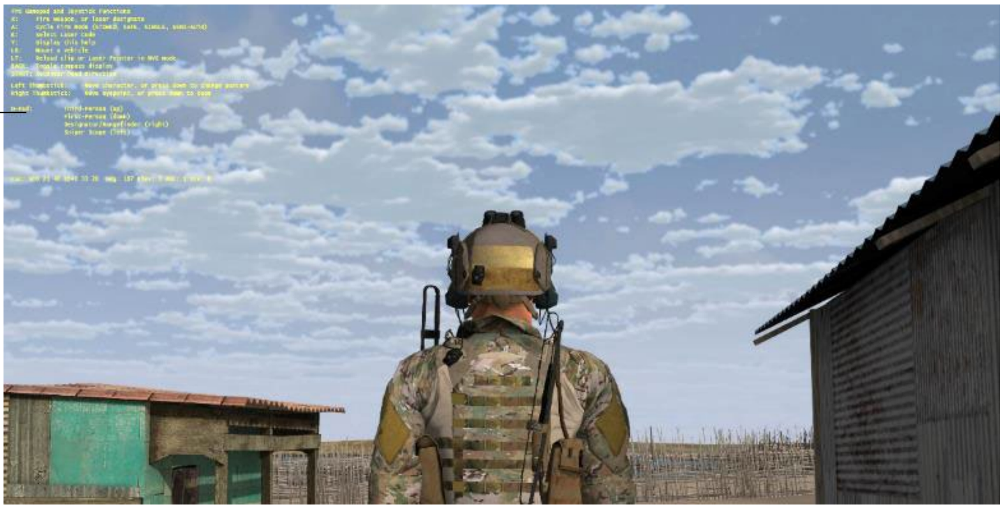
The examples below and on the next page show Help text for the buttons of the 6DOF controller in use and the VRSG keyboard functions**.**



```
SpaceMouse Pro Control Menu
ESC      : Return to GUI
Button 1: Attach to next entity (w/Shift for previous)
Button 2: Attach next preferred entity (w/Shift for previous)
Button 3: Add/Remove entity to/from preferred list
Button 4: Attach/Detach to/from network entity or cultural
          feature located in exact center of screen
Button R: Change attach mode
Button F: Display help text
```

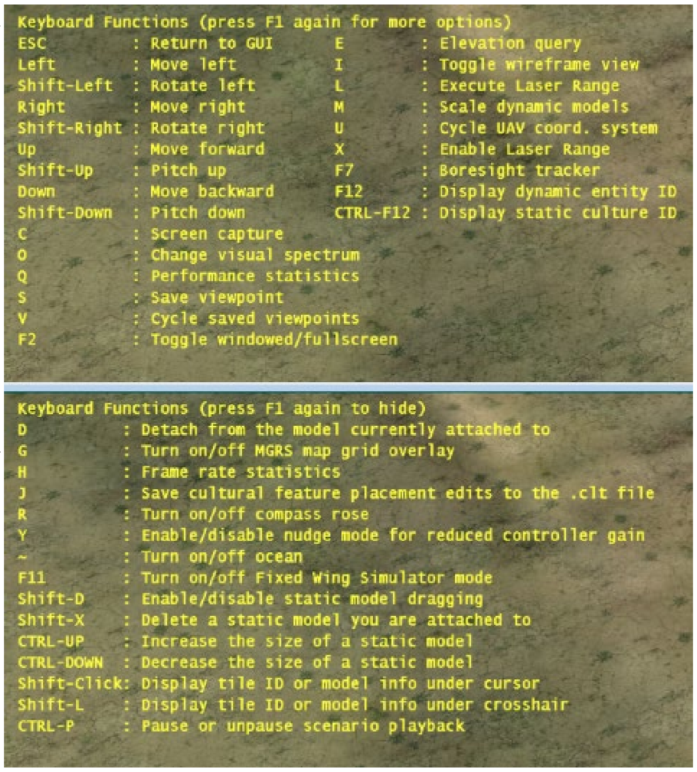
Obtain onscreen help by pressing Button Y on the Logitech F310 gamepad, as shown:

Press Button Y on the Logitech F310 gamepad to display help about the FPS functions mapped to the gamepad controls.



```
FPS Gamepad and Joystick Functions
X:     Fire weapon, or laser designate
A:     Cycle fire mode (stowed, safe, single, semi-auto)
B:     Select laser code
Y:     Display this help
LB:    Mount a vehicle
LT:    Reload clip or Laser Pointer in NVG mode
BACK:  Toggle compass display
START: [illegible]

Left Thumbstick:    Move character, or press down to change posture
Right Thumbstick:   Move eyepoint, or press down to zoom

D-Pad:         Third-Person (Up)
               First-Person (Down)
               Designator/Rangefinder (Right)
               Sniper Scope (Left)

[illegible]
```

Keyboard F1 Help text for keyboard navigation and actions, which spans two screens. Press F1 to display the first screen of keyboard help, press F1 again to display the second screen.



```
Keyboard Functions (press F1 again for more options)
ESC         : Return to GUI       E        : Elevation query
Left        : Move left           I        : Toggle wireframe view
Shift-Left  : Rotate left         L        : Execute Laser Range
Right       : Move right          M        : Scale dynamic models
Shift-Right : Rotate right        U        : Cycle UAV coord. system
Up          : Move forward        X        : Enable Laser Range
Shift-Up    : Pitch up            F7       : Boresight tracker
Down        : Move backward       F12      : Display dynamic entity ID
Shift-Down  : Pitch down          CTRL-F12 : Display static culture ID
C           : Screen capture
O           : Change visual spectrum
Q           : Performance statistics
S           : Save viewpoint
V           : Cycle saved viewpoints
F2          : Toggle windowed/fullscreen
```

```
Keyboard Functions (press F1 again to hide)
D          : Detach from the model currently attached to
G          : Turn on/off MGRS map grid overlay
H          : Frame rate statistics
J          : Save cultural feature placement edits to the .clt file
R          : Turn on/off compass rose
Y          : Enable/disable nudge mode for reduced controller gain
~          : Turn on/off ocean
F11        : Turn on/off Fixed Wing Simulator mode
Shift-D    : Enable/disable static model dragging
Shift-X    : Delete a static model you are attached to
CTRL-UP    : Increase the size of a static model
CTRL-DOWN  : Decrease the size of a static model
Shift-Click: Display tile ID or model info under cursor
Shift-L    : Display tile ID or model info under crosshair
CTRL-P     : Pause or unpause scenario playback
```

Displaying performance statistics about the scene

To obtain performance statistics about the rendered scene, press the "H" key on the keyboard. All displayed times are measured in milliseconds, and frame rates in frames per second.

Current, Average, and Minimum Frame are frame-rate statistics of the last one hundred frames.

Pre-Draw Time is time spent during a frame before VRSG begins drawing the scene.

Draw Time is the time VRSG spends drawing a scene.

Missed Frames is the number of missed frames since last "H" display or for the session.

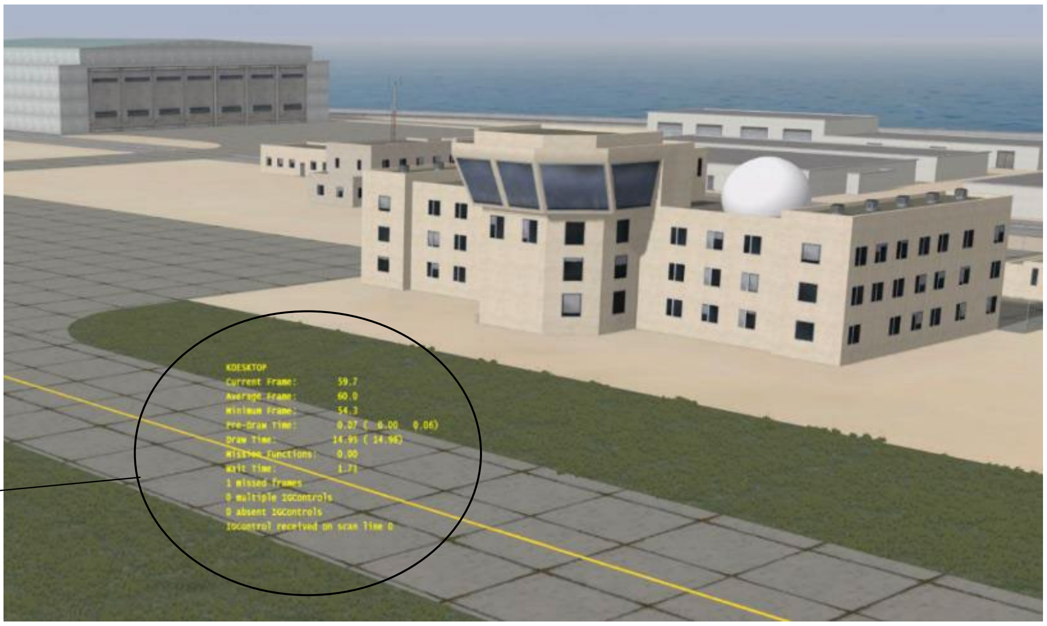
Mission Functions is the compute time to determine collision detection or other mission-function activities.

Multiple IGControls indicates whether the host is running at a faster rate than VRSG. This could mean that VRSG is missing frames or that the host is sending multiple IGControls per frame.

Absent IGControls indicates whether the host is not keeping up with VRSG's frame rate.

Wait Time is the time VRSG spends waiting for vertical retrace.

IG Control Received On Scan Line is a measure of synchronization, VRSG reports the scan line upon which the CIGI IG Control is received from the host. This scan line should be a very small number (less than 50) and fairly stable. If the number increases or decreases, it suggests the host is not synced properly to VRSG. When VRSG sends the Start-of-Frame message during vertical retrace, the host should have the next frame's IGControl precomputed and ready to transmit upon receipt of the Start-of-Frame message.

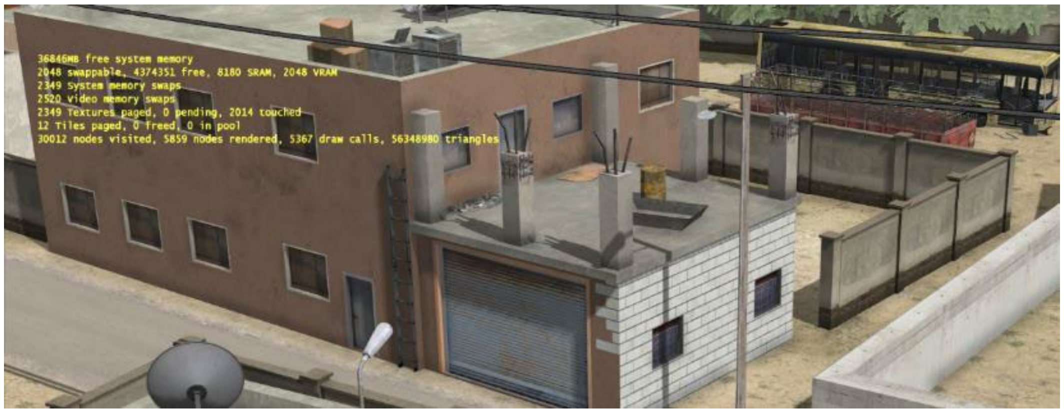


```
KDESKTOP
Current Frame:        59.7
Average Frame:        60.0
Minimum Frame:        54.3
Pre-Draw Time:        0.07 ( 0.00   0.06)
Draw Time:           14.95 ( 14.96)
Mission Functions:    0.00
Wait Time:            1.71
1 missed frames
0 multiple IGControls
0 absent IGControls
IGControl received on scan line 0
```

Keyboard "H" display of a scene's performance statistics.

Displaying system and texture memory statistics

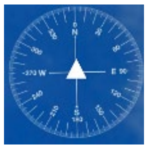
You can obtain system and texture memory statistics about the visualized scene by pressing the "T" key on the keyboard. The statistics appear in the upper-left corner of the visualization window as shown in the following example:



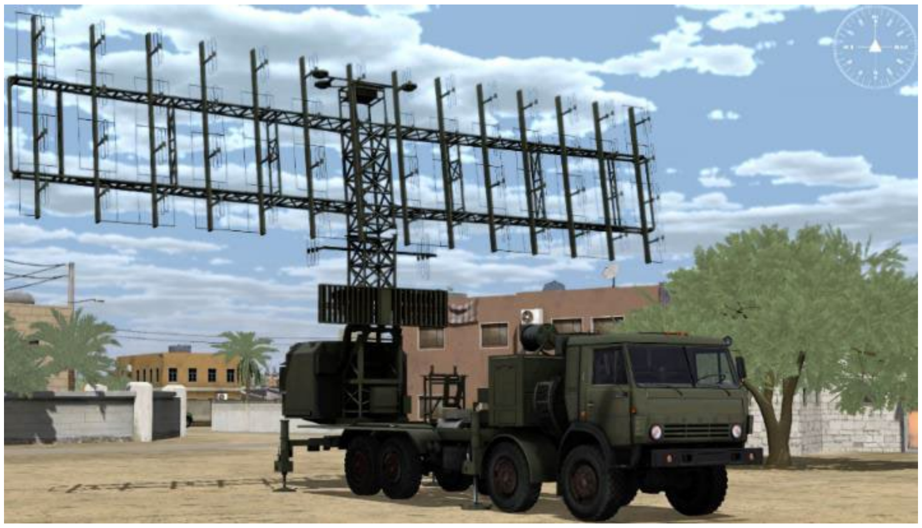
```
36846MB free system memory
2048 swappable, 4374351 free, 8180 SRAM, 2048 VRAM
2349 System memory swaps
2520 video memory swaps
2349 Textures paged, 0 pending, 2014 touched
12 Tiles paged, 0 freed, 0 in pool
30012 nodes visited, 5859 nodes rendered, 5367 draw calls, 56348980 triangles
```

Displaying a compass rose

You can display a compass rose (pointing to magnetic north) in the VRSG visualization window, by pressing the “R” key on the keyboard:



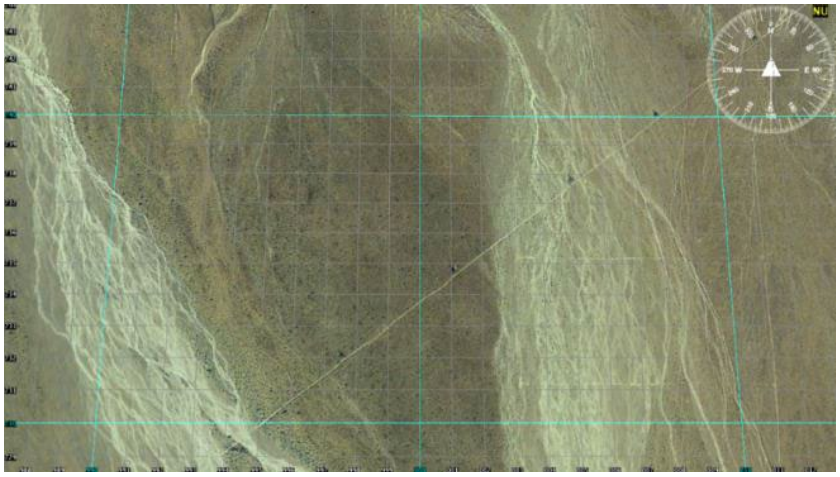
The compass rose, displaying cardinal/intercardinal points, appears in the upper-right corner of the visualization window as shown in the following example:



Press “R” again to remove the compass rose from the VRSG visualization window.

Displaying an MGRS grid

You can display a military grid reference system (MGRS) grid overlay on the terrain. Press the “G” key on the keyboard to display the MGRS grid as shown.



Press the “G” key again to remove the MGRS grid overlay.

You can print the scene with the MGRS grid overlay, by taking a VRSG screen capture of the scene (described later in this chapter) and printing the resulting image.

Playing VRSG scenarios

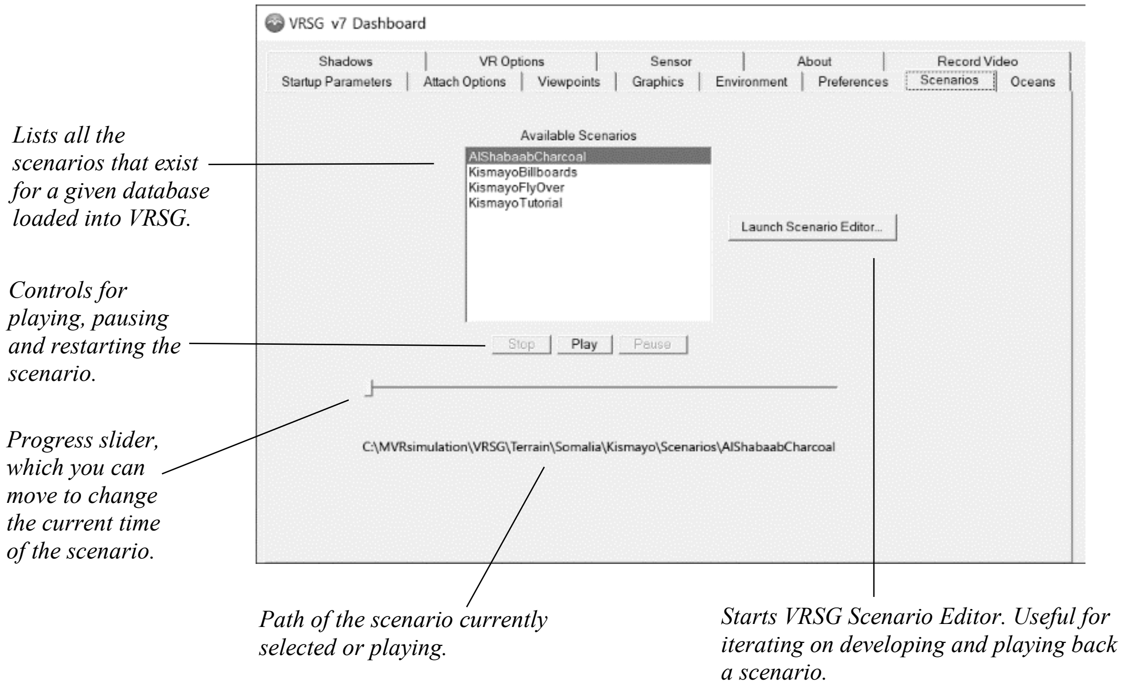
In VRSG you can add scenarios to DIS exercises that contain pattern-of-life activities. These scenarios contain static culture and PDU logs you create in VRSG Scenario Editor (which is installed with VRSG). You can record the movements of characters and vehicles, build up culture content, and play back the resulting scenario in either a network exercise or a standalone VRSG session. See the *MVRsimulation Scenario Editor User's Guide* for more information about creating scenarios.

In VRSG you can play the demo scenarios that are delivered with VRSG and any other scenarios created in VRSG Scenario Editor.

To play a VRSG scenario locally on your machine:

1. Launch VRSG.
2. Ensure the terrain directory and the directory containing the scenario and its associated files are listed in VRSG's search path on the Startup Parameters tab, as described earlier. Also, be sure to select Enable Sensor Modes if that option is needed for the scenario. (It is needed to properly watch MVRsimulation's demo scenarios.)
3. Once the database is loaded in the visualization window, click the Scenario tab.

4. The Available Scenarios list on the Scenarios tab shows all the scenarios that VRSG can find for the terrain specified in the search path.


VRSG v7 Dashboard
Shadows
VR Options
Sensor
About
Record Video
Startup Parameters
Attach Options
Viewpoints
Graphics
Environment
Preferences
Scenarios
Oceans
Available Scenarios
AlShabaabCharcoal
KismayoBillboards
KismayoFlyOver
KismayoTutorial
Launch Scenario Editor...
Stop
Play
Pause
C:\MVRsimulation\VRSG\Terrain\Somalia\Kismayo\Scenarios\AlShabaabCharcoal
Lists all the scenarios that exist for a given database loaded into VRSG.
Controls for playing, pausing and restarting the scenario.
Progress slider, which you can move to change the current time of the scenario.
Path of the scenario currently selected or playing.
Starts VRSG Scenario Editor. Useful for iterating on developing and playing back a scenario.

5. To play a scenario, click the one of interest to select it, and then click Play.
6. VRSG loads the scenario you selected and begins playing it.
7. Click Pause to pause the scenario, and click Play to resume playing it. Click Stop to cease the playback. When VRSG reaches the end of the scenario, it starts playing the scenario again, and will play it repeatedly until you click the Stop button.
8. Temporarily increase the scenario's playback speed 10x by holding down the plus-sign (+) on the keypad or Shift and the plus sign (+) on the keyboard. Increase the playback speed to 100x by also holding down the Control key; Control and plus-sign (+) on the keypad or Shift and Control and the plus sign (+) on the keyboard.
9. Temporarily decrease the playback speed by10x by pressing the minus sign/hyphen (-) on either the keypad or the keyboard.
10. You can stop (and resume) the automatic switching of the eyepoint to the scenario's built-in scripted viewpoints by pressing Ctrl-D. This way you can move around freely in VRSG while the scenario plays.

Playing MVRsimulation's demo scenarios

The terrain that is installed with VRSG includes associated demo scenarios that were created in VRSG Scenario Editor. VRSG is delivered with several 3D terrain datasets that have scenarios. You can play these demo scenarios immediately to see some of VRSG features and Scenario Editor in action. VRSG is set up to initially load the terrain in \MVRsimulation\VRSG\Terrain\Syria\Hajin, which means the scenario in the \Scenarios subdirectory will be available. You can play it as described above, or simply launch it from the VRSG Demos shortcuts located in the MVRsimulation folder on the Windows Start menu.

Note: Turn on the Enable Sensor Modes option on the Dashboard's Startup Parameters tab to properly experience the scenario's sensor view.

Creating and playing real-time VRSG recordings

With VRSG's real-time recording feature, you can record video of VRSG-generated scenes in H.264 or H.265 format. With a current high-end game-level Nvidia card such as the GeForce RTX 4090 or equivalent workstation card such as the RTX 6000 Ada, you can record HD resolution videos at 60 frames-per-second (fps) for either network streaming or file-based recorded output. VRSG can encode UAV Key-Length-Value (KLV) metadata into the video stream to stimulate tactical systems that can exploit that data.

H.264 / H.265 output can be recorded at nearly any resolution you set the VRSG visualization window to be, at 30 or 60 Hz, and can be directed to a local directory on your machine, or streamed to multiple network addresses using the User Datagram Protocol (UDP), Real Time Streaming Protocol (RTSP), or Real Time Transport Protocol (RTP). The MPEG-2 transport stream has an H.264 or H.265 elementary video stream multiplexed within. *(Note that streaming video via RTSP requires that the option be set before you launch the VRSG visualization window.)*

To view VRSG's video feed on the remote end, several viewer options are available, including MVRsimulation's own video player. The MVRsimulation Video Player is a simple, flexible, low-latency player which offers a borderless mode, so it can be embedded easily into cockpit displays. The player also can decode and display encoded KLV metadata. It accepts H.264 /H.265 output created with the UDP MPEG-2 transport stream. The player is distributed with VRSG, with its own installation (Install-MVRsimulation-VideoPlayer-YYYYDDMM .exe), which is located on the distribution media in the \Utilities directory.

The H.264 / H.265 recording plugin supports the latest generation of Nvidia's GPUs which have a hardware H.264 video encoding chip called NVENC. The benefit of this dedicated H.264 or H.265 chip is that most of the processing power of the GPU is available for other tasks.

The ability to record VRSG rendered scenes in real-time is controlled by a plugin (DLL). To activate the plugin, move the file H264.dll from the \MVRsimulation\VRSG\Plugins\VideoRecording subdirectory so that it resides directly in the \Plugins directory, as in: \MVRsimulation\VRSG\Plugins\H264.dll. After you do so, the Record Video tab will appear on the Dashboard the next time you start VRSG.

Note: If your system is not running VRSG on Windows 10 or is not using a newer Nvidia graphics card (or is using a different graphics card altogether), VRSG will not load the H264.dll plugin. Systems running VRSG on Windows 7 or with a different graphics card can use the MPEG.dll instead, which is located in the same \Plugins directory. The MPEG.dll will record the VRSG scene as generally described in this section, but will not encode UAV telemetry, and the recording quality will not be HD.

For recording purposes, you can start VRSG in the sizable or desktop cover window mode. However, do *not* change the window mode or manually resize the VRSG window after you start recording; making either change will cause the recording to stop.

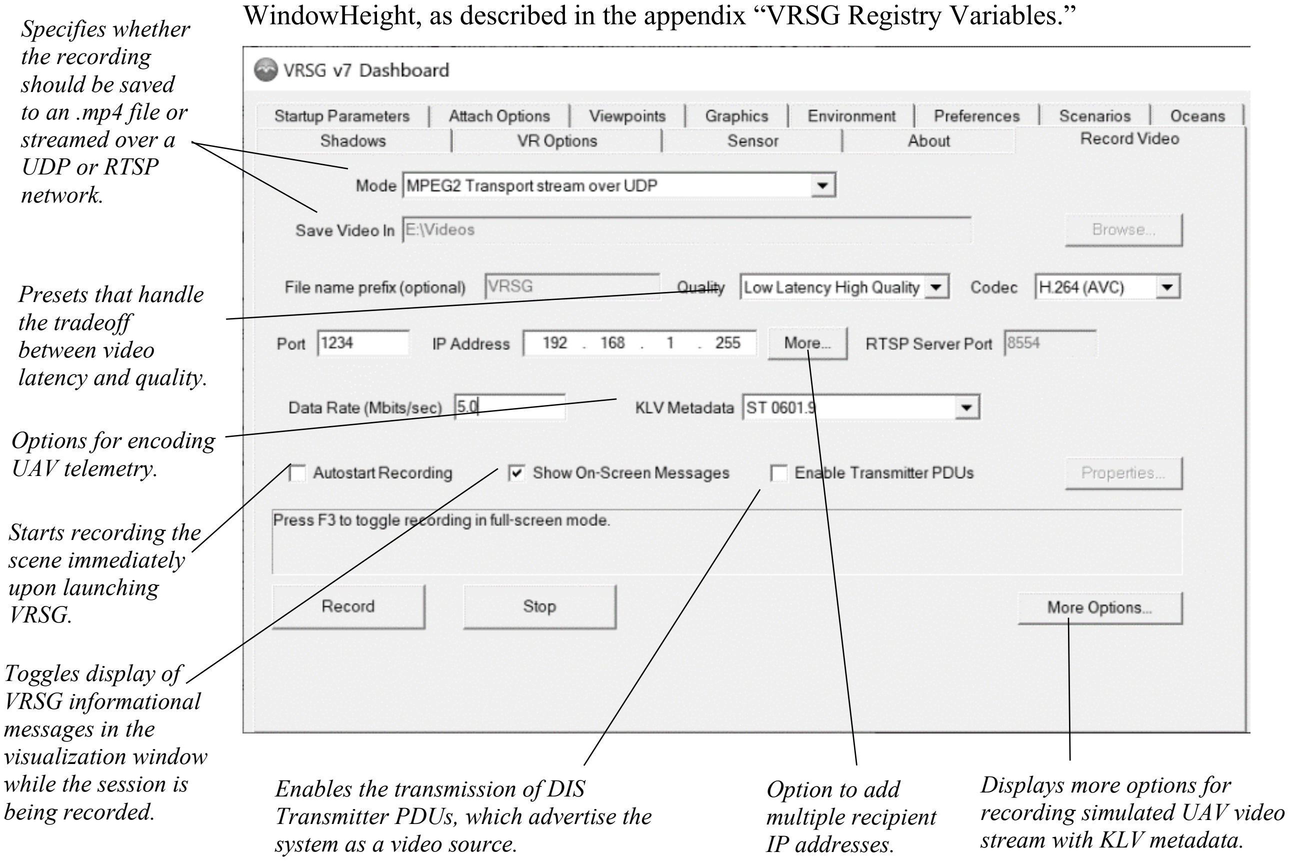
When the file H264.dll is present in the \Plugins directory, the Dashboard displays the Record Video tab as shown in this next example.

To record in Desktop Cover mode without having to access the Dashboard to start and stop the recording, press the F3 key on the keyboard to toggle recording on and off.

If you need the VRSG visualization window to be a specific fixed size on the desktop, set the size by adding the following parameters to the VRSG shortcut in the Windows Start menu:

```
windowRect=x,y,Width,Height
```

The fixed window size can also be set with the registry variables WindowWidth and WindowHeight, as described in the appendix “VRSG Registry Variables.”



Mode: MPEG2 Transport stream over UDP
Save Video In: E:\Videos
File name prefix (optional): VRSG
Quality: Low Latency High Quality
Codec: H.264 (AVC)
Port: 1234
IP Address: 192 . 168 . 1 . 255
RTSP Server Port: 8554
Data Rate (Mbits/sec): 5.0
KLV Metadata: ST 0601.9
[ ] Autostart Recording
[✓] Show On-Screen Messages
[ ] Enable Transmitter PDUs

Press F3 to toggle recording in full-screen mode.

Specifies whether the recording should be saved to an .mp4 file or streamed over a UDP or RTSP network.

Presets that handle the tradeoff between video latency and quality.

Options for encoding UAV telemetry.

Starts recording the scene immediately upon launching VRSG.

Toggles display of VRSG informational messages in the visualization window while the session is being recorded.

Enables the transmission of DIS Transmitter PDUs, which advertise the system as a video source.

Option to add multiple recipient IP addresses.

Displays more options for recording simulated UAV video stream with KLV metadata.

You can have VRSG start recording the scene as soon as it is launched and the terrain is loaded in the visualization window. Select the Autostart Recording checkbox to turn on this feature. Note that to stop the recording, you must unselect Autostart Recording before you click the Stop button.

To suppress the VRSG messages that appear in the visualization window during the recording/streaming session, unselect Show On-Screen Messages.

Recording H.264 or H.265 output

Before you record a VRSG session for H.264 or H.265 output:

- Determine the video dimensions you need. The output resolution of the recording is the resolution of the VRSG visualization window; the default dimensions are 640 x 480 pixels (the default dimensions of VRSG's visualization window in "sizable" windowed mode). To specify alternate video dimensions, you can either set your Windows desktop to the intended size, and start VRSG in Desktop Cover mode, or beforehand, change the dimensions of VRSG's visualization window in windowed mode by creating two DWORD registry variables named *windowWidth* and *windowHeight* as described in the appendix "VRSG Registry Variables." (Note that the display mode setting on the Preferences tab does *not* affect the video dimensions.) *Do not* manually resize the VRSG visualization window once it has been launched.

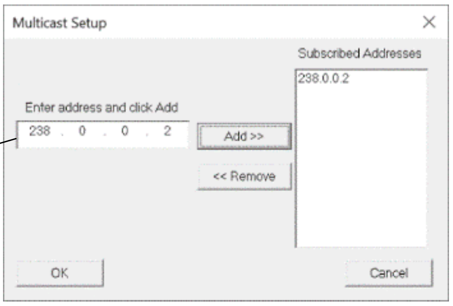
- If you will live stream the video from VRSG, determine whether you need to stream it via RTSP rather than a UDP network stream. VRSG can stream video output to a network address via UDP, RTSP, or RTP, but UDP is the default. If you intend to live stream via an RTSP network stream instead, you *must* select the RTSP option on the Record Video tab *before* you launch the VRSG visualization window. The RTSP option will *not* be available to select once the VRSG visualization window is launched.

To record a VRSG session after the above two considerations have been set:

1. On the Record Video tab, select the following options as appropriate:
    - To write the recorded output to a file, select the Write to File option and specify a path and filename for the .mpg output file.
    - To stream recorded output to a UDP, RTSP, or RTP network address:

      Select MPEG2 Transport Stream Over UDP, Real Time Streaming Protocol (RTSP) Server, or RTP Over UDP, and enter the receiving IP address and port. To transmit to multiple receiving IP addresses, click More and enter the additional IP addresses and ports. (VRSG expects all DIS traffic to be on a single port.)

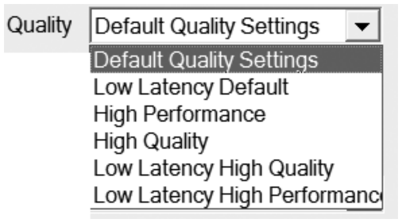
Enter each IP address and port combination and then click Add.



2. Specify the following options as appropriate:
    - Choose H.264 or H.265 codec. VRSG can now encode the video stream into the H.265/MPEG-H format.
    - Optionally, change the default data rate. Higher data rates produce higher quality video, but at the expense of higher bandwidth usage or larger files on disk.
    - When you run VRSG in a UAV camera simulation mode, VRSG can infer the required telemetry during a recording. VRSG supports a compliant subset of the NATO standard STANAG 4609, ST 0601.1, ST 0601.9, and ST 0601.17 KLV metadata and MISB security metadata standard 0104.5. To enable the encoding of UAV telemetry, select one of the using KLV encoding metadata options: ST 0601.1,

ST 0601.9, ST 0601.17, or MISB 0104.5. When one of these options is selected, VRSG generates an MPEG-2 transport stream with two embedded streams: one for video and the other for the selected KLV metadata. If the KLV option None is selected, VRSG records an elementary video stream only, broken into 1024 byte packets.

- Optionally, turn off the display of VRSG informational messages that appear on the screen during recording.
- Optionally, change the Quality setting handle the trade-off between performance and quality:



- Optionally, select Enable Transmitter PDU to have VRSG send a PDU to announce the video source to available channels that can tune in to the streaming video. Click Properties to enter values for the frequency, bandwidth and signal stretch in the Transmitter PDU Properties dialog box.
- Click Advanced to display the Advanced Options dialog box where you can specify more options for recording simulated UAV camera view output with KLV metadata:
- Specify the recorded frame rate as 30 or 60 Hz.
- Optionally, change the default Group of Pictures (GOP) size parameter to optimize the encoding for your output.
- Optionally, unselect B-Frame generation to minimize decoder latency in a network streaming application. B frames enable higher quality video for a given data rate, but incur some latency. B frames should be used when recording to a file, where latency is not a concern.
- Optionally, change the threshold value for the number of hops routers will forward your multicast data packet from VRSG.
- Optionally, change the defaults for Static Metadata encoding options as they pertain to your use of the KLV data set. Any meaningful information you enter in the free text fields will be propagated to the KLV stream.

3. With the terrain loaded in the VRSG visualization window, click Record to start recording the rendered scene. (Or, if you are running VRSG in full-screen mode and do not have the Dashboard displayed, press the F3 key on the keyboard.) During the recording session, this tab shows output information of the session. Anything in the VRSG scene that is written to the frame buffer will be recorded (including any overlays). For information about using more than one viewport, see the section “Using multiple viewports” later in this chapter. If VRSG is running in full-screen mode, you will not be able to access the Record Video tab (or the Record button on that tab) to initiate recording. Press the F3 key on the keyboard to initiate recording while VRSG is in full-

screen mode. VRSG will sound a beep to indicate recording has begun. Press F3 again to stop recording. VRSG will emit a lower-octave beep to indicate recording has ceased.

Enables higher quality video; unselect for live streaming to avoid incurred latency.

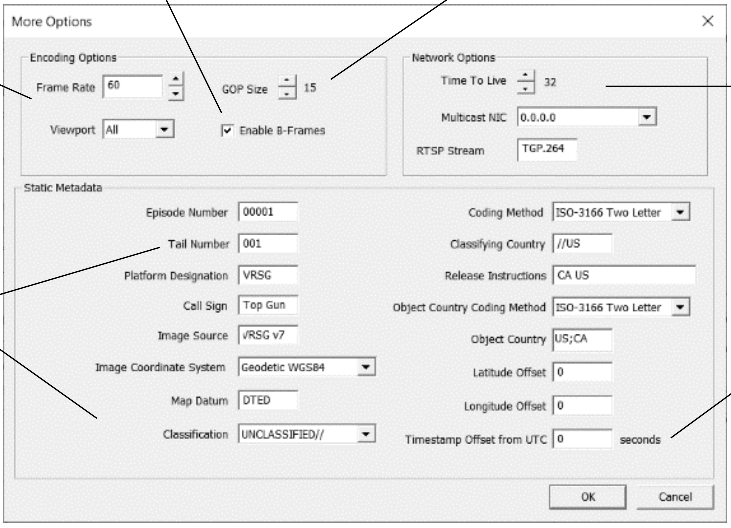
The default Group of Pictures (GOP) size parameter to optimize the encoding for your output.

The recording's frame rate, either 30 or 60 Hz.

Threshold value for number of hops routers will forward a multicast data packet.

Free text fields which can be populated with data that is meaningful for your KLV stream.

A "leap second" offset in generated timestamps in the KLV stream.

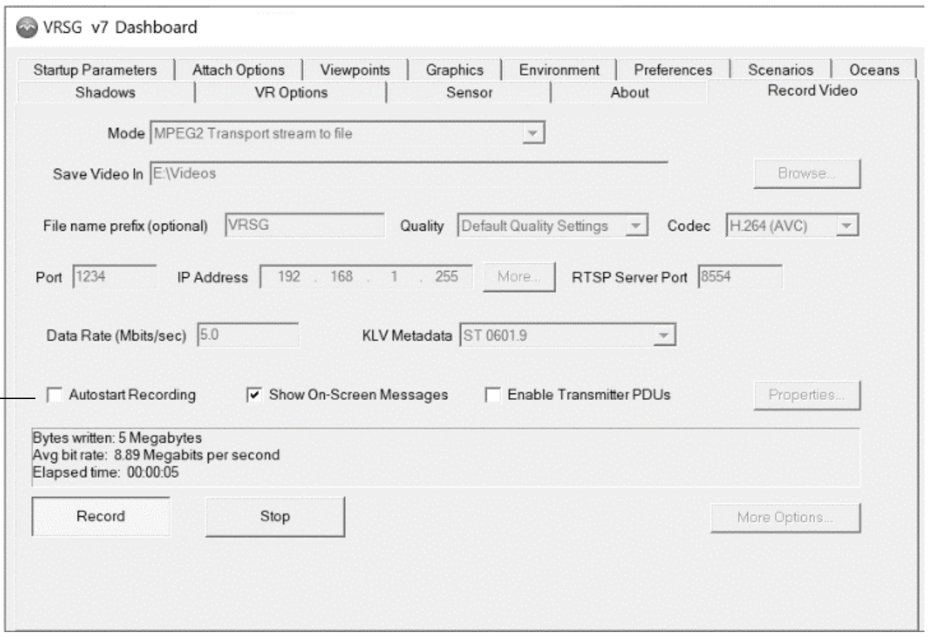


Frame Rate 60
GOP Size 15
Viewport All
✓ Enable B-Frames
Time To Live 32
Multicast NIC 0.0.0.0
RTSP Stream TGP.264
Episode Number 00001
Tail Number 001
Platform Designation VRSG
Call Sign Top Gun
Image Source VRSG v7
Image Coordinate System Geodetic WGS84
Map Datum DTED
Classification UNCLASSIFIED//
Coding Method ISO-3166 Two Letter
Classifying Country //US
Release Instructions CA US
Object Country Coding Method ISO-3166 Two Letter
Object Country US;CA
Latitude Offset 0
Longitude Offset 0
Timestamp Offset from UTC 0 seconds

4. When you finish recording the scene, click Stop.

During the recording session, the Record Video tab displays output information. Anything in the VRSG scene that is written to the frame buffer will be recorded.

Area where output information is displayed during recording.



Mode MPEG2 Transport stream to file
Save Video In E:\Videos
File name prefix (optional) VRSG
Quality Default Quality Settings
Codec H.264 (AVC)
Port 1234
IP Address 192 . 168 . 1 . 255
RTSP Server Port 8554
Data Rate (Mbits/sec) 5.0
KLV Metadata ST 0601.9
☐ Autostart Recording
✓ Show On-Screen Messages
☐ Enable Transmitter PDUs

```
Bytes written: 5 Megabytes
Avg bit rate: 8.89 Megabits per second
Elapsed time: 00:00:05
```

For recording simulated UAV sensor output, note that not all MPEG viewers are capable of using KLV metadata. MVRsimulation's Video Player can decode and display KLV metadata and offers an alternative for customer applications that cannot use VLC for VRSG streaming/playback.

VRSG's generated H.264 / .265 video stream with KLV metadata is fully compliant with the latest standards recommended by MISB. The video output's compliance was tested with NGA's Community Motion Imagery Test Tool (CMITT), which validates video and metadata conformance.

Working with recorded file output

To output the recording to a file, you specify an output directory and optionally a file name prefix. Using the specified prefix, VRSG will append sequential numbers to the names of H.264 / H.265 clips as they are generated.

You can play back the recorded .mpg file, edit it, or convert it to another format.

Examples of recorded scenarios are available on the MVRsimulation website at: www.mvrsimulation.com.

Several viewer options are available for video playback, including MVRsimulation's own video player, which is delivered with VRSG and described later in this section. The MVRsimulation Video Player is lean and flexible and has lower latency than commercially available players such as VLC. The player offers a borderless mode, so it can be embedded easily into cockpit displays, and it can decode and display KLV metadata. If your simulation can use the MPEG2 transport stream, this video player should meet your needs.

Other MPEG viewers, available free of charge, include:

- QuickTime – download is available from www.apple.com/quicktime/download/.
- VLC – download is available from www.videolan.org/.
- Windows Media Player – available on most Windows systems.

For recording UAV camera output, note that the three MPEG viewers listed above are not capable of using the KLV metadata.

MVRsimulation's video player and the GOTS GV 3.0 viewer and metadata editor (the latter available from www.pargovernment.com/gv3.0) are viewers that can decode VRSG's MPEG-2 stream and embedded KLV metadata.

If your recorded playback contains irregular coloration and imagery, check whether your desktop is set to 16-bit color. If it is, change the setting to 32-bit color.

Note: As of VRSG 7, the H.264 plugin will return the encoded KLV metadata via the plugin API, enabling a subsequent executing plugin the ability to access the encoded KLV bitstream. For more information, see the Plugins.h API header file on the MVRsimulation Download Server, in the /Software/Interfaces directory.

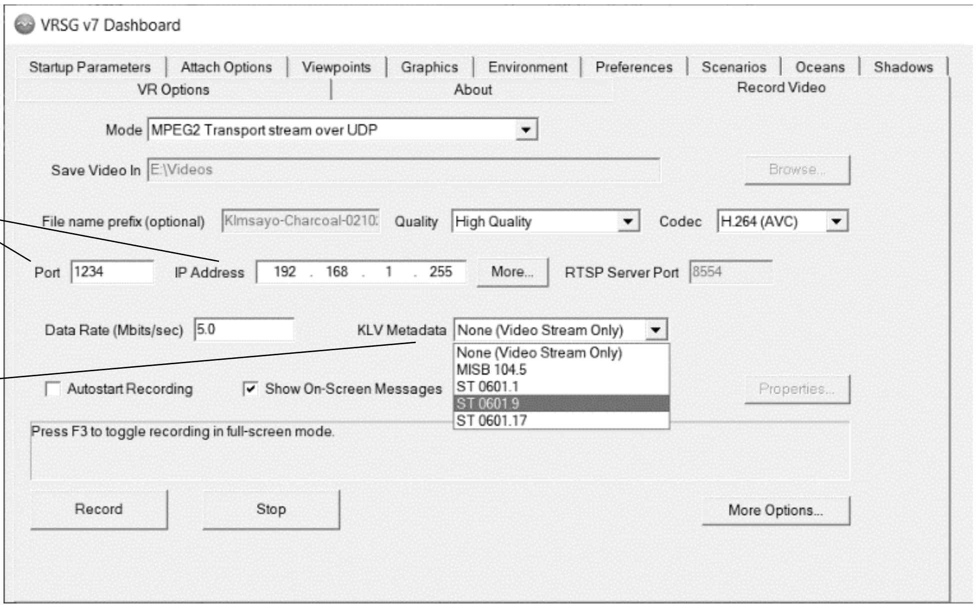
Working with live network streams

Instead of writing an H.264/H.265 stream to a disk file, you can choose to output the recording to a live network stream using UDP or RTSP. If you select MPEG2Transport Stream Over UDP or RTP Over UDP, enter the UDP port and IP address to transmit the video stream, for example, the UDP port and IP address of the machine running the MVRsimulation video player or GV viewer. The IP address can be a peer endpoint, broadcast, or multicast address.

As described above, for UAV camera simulation you can optionally select a KLV Metadata option to have VRSG generate 1316-byte UDP packets, which consist of 7 188-byte MPEG Transport Stream packets. Among the viewers capable of viewing live network streams, only MVRsimulation's video player and GV are capable of decoding and displaying the UAV telemetry encoded in the metadata.

Address and port to receive the recording as a transmitted video stream.

Generates an MPEG2 transport stream with two embedded streams: one for video and another for KLV metadata.



Mode: MPEG2 Transport stream over UDP
Save Video In: E:\Videos
File name prefix (optional): Klmsayo-Charcoal-0210:
Quality: High Quality
Codec: H.264 (AVC)
Port: 1234
IP Address: 192 . 168 . 1 . 255
RTSP Server Port: 8554
Data Rate (Mbits/sec): 5.0
KLV Metadata: None (Video Stream Only)
- None (Video Stream Only)
- MISB 104.5
- ST 0601.1
- ST 0601.9
- ST 0601.17

Autostart Recording [ ]
Show On-Screen Messages [✓]

Press F3 to toggle recording in full-screen mode.

You can direct the streaming output to one of these three viewers:

- MVRsimulation's video player – delivered with VRSG and located in \MVRsimulation\Video Player (for streaming VRSG video in H.264 or H.265 format).
- VLC – shareware available from www.videolan.org/.
- GV 3.0 – The GOTS package GV 3.0 viewer and metadata editor available from www.pargovernment.com/gv3.0.

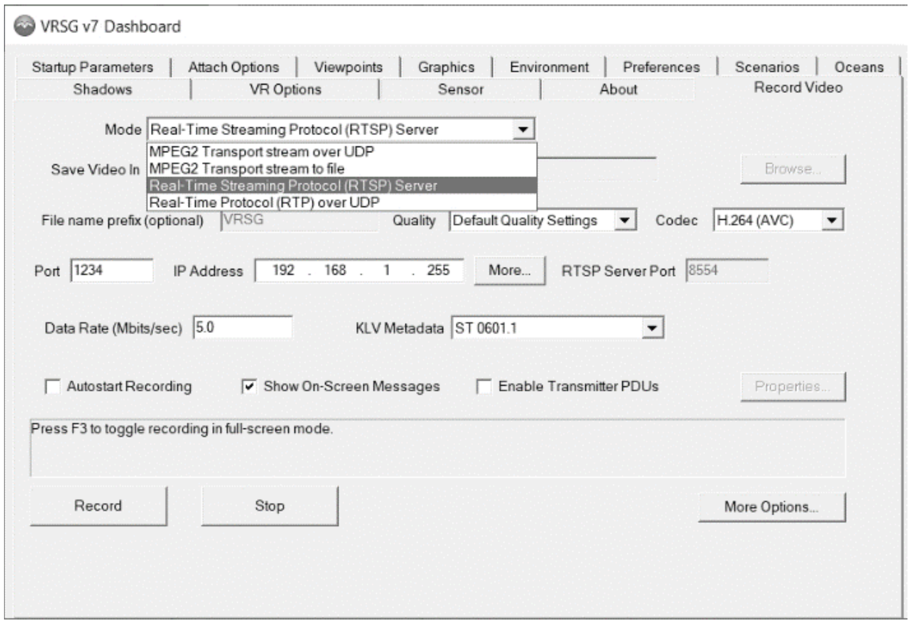
Windows Media Player is *not* capable of playing the H.264 in a transport stream over UDP.

There is no perfect setting for a recording, especially in high definition and high frame rate situations. The balance between data rate (Mbits/sec) and recording preferences is up to you. Using an NVidia RTX 3080 video card and a data rate of 25 Mbits/sec with the H.264/265 codec yields a high-quality result. Reducing the data rate will reduce the fidelity of the recording. Increasing the data rate will eventually overload the processor and GPU, resulting in a choppy and stuttering video recording.

Post-processing, while not necessary, is also user dependent. Stretching and shrinking the picture may give unwanted results upon rendering in VRSG. MVRsimulation recommends you keep the recording in the native resolution. If a higher resolution than the native alternative is needed, it is best to increase the resolution in VRSG, and re-record the exercise or activity.

Streaming output via RTSP

If you intend to live stream via an RTSP network stream instead, you *must* select the RTSP option and, optionally the RTSP Server Port on the Record Video tab *before* you launch the VRSG visualization window. The RTSP Server options will *not* be available once the VRSG visualization window is launched.


VRSG v7 Dashboard

Mode: Real-Time Streaming Protocol (RTSP) Server
- MPEG2 Transport stream over UDP
- MPEG2 Transport stream to file
- Real-Time Streaming Protocol (RTSP) Server
- Real-Time Protocol (RTP) over UDP

File name prefix (optional): VRSG
Quality: Default Quality Settings
Codec: H.264 (AVC)
Port: 1234
IP Address: 192 . 168 . 1 . 255
RTSP Server Port: 8554
Data Rate (Mbits/sec): 5.0
KLV Metadata: ST 0601.1
Autostart Recording: unchecked
Show On-Screen Messages: ✓
Enable Transmitter PDUs: unchecked

Press F3 to toggle recording in full-screen mode.

The RTSP Server Port is the loopback address to use if the video player (such as MVRsimulation's video player or VLC) is located on the same machine as VRSG. Specify the RTSP Server Port in the form of rtsp://127.0.0.1:8554/TGP.264, as shown in the VLC example as shown in the example in the section "Using VLC." If the video player is located on another machine, use the IP address of VRSG on the remote machine. The VRSG visualization window *must be launched* and rendering before making the connection to the remote machine.

Note: The inclusion of KLV metadata is not available with the RTSP option.

Streaming output via RTP

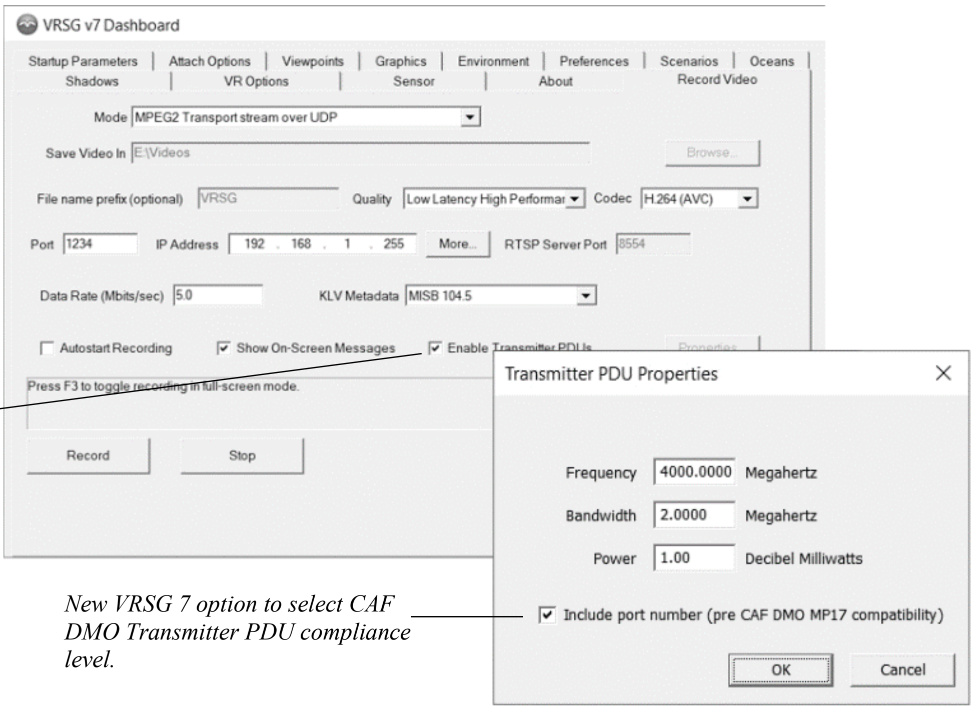
You can stream video via RTP over UDP. RTP is the DMO standard for tactical video streaming. RTP can output KLV metadata streaming through a supported viewer like MVRsimulation's video player.

Emitting transmitter PDUs

VRSG's streaming feature can emit transmitter PDUs, as shown next.

1. Select the Enable Transmitter PDUs checkbox and click Properties.
2. When the Transmitter PDU Properties dialog box appears, enter the frequency, bandwidth and signal strength information for other channels to tune into the video stream, and click OK.

Select the checkbox, click the Properties button, and in the Transmitter Properties dialog box, enter the frequency, bandwidth and signal strength.


VRSG v7 Dashboard

Startup Parameters | Attach Options | Viewpoints | Graphics | Environment | Preferences | Scenarios | Oceans
Shadows | VR Options | Sensor | About | Record Video

Mode: MPEG2 Transport stream over UDP
Save Video In: E:\Videos
File name prefix (optional): VRSG
Quality: Low Latency High Performar
Codec: H.264 (AVC)
Port: 1234
IP Address: 192 . 168 . 1 . 255
RTSP Server Port: 8554
Data Rate (Mbits/sec): 5.0
KLV Metadata: MISB 104.5
☐ Autostart Recording ☑ Show On-Screen Messages ☑ Enable Transmitter PDUs

Press F3 to toggle recording in full-screen mode.

Transmitter PDU Properties

Frequency: 4000.0000 Megahertz
Bandwidth: 2.0000 Megahertz
Power: 1.00 Decibel Milliwatts
☑ Include port number (pre CAF DMO MP17 compatibility)

New VRSG 7 option to select CAF DMO Transmitter PDU compliance level.

Streaming multiple viewports

You can stream multiple viewports from within a single MVRsimulation channel over the network. The video recording feature is viewport-aware, which means you can select the viewport to encode on the More Options dialog box (accessed from the Record Video tab). To set this up, you will need to install multiple H.264 DLLs, by simply making copies of the DLL file. For example, make a copy of the file \Plugins\H264.dll in the same directory and name it H264_2.dll. Doing so creates two Record Video tabs on the Dashboard, and each can record a different viewport. (For information about using more than one viewport, see the section "Using multiple viewports" later in this chapter.)

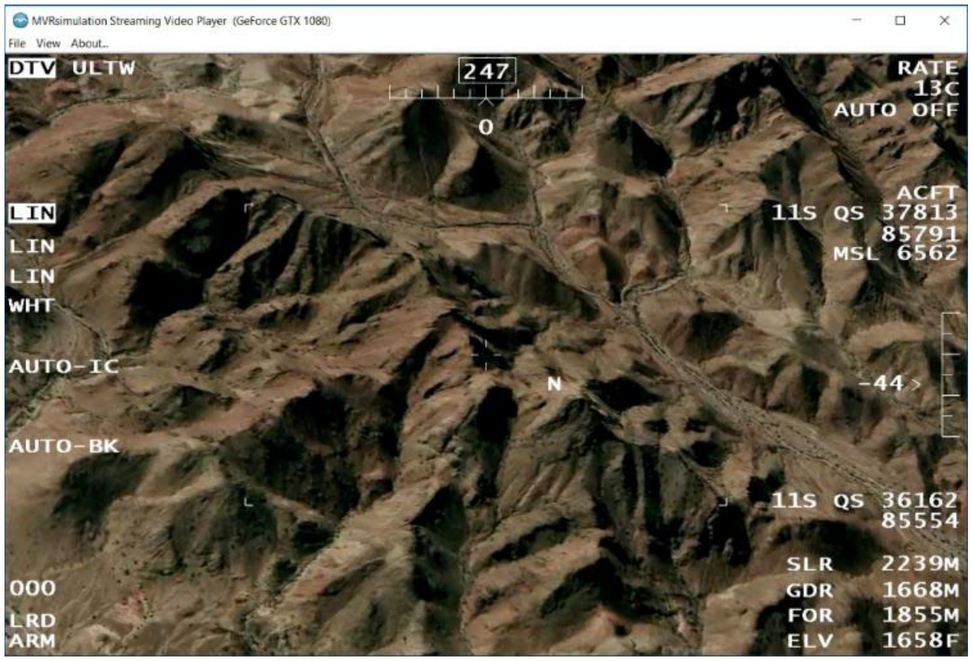
Using MVRsimulation's video player

MVRsimulation's video player is distributed with VRSG, with its own installer (Install-MVRsimulation-VideoPlayer-YYYYDDMM.exe), which is located on the distribution media in the \Utilities directory.

To install MVRsimulation's video player, run Install-MVRsimulation-VideoPlayer-YYYYDDMM .exe on the system on which you intend to run the video player. By default the video player will be installed in \MVRsimulation\Video Player.

To launch the MVRsimulation Video Player:

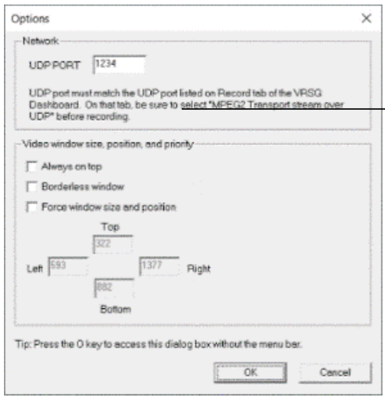
1. Either choose MVRsimulation Video Player from the Windows Start menu or double-click the Video Player.exe file.
2. The player begins by playing a pre-recorded test stream (test.mpg), until it detects a video on the network. (You can move the test.mpg out of the directory if you do not want the Video Player to play it when it starts.)



Prerecorded video, test.mpg, which plays by default (unless you move it out of the video player directory) until the player detects a video streaming on the network.

3. Choose File > Options. On the Options Dialog box select the UDP port to listen to for video from your VRSG system. Ensure this UDP port matches the UDP port you set on the Record Video tab on the VRSG Dashboard. When the video player detects a video stream, it resizes its window to match the dimensions of the video. You can override the window size as described next, if you want the video to appear to come from an underlying display, such as a simulated aircraft cockpit multi-function display (MFD).

4. On the Options dialog box, turn on settings to control the size, position, appearance, and priority of the video window:



Options

Network

UDP PORT 1234

UDP port must match the UDP port listed on Record tab of the VRSG Dashboard. On that tab, be sure to select "MPEG2 Transport stream over UDP" before recording.

Video window size, position, and priority

- [ ] Always on top
- [ ] Borderless window
- [ ] Force window size and position

Top 322, Left 593, Right 1377, Bottom 882

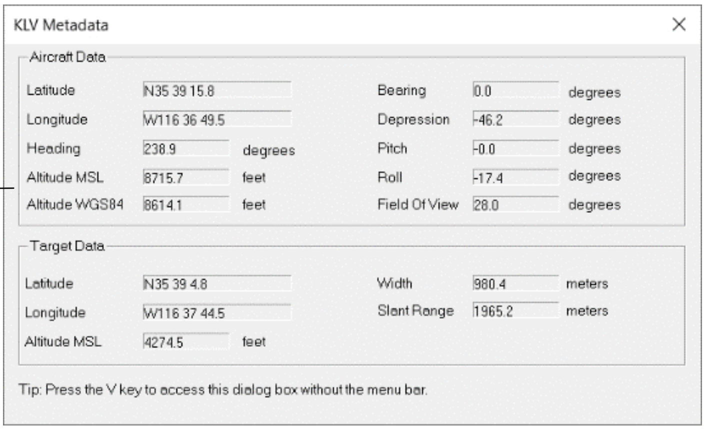
Tip: Press the O key to access this dialog box without the menu bar.

Click to activate the dimension fields, where you enter the intended dimensions of the video streaming window.

- Always on top – displays the video window above all other applications.
- Borderless window – displays the video window without borders, resize corners, or a menu bar.
- Force window size and position – the video window will use the provided screen-space rectangle to describe the position and size of the window. If the forced window size does not match the video dimensions, the window will be padded with black borders, either horizontally or vertically, and scaled to retain the proper aspect ratio of the video.

5. Choose View > KLV Metadata to display the STANAG 0601.1, 0601.9, or 0601.17 formatted metadata embedded in the video stream.

Decoded KLV metadata, which will be displayed if the KLV metadata option is selected on the Record Video tab of the VRSG Dashboard.



KLV Metadata

Aircraft Data

Latitude	N35 39 15.8		Bearing	0.0	degrees
Longitude	W116 36 49.5		Depression	-46.2	degrees
Heading	238.9	degrees	Pitch	-0.0	degrees
Altitude MSL	8715.7	feet	Roll	-17.4	degrees
Altitude WGS84	8614.1	feet	Field Of View	28.0	degrees

Target Data

Latitude	N35 39 4.8		Width	980.4	meters
Longitude	W116 37 44.5		Slant Range	1965.2	meters
Altitude MSL	4274.5	feet			

Tip: Press the V key to access this dialog box without the menu bar.

While the video is streaming, you can use the following keyboard shortcuts to display the dialog boxes without having to access the menu bar on the video window:

- ESC or F2 – to toggle between borderless and border (with menu bar) modes.
- O – to display the Options dialog box.
- V – to display the KLV metadata dialog box.

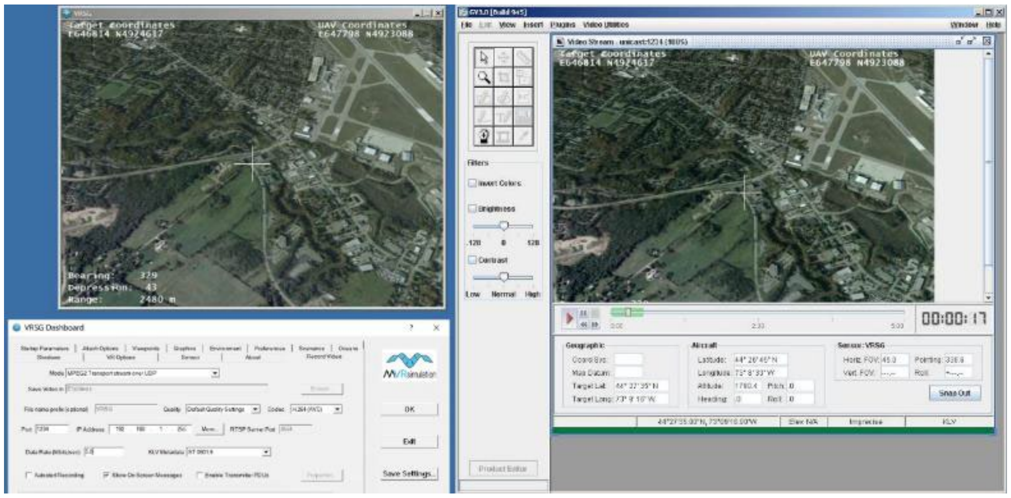
When the streaming session is finished, choose File > Exit to close the video player.

Using the GOTS GV 3.0 viewer

The GOTS package GV 3.0 viewer and metadata editor (available from www.pargovernment.com/gv3.0) can decode VRSG's MPEG stream and embedded KLV metadata.

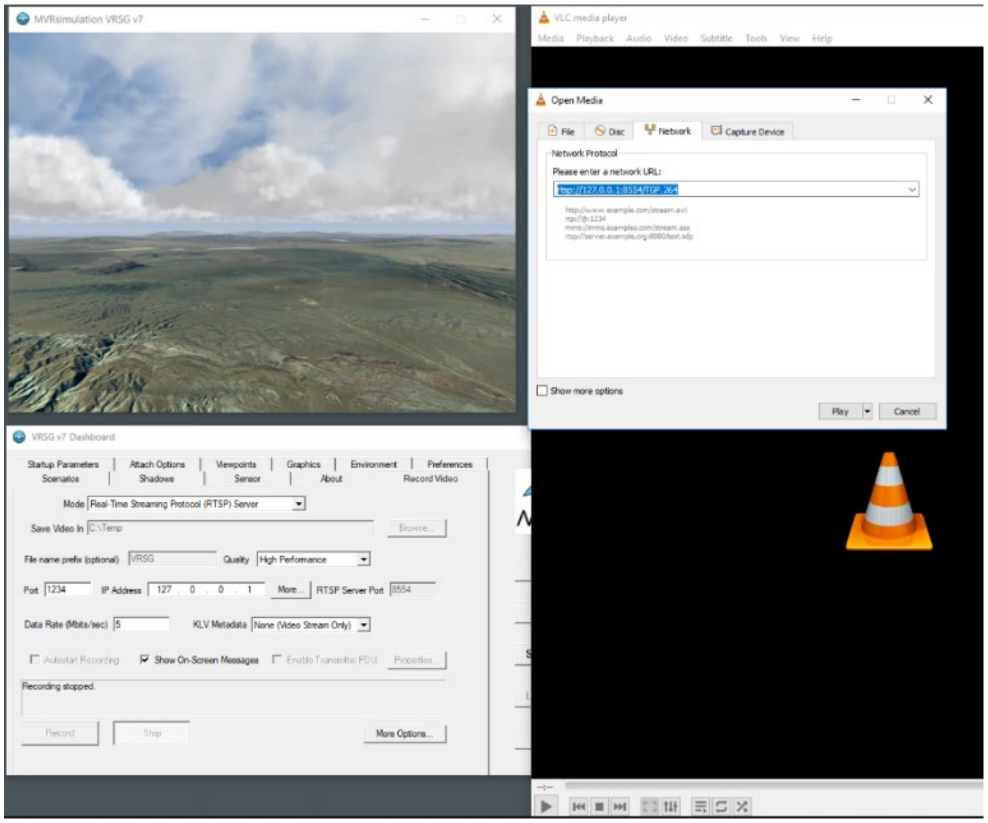
VRSG must be recording to the network before you attempt to open the stream in GV. GV will quickly time out if it cannot see a stream on the network. If GV is not decoding video correctly you might need to unselect its “Enable hardware rendering” preference setting.

The following image is a GV screen capture with the metadata fields extracted from the transport stream:



Using VLC

VLC requires a network URL syntax for entering the protocol and port upon which it will receive video. For the UDP URL, enter a string such as “udp://@:1234” where 1234 is the default port for video transmission on the Record Video tab.



Mode: Real-Time Streaming Protocol (RTSP) Server
Save Video In: C:\Temp
File name prefix (optional): VRSG
Quality: High Performance
Port: 1234
IP Address: 127 . 0 . 0 . 1
RTSP Server Port: 8554
Data Rate (Mbits/sec): 5
KLV Metadata: None (Video Stream Only)
[ ] Autostart Recording
[✓] Show On-Screen Messages
[ ] Enable Transmitter PDU
Recording stopped.

Network Protocol
Please enter a network URL:
rtsp://127.0.0.1:8554/TGP.264
http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
[ ] Show more options

When using VLC to view the streaming H.264 video, there might some degree of latency between when the VRSG screen is updated and when VLC is updated, which is normal for H.264/265 output. VLC has a large streaming buffer by default. To avoid latency issues, consider using MVRsimulation's video player instead. If you must use VLC, try the following to minimize the latency:

- On the Record Video tab, click the Advanced button, and then on the Advanced Options dialog box unselect the Enable B-Frames checkbox. B frames enable higher quality video for a given data rate, but incur additional latency. B frames should be used when recording to a file, where latency is not a concern.
- On the Graphics tab, click the More Options button, and then on the More Graphics Options dialog box, ensure the Vertical Retrace Sync option is selected.
- Use the latest version of VLC.
- In VLC, choose File > Open Network Stream, and then click the Network tab of the Open Media dialog box. Click the Show More Options checkbox to display the Caching option. Test this option to see what works to minimize the delay on your network. Latency can be reduced to fractions of a second running on the same machine via loopback.

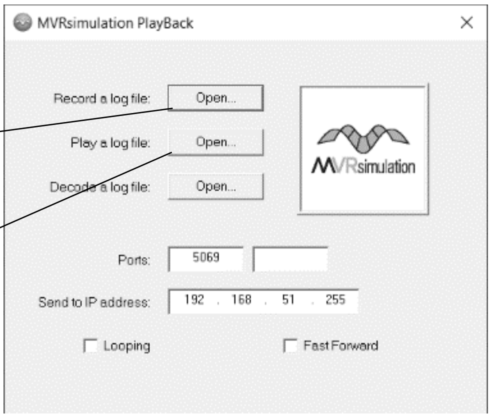
Recording and playing back a PDU log of network traffic

VRSG is delivered with a utility, PlayBack, which records and plays back traffic on the network via a PDU log. This utility is installed in

\MVRsimulation\Common\Util\Playback.exe. The PlayBack utility is useful for recording and playing back full DIS network exercises.

Click to specify a filename for the PDU log (default extension is .pdulog).

Click to locate the PDU log to play back, and optionally set the playback options.



Record a log file:

Play a log file:

Decode a log file:

Ports: 5069

Send to IP address: 192 . 168 . 51 . 255

Looping

Fast Forward

The port setting(s) must match the port(s) you use for DIS traffic.

The flight path of a Wasp Block III entity which was recorded and played back in VRSG.

While VRSG is running, this PDU recorder can record all network traffic and write the PDUs to a log file (vrsg.pdulog). To play back what was recorded, start VRSG, and then start PlayBack.exe, browse for the PDU log of interest and click Play.

In cases where you need to record both a DIS feed and CIGI to fully recreate your scenario, enter the ports of both protocols.

Two other methods of playing a PDU log are also available in ways that bypass the interface.

- Double-click a PDU log listed in the Windows File Explorer. The PDU log will play immediately in Playback.exe, broadcasting on the same port and address last used.
- Invoke PlayBack.exe from the command line instead of accessing the interface. In this case, the PDU log will play immediately and will broadcast on the same port and address last used.

These methods can be useful for playing back not only PDU logs of recorded DIS network exercises, but scenarios created in VRSG Scenario Editor, if you want to play back such a scenario without accessing the scenario from the VRSG Dashboard.

Note: When you play back a scenario's PDU log (vrsg.pdulog) that was exported from VRSG Scenario Editor, only the scripted entities will be sent over DIS. Be sure to add the exported vrsg.clt file to VRSG's search path so that the culture appears in the scene.

Taking screen captures of the rendered scene

You can capture a still image of the 3D scene displayed in the visualization window (in any display mode) and save it to the Windows Clipboard or to an image file in one of several popular formats. You can then use the image in any application that accepts image files. All formats generate full color RGB images, except for NITF, which generates 8-bit per pixel grayscale images.

To take a screen capture in VRSG:

1. Confirm the intended Screen Capture settings on the Preference tab, as described earlier in this chapter in the section, "Setting preferences."
2. In the visualization window, navigate to the area of which you want to capture a still image. Decide whether you want a full-screen image or smaller windowed mode image. The capture will have the dimensions of the displayed VRSG scene. (In the case of a multichannel system, the capture will be of one channel.)
3. Press the "C" key on the keyboard. VRSG displays a message in the window indicating it has successfully created the screen capture. A still image of the area is copied to either the Windows Clipboard or to a file in a specified directory, depending on the Screen Capture settings on the Preference tab.
4. If the image is saved to the Clipboard after the capture, paste the image from the Clipboard into any application that accepts images, such as Microsoft PowerPoint or Word, or an image editing application.
5. If the image is saved to a file after the capture, they will be saved to the default \MVRsimulation\VRSG\Snapshots directory, unless a different directory is specified in the Screen Capture settings on the Preference tab.
6. If the image is saved to a file, VRSG also saves the viewpoint at the time of capture as well as the settings that were in effect at the time of the capture, both as metadata. If the image is saved in JPG format, VRSG will embed the viewpoint and settings metadata directly into the JPG image. Essentially, the image is saved as a visual viewpoint. You can simply drag the captured image from the Windows Explorer to the VRSG visualization window to restore the viewpoint of the captured scene. As you drag the image to the VRSG scene, you are prompted to restore all Dashboard settings (in addition to the viewpoint) that were in effect in the scene at the time the screen capture was taken.
7. If the image is saved in any other supported image format (BMP, NITF 2.1, PGM, PNG, or DDS), the corresponding viewpoint an settings metadata files are written to the same directory as the image file, using the same filename, with the extensions .viewpoint and .json, as shown in the following example:

```
\MVRsimulation\VRSG\Snapshots\snap-kismayo-004.bmp
\MVRsimulation\VRSG\Snapshots\snap-kismayo-004.viewpoint
\MVRsimulation\VRSG\Snapshots\snap-kismayo-004.json
```

You can return to the exact location on the terrain depicted in the saved screen capture by dragging the saved .JPG image file, or the viewpoint file of an image saved in another format, from the Windows File Explorer and dropping it on the VRSG visualization window. Doing so moves the eyepoint to the location of the screen capture. Once you use this viewpoint, it becomes persistent; this viewpoint is saved to the vrsg.viewpoint file.

Still images of a scene captured in VRSG can be important for:

- After Action Review (AAR) presentations of simulation experiments. You can place the virtual world images in a document for an annotated presentation of a simulation exercise.
- Shared whiteboard applications.
- Communication with MVRsimulation support staff. In certain cases, you could show a support representative an image that illustrates a question or issue you have.

MVRsimulation uses screen captures of virtual worlds rendered in VRSG in documents such as conference papers, marketing literature, advertisements, and product user's guides. This feature is also used to create the images for the MVRsimulation website.

Using the laser range finder

VRSG has a built-in laser range finder. This feature is typically invoked by simulation applications using CIGI. However, you can use the engineering-level control from the keyboard.

To use the laser range finder, do one of the following:

Press the left mouse button. The coordinates of the lased point under the mouse cursor are displayed in the upper-left corner of the visualization window.

Press the letter "x" key on the keyboard, and then press the letter "l" key (that's l as in "laser" not the number 1). First, a 2D-overlay image of a crosshair is displayed. The intersection of the vertical and horizontal crosshair is the location where the range will be determined. When you press the keyboard "l" key, VRSG computes the linear distance from the crosshair intersection to the first polygon intersection and returns the range in meters on the bottom of the display.

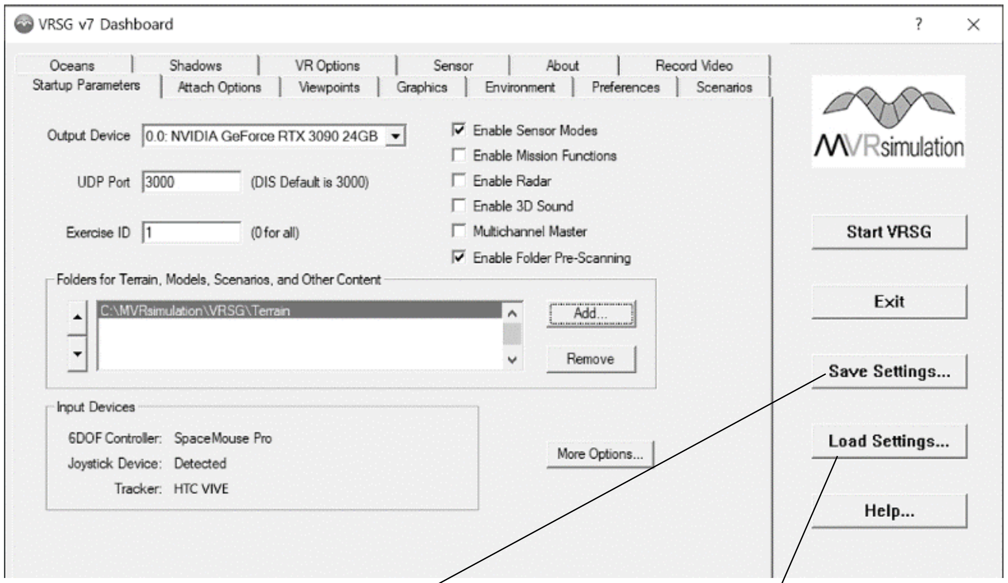
As shown in the next example, the azimuth and elevation are shown on top of the display. The azimuth appears first with an asterisk over the 2D AZ text indicating that the displayed value is for the azimuth. The azimuth then is replaced by the elevation with an asterisk over the 2D EL text. All readings are from the centroid of the initiating entity. Press the "x" key to turn off the laser range finder. Similar to the mouse-invoked lase, the coordinates of the lased point are displayed in the upper-left corner of the visualization window.

Coordinates of the lased point.

Saving settings for a particular terrain configuration

You can save settings specific to a particular configuration in a unique JSON settings file (*.json). Doing so can be useful if your site uses several terrains for training, and you want an easy way to access the settings of previous VRSG sessions. A settings file saves most the parameters that were set in the Dashboard, such as the search paths, network and environment settings, and preferences.

On the VRSG Dashboard, click Save Settings to save the configuration of your current VRSG session. When the Save as dialog box appears, give the file a name and save it to \MVRsimulation\VRSG\Settings, which is also the location of your default configuration.



Output Device: 0.0: NVIDIA GeForce RTX 3090 24GB
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

[✓] Enable Sensor Modes
[ ] Enable Mission Functions
[ ] Enable Radar
[ ] Enable 3D Sound
[ ] Multichannel Master
[✓] Enable Folder Pre-Scanning

C:\MVRsimulation\VRSG\Terrain

6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: HTC VIVE

Click to save the VRSG session configuration to a settings file (.json).

Click to use a specific settings file for your current VRSG session.

To use a settings file, click Load Settings and browse for a saved JSON file of interest. In this way, you could have several settings files to use for loading terrain tiles of different regions of the world.

Error logging and information reporting

VRSG creates two informational log files in the \MVRsimulation\VRSG directory: VrsgInfo_*.txt and VrsgError_*.txt. VRSG appends the filename with the name of the machine running the session, such as VrsgInfo_Austin1.txt. The files contain the following information:

VrsgInfo_*.txt contains information about hardware capability and what VRSG loaded in the last session. (This information can help to identify the ModelMap.ini and cultural feature files used in the last session.)

VrsgError.txt contains information about problems that VRSG encounters, such as textures that are missing from the search path.

The contents of both files are refreshed for each session of VRSG.

If any errors occur while you run VRSG, VRSG asks whether you want to view the "error log" upon exiting from the program. If you choose to view it, VRSG displays the VrsgError.txt file in your default text editor. For example:

```
Failed to find texture file dirt_road_desert.rgb
Failed to find texture file concrete_surface_1_large.rgb
```

If something unusual occurs in a VRSG session after you have started it in Desktop Cover mode, and no error message appears, try restarting VRSG in Sizeable Window mode. Often error messages are hidden when VRSG is running in Desktop Cover mode.

Note: If you are running VRSG on a system that has User Access Control (UAC) activated, the VrsgInfo.txt and VrsgError.txt files will not be saved or updated in the VRSG installation directory if the \MVRsimulation\VRSG directory is located in the C:\Program Files path. Instead the updated file will appear in the directory C:\Users\<username>\AppData\Local\VirtualStore\Program Files\MVRsimulation\VRSG.

Using multiple viewports

A viewport is a scene rendered in a VRSG channel. Multiple viewports are multiple scenes that are rendered from a single VRSG channel (instance). Viewports enable you to divide the framebuffer into subset rectangles that can receive/render different scenes. Multiple viewports is the recommended way to achieve multiple concurrent scenes on the same system.

Use of multiple viewports requires a simulation host capable of using this feature of VRSG. Viewport limits are associated with the license for a given VRSG channel and are listed on the Dashboard's About tab.

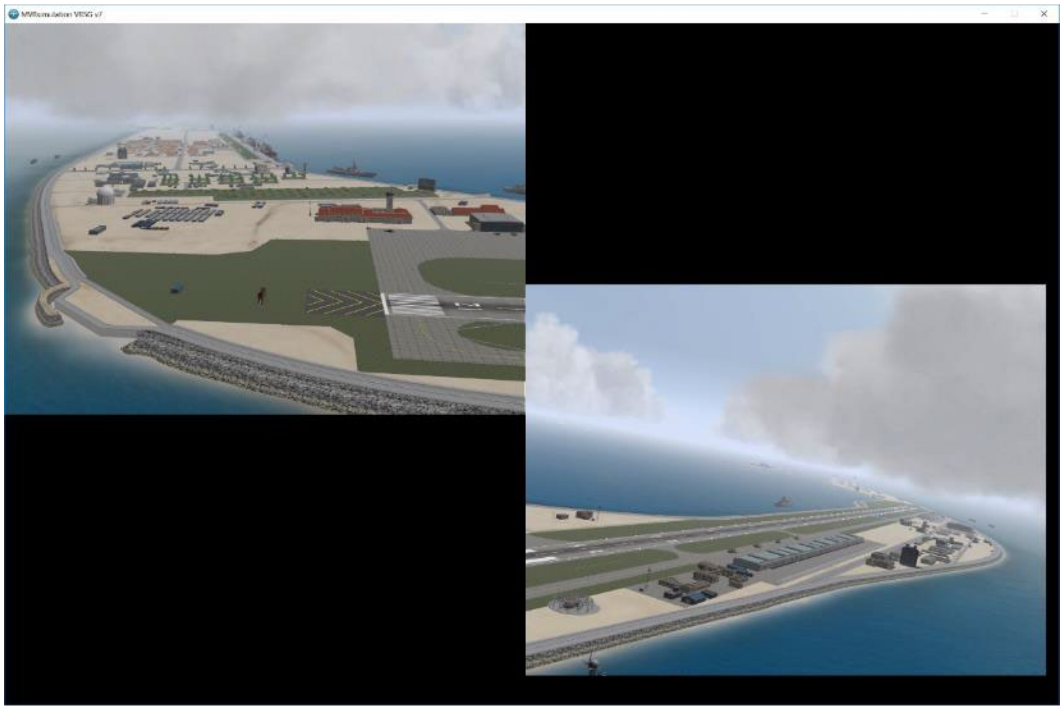
Applications with multiple viewports from one VRSG instance include:

- Picture-in-picture or picture-by-picture arrangements.
- UAS simulation, where often the nose camera and the articulated sensor camera are each given a dedicated viewport.
- VR HMD devices, which require at least two viewports (one for each eye) or four viewports (two for each eye), depending on the manufacturer.
- Two or more projectors on a dome display; one viewport could be allocated to a skyward scene, and the other to a denser terrain scene, as a way of achieving some load balancing.

- Multiple monitors of the same make and model connected with the Nvidia Surround feature into a single logical extra wide display. VRSG sees the single-wide desktop as a single-wide framebuffer. Viewports can be created at the monitor boundaries such that each monitor receives a different scene. This setup is a popular use of multiple viewports.

Multiple viewports on a single VRSG channel can be overlapping or spatially separated. The viewports can also be horizontally mirrored to support applications that demand this (such as a rear-view mirror), or display systems whose optics impose a horizontal reversal of the image.

Multiple viewports work best when the camera origins are on the same entity. For example you could have one viewport allocated to a nose camera on a UAV and another one allocated to an articulating sensor. (Using CIGI, you could attach viewports to different entities, as long as they were geographically close.)



Note: Creating multiple viewports does *not* add more graphics processing power to the system. They divide up the existing processing bandwidth across multiple scenes; the maximum possible scene content density is reduced with multiple viewports. Note this places significant additional demand on the system.

To use more than one viewport with a given VRSG channel, you create a Viewports.txt file and place it directly in \MVRsimulation\VRSG\ with one entry per line for each viewport. Each entry directs VRSG where to draw each viewport on the screen, the field of view, and the visual spectrum.

The syntax for an entry is:

```
x, y, width, height, lfov, rfov, tfov, bfov, yaw, pitch, roll,
spectrum
```

The spectrum options are:

otw – out the window	irwh – ir white hot
Eo – electro-optic	nvg ir – white hot
irbh – ir black hot	

If no spectrum option is given, the display defaults to out-the-window (otw).

For example, these entries in a Viewports.txt file render the two-viewport VRSG display example on the previous page:

```
0      0 1024 768 25 25 20 20 0 0 0 otw
1024 512 1024 768 25 25 20 20 75 0 0 otw
```

The meaning and order of the values for the entries in the Viewports.txt file, starting from the left, are:

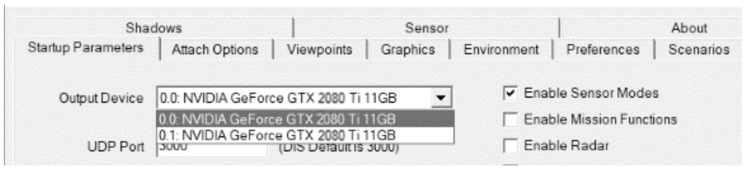
- The 1st pair of values are the starting position of the viewport, in pixels.
- The 2nd pair of values are the width and height of the viewport, in pixels.
- The 3rd pair of values are the left and right angles defining the perspective projection matrix, in degrees.
- The 4th pair of values are the top and bottom angles defining the perspective projection matrix, in degrees.
- The next values define yaw, pitch, and roll angles for the view direction of the viewport, in degrees.
- The final value defines the spectrum type, which in this case out-the-window (otw).

Multiple viewports are supported by applications using CIGI, not using DIS-based (or legacy MUSE) interfaces. For information about how to create multiple viewports for a CIGI application, see the appendix “CIGI Version 4.0 Support.”

Note: Due to the increased demand on your system, if you experience performance degradation using multiple viewports, you may have reached a point where you are asking for more bandwidth than your system can deliver. Try turning off any resource intensive features you do not need, such as volumetric clouds and shadows. If your scenario is totally land-based, disable 3D oceans as well.

Using VRSG on multiple monitors

VRSG supports the use of multiple monitors in one system. The additional output devices will appear as options on the Startup Parameters tab of VRSG's Dashboard. The following example shows the output devices of a machine with two monitors (and one graphics card):



Output Device 0.0: NVIDIA GeForce GTX 2080 Ti 11GB
0.0: NVIDIA GeForce GTX 2080 Ti 11GB
0.1: NVIDIA GeForce GTX 2080 Ti 11GB
UDP Port 3000 (DIS Default is 3000)

✓ Enable Sensor Modes
☐ Enable Mission Functions
☐ Enable Radar

The first number is the video card index and the second is monitor index. Following the name of the video card is the amount of memory on the card.

VRSG always displays the Dashboard on the first/primary monitor. If you select another monitor as the output device, VRSG displays the visualization window on the selected monitor, and VRSG retains that monitor preference. You can use Desktop Cover to display the visualization window on a second or third monitor to enable other applications to obtain focus while VRSG is still running.

Because the visualization window on the second or third monitor has no options, you cannot move, resize, or minimize it. By using this extended desktop mode in a training setup, an instructor could manipulate the controls on the Dashboard displayed on one monitor without disturbing the rendered imagery in the visualization window displayed on another monitor. All monitors must be turned on before you start VRSG as it populates the device list with available devices upon startup.

3D Vision Surround from Nvidia enables the expansion of VRSG onto two, three, or four displays. In this case there is no second or third output device. VRSG views the desktop as a single large logical monitor. If VRSG started in full-screen mode, the 3D scene will span all monitors.

VRSG support for multiple projector displays on arbitrary-shaped surfaces

VRSG has multiple solutions for distortion correction and edge blending for multiple-projector displays on non-planar display surfaces, such as the curved surfaces of hemispherical domes or cylindrical displays. VRSG is delivered with an MVRsimulation plugin for making simple manual adjustments to correct projected imagery on a curved surface. Also delivered are plugins for using VRSG with third-party warp/blend systems of Scalable Display Technologies and VIOSO Projection. These plugins are installed in \MVRsimulation\VRSG\Plugins\Displays. For more information, see the chapter "Distortion Correction and Edge Blending."

CHAPTER 3

Running VRSG With Synchronized Channels

VRSG support for synchronized channels with or without a simulation host, enables a set of multiple VRSG channels (instances or computers) coupled in such a way as to create a multichannel image generator.

Training simulators with this kind of setup typically have multiple displays (such as domes, monitors, VR headsets, or emulated military equipment). Each display is driven by a different VRSG channel, and in some cases a simulation host controls all the channels to provide a coordinated multiple display environment.

U.S. Navy Combined Arms Virtual Environment (CAVE) training dome with multiple synchronized VRSG channels at the Expeditionary Warfare Training Group Pacific (EWTGPAC) in San Diego, CA.

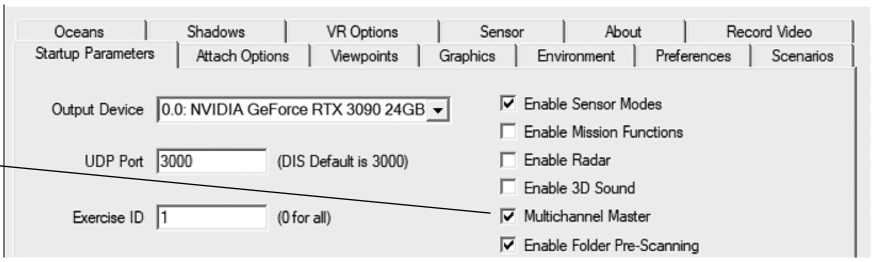
Use the VRSG MultiChannel Master option on the Startup Parameters tab of the VRSG Dashboard and the settings on the Client Views tab when you want to create a multi-channel stealth system not controlled by a simulation host.

VRSG *MultiChannel* acts as the simulation host, controlling multiple VRSG systems as a single synchronized multi-channel system; in this configuration. This setup is useful if you want to have a “stealth” view, which is controlled by a 6DOF input device.

If you intend to control VRSG channels from your simulation host, for example through CIGI, do *not* use VRSG MultiChannel. This chapter is intended for users who want to set up a multi-channel stealth system using VRSG MultiChannel.

To use the VRSG in a multichannel setup without a simulation host, start VRSG and select the Multichannel Master option on the Dashboard’s Startup Parameters tab as shown below.

Select this checkbox to enable VRSG Multichannel and access to the Client Views tab on the VRSG Dashboard.



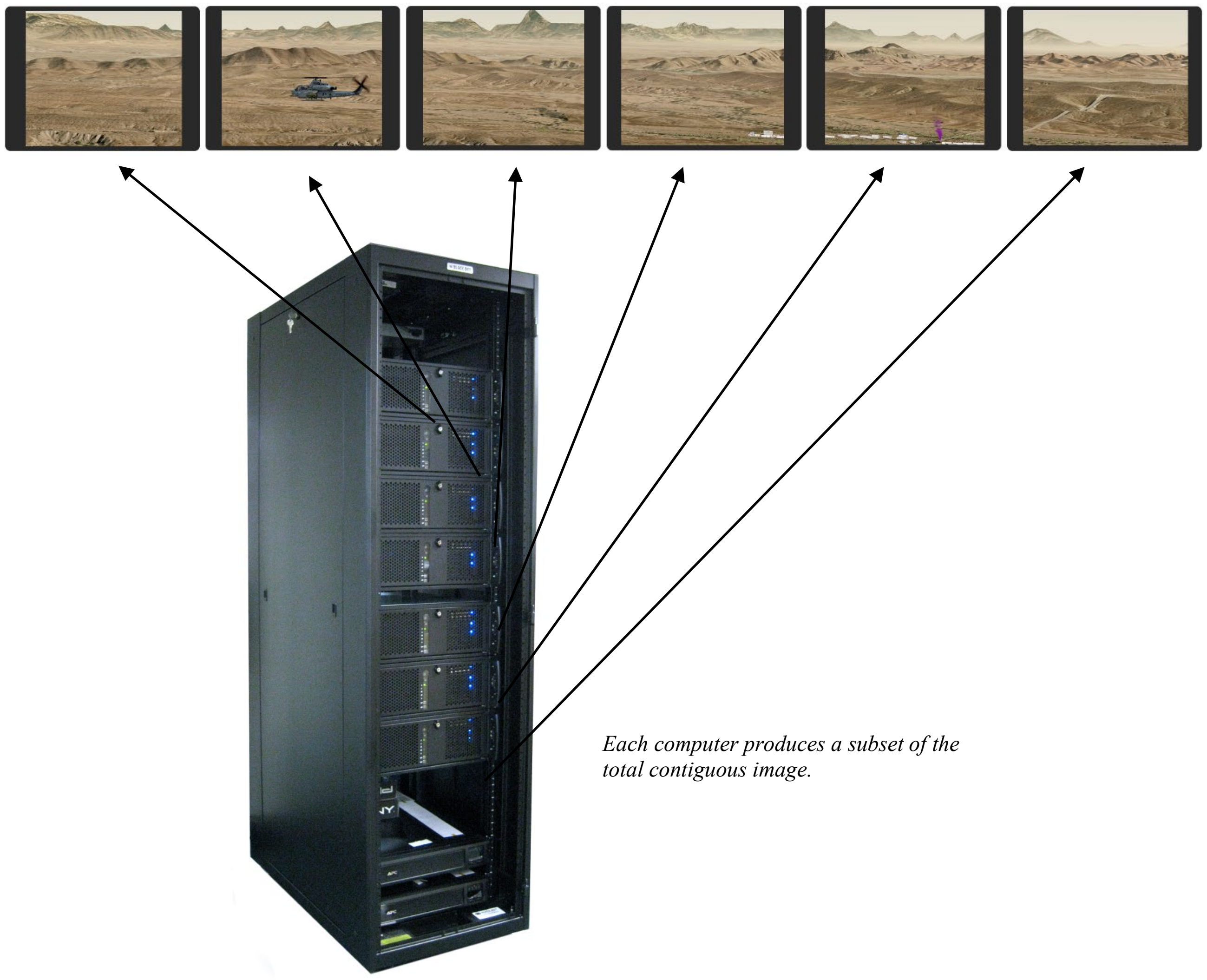
Output Device: 0.0: NVIDIA GeForce RTX 3090 24GB
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

[✓] Enable Sensor Modes
[ ] Enable Mission Functions
[ ] Enable Radar
[ ] Enable 3D Sound
[✓] Multichannel Master
[✓] Enable Folder Pre-Scanning

There are three primary configurations for which you can use VRSG’s multichannel synchronization:

- As a single-interface machine that communicates with client views using the same physical network as the simulation network.
- As a multiple-interface machine acting as a bridge between the simulation network and the private visual channel network. Each client channel has a single network interface. This represents the typical configuration.
- As a multiple-interface machine in which the client views are also multiple-interface machines. In this configuration, the clients are physically connected to the simulation network and are also physically connected to the private visual channel network.

The image on the next page shows how, by using multiple VRSG channels, multiple computers can display a panoramic view. Each computer displays a single image. The top channel is a simulation host or a VRSG Multichannel master channel. Side by side, the displays on the monitors create a panoramic view.


Each computer produces a subset of the total contiguous image.

How multichannel synchronization works

When VRSG is run with multiple channels, the channels are controlled by either a simulation host (such as a flight simulator) or by VRSG MultiChannel. VRSG MultiChannel contains the Client Views tab on the Dashboard that allows you to configure the client channels. Client channels must be configured to specify their view attachment position, view orientation, and field-of-view angles. When your simulation host controls VRSG channels, you do not use VRSG MultiChannel. Instead, the CIGI host controls the channels contains the mechanisms to specify the client channel's attachment position, orientation, and field-of-view.

MVRsimulation, BSI, and IDSI's Fallon Familiarization Solution, featuring three VRSG channels.

Once you start VRSG, it can either be controlled by a master channel or operate as an independent stealth view. Once you click the VRSG Start button on any client to launch VRSG in visualization mode, the application becomes an independent stealth and cannot be controlled by a simulation host or VRSG MultiChannel. When VRSG MultiChannel or your simulation host controls a channel, that channel is started automatically by the controlling system. You only need to launch the VRSG Dashboard, you do not click Start VRSG to launch all of the other channels into visualization mode.

The master channel controls the client views though messages exchanged over a local area network (LAN). This network could be the same network used for simulation traffic, but this setup is not recommended because of the large amount of bandwidth needed for the master channel and client views to interact. The master channel provides a means in which the viewport coordination traffic may be isolated onto a private LAN. In a typical configuration, the master channel has two network cards. The client views are physically connected to a private LAN shared with the VRSG master channel. The master channel uses a second network card to connect to the simulation LAN. In this configuration, the master channel machine forwards simulation network messages onto the private LAN so that the client views may also see the simulation network messages.

You could think of the master channel as a simulation host, and the VRSG systems controlled by the master channel as the multi-channel image generator for the simulation host. The master channel provides all the dynamics computations that drive the location and orientation of the logical platform onto which the client views are attached. In addition, the master channel provides the center view located at the origin of the platform. As an alternative to using VRSG MultiChannel for controlling the client views, the client views may be controlled by any application using Common Image Generator Interface (CIGI). In this setup, the master channel is not needed because the simulation host application (e.g. rotary-wing aircraft simulator, M1 tank simulator) computes the dynamics for the location and orientation of the platform onto which the client views are attached. Like the master channel computer, a simulation host computer should have two network cards to isolate the CIGI messages from the simulation network.

Setting up a synchronized multi-channel system

Setting up a synchronized multi-channel VRSG system includes the following tasks:

- Configuring the TCP/IP settings of the private and DIS networks.
- Assigning site/host/entity triplets to each client view and the master view.
- Adding and configuring the client views to the master channel.
- Disabling flow control on the network card and/or the network switch.
- Run 3D benchmarking tests for each client.

The rest of this chapter describes each task in detail.

For using multiple projectors on a single display surface, see the chapter "Distortion Correction and Edge Blending with VRSG" for information about using warping/blending plugins that are delivered with VRSG.

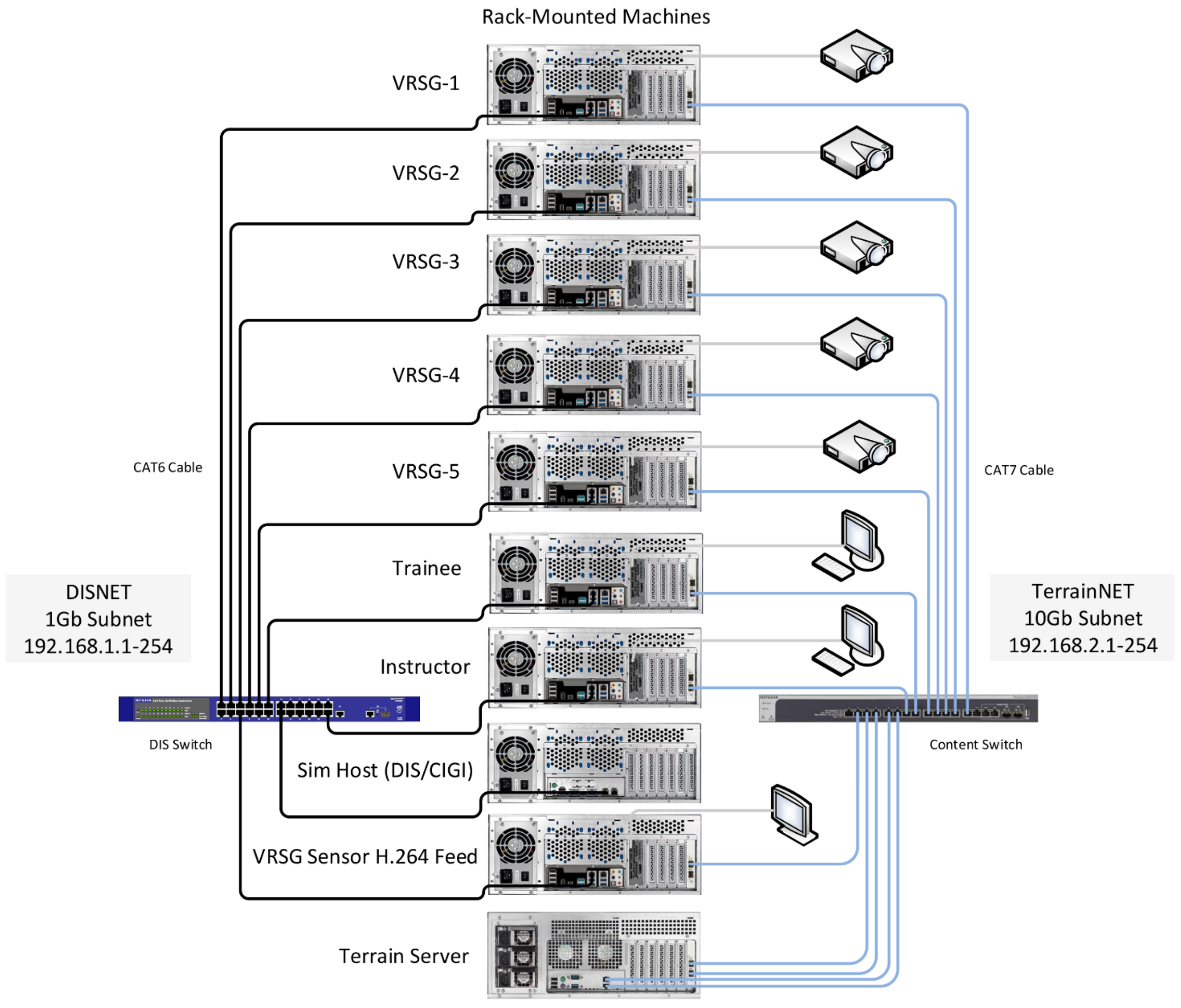
U.S. Navy CAVE training dome with synchronized VRSG channels at the EWTGPAC in San Diego, CA.

Setting up the network

See the Windows documentation for information about how to specify IP addresses and network masks for the private multi-channel network and the simulation network. Make a note of the network broadcast addresses for the simulation LAN and the multi-channel LAN, because you will need these addresses to complete the system configuration.

The following diagram illustrates MVRsimulation's recommended multi-channel simulation network configuration. In this configuration, which has evolved as our best practice for a dome or classroom setup, DIS and CIGI broadcast resides on a separate network (DISNET in the diagram) from the Terrain Server where VRSG and the 3D terrain and model libraries reside (TerrainNET).

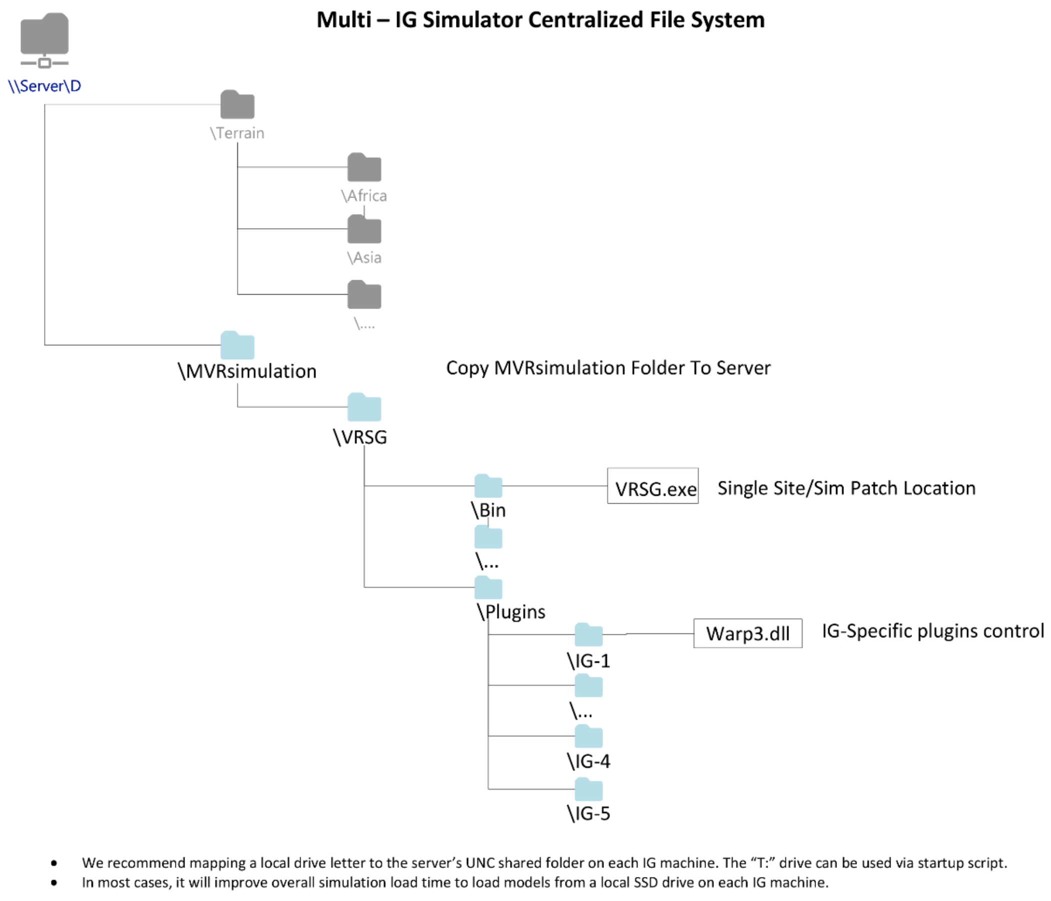
Recommended multi-channel simulation network configuration



The centralized Terrain Server is from which all the networked VRSG simulation computers (again, for example a multiple-projection or classroom setup) run VRSG and pull 3D content. Although each machine must have a VRSG installed and a VRSG dongle or software license present, the machines all run VRSG and pull 3D content from the centralized Terrain Server.

This setup keeps maintenance, VRSG patches/upgrades, plugins, and 3D content refreshes easy to manage at a single location. If the client computers have small C: drives, this setup eliminates the need for them to store MVRsimulation's large 3D model libraries.

The next diagram shows the recommended file setup on the Terrain Server.


Multi – IG Simulator Centralized File System
\\Server\D
\Terrain
\Africa
\Asia
\....
\MVRsimulation
Copy MVRsimulation Folder To Server
\VRSG
\Bin
VRSG.exe
Single Site/Sim Patch Location
\...
\Plugins
Warp3.dll
IG-Specific plugins control
\IG-1
\...
\IG-4
\IG-5

- We recommend mapping a local drive letter to the server's UNC shared folder on each IG machine. The "T:" drive can be used via startup script.
- In most cases, it will improve overall simulation load time to load models from a local SSD drive on each IG machine.

1. Copy the MVRsimulation installation folder to the Terrain Server.
2. Map a local drive letter, such as "T", to the Terrain Server's shared UNC folder on each VRSG IG machine. You can assign the local "T" drive letter to the UNC share by adding the command line "net use T:\\*terrainserver**sharename*" to the VRSG IG machine's startup batch file.
3. Optionally, specify VRSG plugins (such as Warp3.dll and H.264.dll shown in the diagram) to run only on certain machines. VRSG will first check for a folder named \Plugins_*hostname* before checking the \Plugins folder.

Note: In most cases, it will improve overall simulation load time to load models from a local SSD drive on each VRSG IG machine.

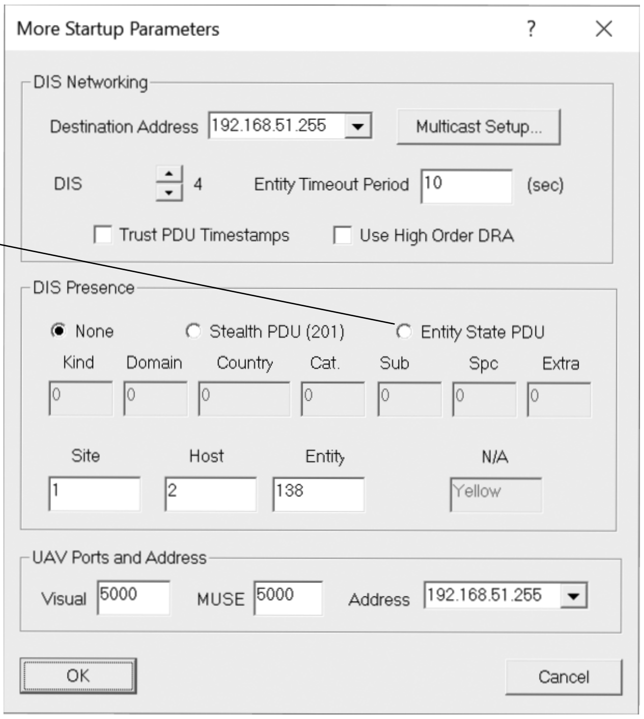
Assigning addresses to clients

For each client view and the master view you must assign a unique site/host/entity triplet to identify the view on the network. MVRsimulation recommends that you assign all client views the same site and host numbers, but assign unique entity numbers to each view.

To assign the site, host, and entity numbers:

1. On the Startup Parameters tab, click the Advanced button.
2. In the More Startup Parameters dialog box, enter the site, host, and entity numbers. Then click OK.

Do not turn on the Enity State PDU option for client views.



Destination Address: 192.168.51.255
DIS: 4
Entity Timeout Period: 10 (sec)
Trust PDU Timestamps: ☐
Use High Order DRA: ☐

[✓ None] Stealth PDU (201) Entity State PDU

Kind	Domain	Country	Cat.	Sub	Spc	Extra
0	0	0	0	0	0	0

Site	Host	Entity	N/A
1	2	138	Yellow

Visual: 5000 MUSE: 5000 Address: 192.168.51.255

Note: The client views should *not* have the Entity State PDU option selected. *Only* the master view, if any, should have this option selected.

555th Fighter Squadron F-16 Fighting Falcon pilot, operates an AFRL F-16 Deployable Tactical Trainer (DTT) simulator. The multi-channel synchronized view is driven by VRSG. Photo: U.S. Air Force photo by Senior Airman Matthew Lotz/Released.

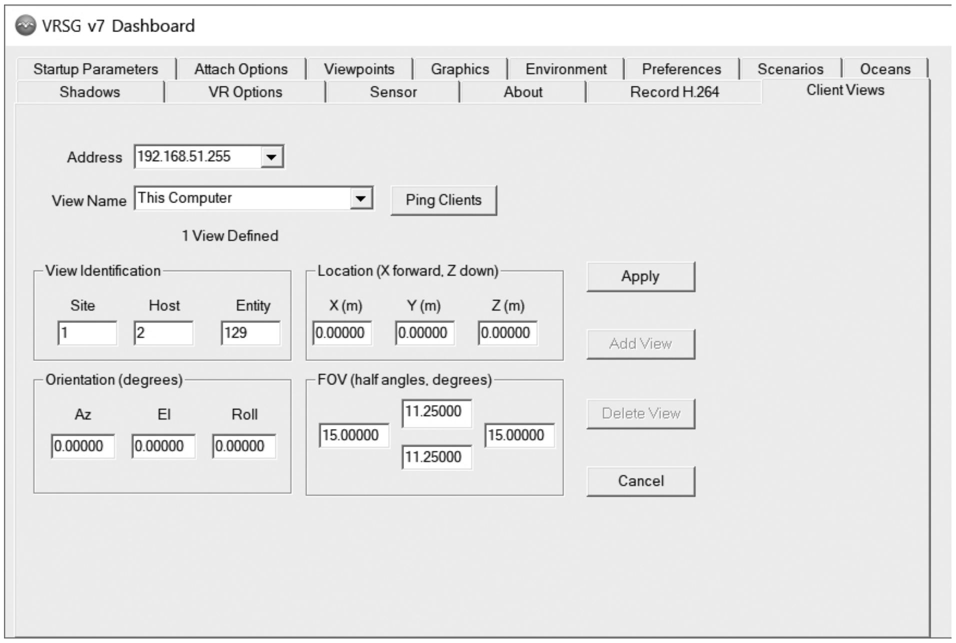
Configuring client views

You configure the client views in the Client Views tab of the VRSG Dashboard. This tab is displayed when you select the Multichannel Master checkbox on the Dashboard’s Startup Parameters tab.

The Client Views tab contains the controls that enable you to identify which VRSG clients will be controlled by the master channel and specifies the relative locations and orientations of the client views. If your simulation host will be controlling the client views, it will configure the client views through CIGI.

The view generated by the VRSG Multichannel Master channel is set up automatically and appears in the view list on the Client Views tab as “This Computer.” This view is typically used as the center channel, but you can modify this default behavior by selecting “This Computer” and making the appropriate modifications.

Select from the Address list the broadcast address of the network for the visual channels. The default broadcast address of each network interface will appear in this list. If you are using subnets, you must change the default broadcast accordingly.



Address 192.168.51.255
View Name This Computer
1 View Defined

View Identification: Site 1, Host 2, Entity 129
Location (X forward, Z down): X (m) 0.00000, Y (m) 0.00000, Z (m) 0.00000
Orientation (degrees): Az 0.00000, El 0.00000, Roll 0.00000
FOV (half angles, degrees): 11.25000, 15.00000, 15.00000, 11.25000

Editing views

To edit a view on the Client Views tab:

1. Select from the View Name drop-down list the view you want to modify. This list contains all configured views. When you select a view, all the editable attributes of the view appear in the appropriate fields on the tab.
2. Optionally, change the view name, its network identification, location relative to the center view, orientation relative to the center view, and the field-of-view.
3. Click the Apply button to have the changes take effect. (Click Cancel to discard your changes and revert the view back to its original settings.)

Adding views

To add a new view on the Client Views tab:

1. Click the Add View button. Initially the fields on this tab contain default values.
2. Enter the appropriate values for this new view:
   - Assign a name to the view in the View Name field so you may readily select it from the list at a later time if you need to modify or delete the view. An example of a view name would be "Far Left, -60 degrees."
   - In the Client Identification fields, specify the site, host, and entity numbers of the VRSG client you are assigning to this view. On the VRSG client, the site, host, and entity numbers are assigned in the Advanced Parameters settings of the Startup Parameters tab.
   - For Location specify the location of the eye point relative to the center view. The center view eye point is located at (0,0,0). The location of the client view is specified relative to the center view using the DIS Z-down entity coordinate system. In this

coordinate system, the positive X-axis is forward, the positive Y-axis to the right, and the positive Z- axis is down. Units are given in meters.

- For Orientation specify in degrees the orientation of the view relative to the center view. Positive azimuth is clockwise rotation relative to the center view. Positive elevation (Elev.) pitches the view upward relative to the center view.
- Specify the field-of-view (FOV) is specified as 4 potentially asymmetric positive half angles. For example, if you want a 40 degree horizontal (full-angle) by 30 degree vertical (full angle) FOV, you would enter 20 for the outer fields corresponding to the left and right FOVs, and 15 for the center fields corresponding to the top and bottom FOVs.

Removing views

To remove a view from the configuration, on the Client Views tab select the view from the View Name drop-down list and click the Remove button.

Testing the configuration

Once you have configured all of the views, and have assigned the correct site-host-entity numbers to the client VRSG views, you can test the configuration by clicking the Ping Clients button.

If the configuration is successful, VRSG displays the message “Reply received from all *N* clients,” where *N* is the number of views you have in your configuration.

If some or all clients do not respond, the message “Some or all clients failed to respond” is displayed. If this message appears, verify the following:

- All client channels have been correctly addressed. Verify the IP address and the network mask of all the client views. Consult your Windows documentation for more information.
- The Address field is set to the correct broadcast address of your private visual channel network.
- All client VRSGs have the correct unique site, host, and entity designations.

Configuration scenario

The following scenario illustrates how to configure a three-channel system. Note that the IP addresses used in the scenario are for illustration purposes only.

In this three-channel system:

- Channel 1 is the left view and has a single Ethernet card with the address 140.140.1.1. Since this is a class B network, the broadcast address is 140.140.255.255.
- Channel 2 is the center view and has two Ethernet cards. One Ethernet card is for the private visual channel network (140.140.255.255) and the other network card is for the DIS network. For this example, the address 192.55.242.2 is assigned to the center

  channel’s DIS interface. This is a class C network with a broadcast address 192.55.242.255. The visual channel network card is assigned the address 140.140.1.2.
- Channel 3 is the right view and has a single Ethernet card with the address of 140.140.1.3.

Channels 1 and 3 run the VRSG client software. In the Advanced Startup Parameters dialog box (which you access from the Startup Parameters tab), the site/host/entity is assigned 1/1/1 to channel 1 and 1/1/3 to channel 3. Channel 2 runs VRSG's multi-channel synchronization support as the master channel and is assigned site/host/entity values of 1/1/2. In the Advanced Startup Parameters dialog box, the center channel is assigned a broadcast address of 192.55.242.255, which enables it to send DIS messages onto the DIS network. On the Client Views tab, the Address is set to 140.140.255.255 to allow channel synchronization messages to be isolated on the private network.

Next, the views are set up on the Client Views tab. Click Add View to insert a new view into the configuration. In the View Name field type "Left View, -35 degrees." For location choose the default of (0,0,0). For azimuth enter -35 degrees. Keep the elevation at 0.0. For the field-of-view (FOV) enter 15 degrees for left and right, and 12 degrees for top and bottom. This yields a 30-degree horizontal FOV while providing a 5-degree gap between the center view and the left view to allow for the spacing between monitors. The vertical FOV is commensurately smaller to account for the aspect ratio of the display (1.25 being a typical aspect ratio). Click Apply to have the new view take effect.

Click Add View again to insert another view, the right view, into the configuration. In the View Name field, type "Right View, +35 degrees." For location keep the default of (0,0,0). For azimuth enter 35 degrees. Keep the elevation at 0.0. For the field-of-view (FOV) enter 15 degrees for left and right, and 12 degrees for top and bottom. Click Apply to have the new view take effect.

For the client views, start the application but *do not* click the Start VRSG button to enter visualization mode. When the master channel view starts, it will automatically start the client views.

Note: If you start a client view manually by clicking the Start VRSG button, that view becomes an independent "Stealth" and will not respond to viewpoint coordination commands from the center view.

Next, click the Ping Clients button to confirm a successful hardware and software setup. If the system reports that a successful reply was received from the two views, click Start VRSG on the master channel machine (view 2, center view) to start all three views.

Troubleshooting

If you encounter problems with the VRSG master channel starting the client channels:

- Verify the IP address on the VRSG Dashboard's Client Views tab is appropriate for your subnet. If you reconfigured the LAN, a stale address might be listed on the tab.
- Check the Site/Host/Entity IDs of each client view. Make sure those entries match what the other channels are set to under their Advanced Startup Parameters settings.
- Ensure the issue is not the Windows firewall. When VRSG is run for the first time on a machine, Windows prompts whether to unblock it. If someone at your site confirmed that Windows should block VRSG, it cannot communicate with client channels. Open the firewall settings in the Windows Control Panel and add the VRSG executable to the exceptions list.

Projection considerations for multi-channel configurations

In a perspective projection, the angular distance between pixels is non-uniform, independent of the display surface. The angular distance between the pixel at the center of projection and its neighbor is the smallest of the entire scene. Conversely at the edge of the display, the angle between the pixels at the edge and their neighbors are the largest. This is because the pixels are equidistant on the plane of projection, but form increasingly larger angles when a ray is shot between the camera and pixels. The degree of non-uniformity increases with larger FOVs. An image generator must first produce an image on a perspective projection, which is then resampled during a warp/blend phase to adjust for the non-planar display surface.

The input image can become under-sampled during the warp/blend phase, which introduces aliasing. More distortion is introduced in the input image with excessively wide FOVs, and it has an impact on any given IG's performance. Wider FOVs force VRSG to draw more content, such that the density of maximum allowed content would need to be scaled back commensurate with the FOVs. MVRsimulation recommends keeping VRSG's FOV to 90 degrees or less in these projection setups for curved displays or domes.

In an attempt to lower the cost of a dome simulator, one recently fielded pre-production solution has split the projectors into two frusta, one for the left half of the projector and another for the right half. In this case, each VRSG channel must render two scenes instead of one. Specifically, the IG renders two separate simultaneous windows; for example, rendering two 2048 x 2160 scenes instead of a single 4096 x 2160 scene. As result, the scenes have much smaller FOVs and better aspect ratio. The narrower FOV images have more uniform pixel spacing and less geometric distortion, resulting in an improved result of the warp/blend pass. This solution can help with distortion and resolution issues, however there is an overhead cost in having VRSG render two scenes per frame. Performance-wise there is an additional tradeoff in the overhead associated rendering two frusta, as VRSG must perform per frame two database traversals, two model culling passes, two cloud passes, and so on. Although the warping/blending result is improved, more workload has been shifted to the IG.

In addition, the FOV of the combined two frusta will be larger than the single one, so the total content VRSG must render per frame will increase. For example, in using eight (8) 4K projectors instead of 14 projectors in a 5-meter dome, the 4k projectors are nearly a 2:1 aspect ratio at 4096 x 2160. The projectors deliver the VRSG scene onto the dome with significant overlap, and therefore the wide aspect ratio requires the very large FOV. The warping and blending software must resample the huge FOV, where the image is undersampled in many places, leading to aliasing and loss of resolution. The performance impact of this approach of doubling up VRSG workload per channel needs to be evaluated on a case-by-case basis, with two more considerations. When a system is configured such that two views from one visual channel are generated, the load on VRSG is such that some features cannot be displayed simultaneously while maintaining the required 60 Hz frame rate.

Keep in mind that what might be acceptable in VRSG performance today may not be acceptable in the future with new VRSG features that require significantly more rendering capacity. To improve the realism of the warfighter's training experience and take advantage of industry improvements in graphics technology, MVRsimulation enhances VRSG with new features on an ongoing basis. As well, improvements are continuously made to the 3D content. As a rule, MVRsimulation develops all VRSG features and 3D content for 60 Hz performance and assumes a 60 degree FOV for a single render channel.

Disabling flow control

Flow control is a feature of a system's network card and/or network switch. Disabling flow control is crucial to ensure real-time delivery of data to the channels, and to avoid data buffering at the network switch level. If enabled, the switch will buffer data and not deliver it to the channels in real time. Flow control can typically be disabled at the PC end via the hardware properties of the network card. If your system is using a managed network switch, you can disable flow control via the switch's user interface to. This setting might need to be put into effect on a per-port basis.

Using test patterns delivered with VRSG

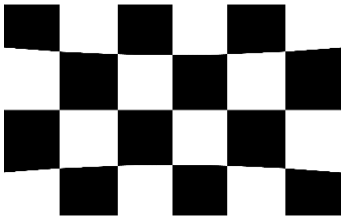
VRSG is delivered with two test pattern models that contain 1 meter wide black and white stripes. These models, resolution_horz.hpx and resolution_vert.hpx, are located in \Models\Other\. The host system can place them at various ranges to conduct the necessary measurements or subjective assessments.

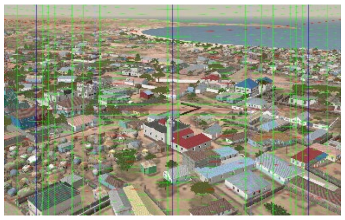
VRSG is also delivered with a plugin with a set of test patterns (using the abovementioned models) that can be used to work with misalignment of projectors in a display or to identify a field-of-view that does not match the display geometry.

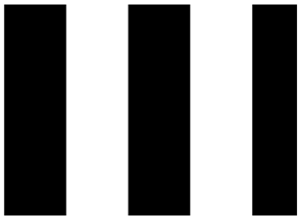
To activate the plugin, move the file TestPatterns.dll from the \MVRsimulation\VRSG\Plugins\Displays subdirectory so that it resides directly in the \Plugins directory, as in: \MVRsimulation\VRSG\Plugins\TestPatterns.dll.

To access the test patterns in VRSG, press F3. Doing so cycles through three patterns, which are displayed as full-screen overlays.

- The first is a black/white checkerboard pattern, where each square is 10 degrees by 10 degrees. This pattern is useful for measuring contrast.
- The second pattern is comprised of a grid of latitude and longitude lines displayed on the loaded terrain. Each line is 1 degree apart, creating a grid of 1 degree by 1 degree squares. This pattern is useful for verifying the warp/blend calibration in a curved display, such as in an immersive dome setup. If the calibrations are off, the lines would appear wavy in places.
- The third is a resolution pattern: of 3 vertical white bars on a black background. You can zoom in and out of this image by pressing the keyboard I and O keys; each keystroke moves the pattern 1% in or out from its current range. Move the pattern up/down and left/right with the arrow keys. Press the right-arrow to increment the azimuth, up-arrow to increment the elevation. Press the left-arrow to decrement the azimuth, down-arrow to decrement the elevation. Press the R key to roll the pattern in 90 degree increments.


Contrast test pattern

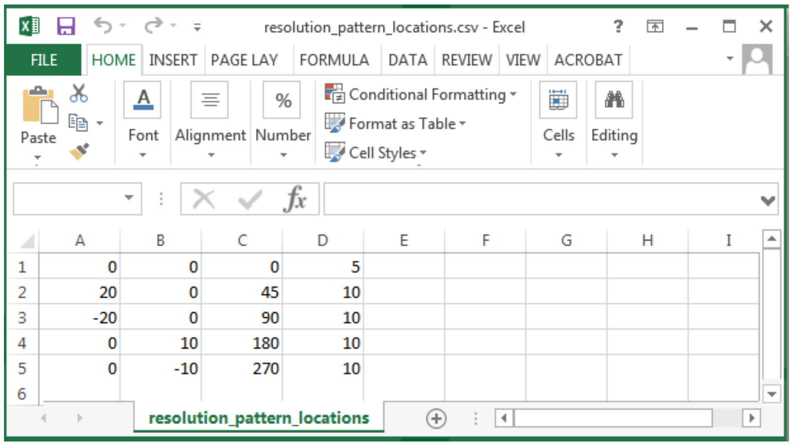

Warp/blend calibration test pattern


Resolution test pattern

Instead of manually moving the placement of the resolution test pattern, you can map the arrow keys to a set of predetermined azimuth-elevation-roll-range combinations in a .csv file and then press the arrow keys to cycle display of the resolution test pattern among the specified locations. In the .csv file, denote the azimuth, elevation, and roll in degrees, and range in meters.

(Note that the built-in test pattern object is 3 meters by 3 meters, which means if you placed it 3000 meters away, it will subtend 1 milliradian on the display.)

Name the .csv file as “resolution_pattern_locations.csv” and place it directly in the VRSG installation directory (\MVRsimulation\VRSG).



	A	B	C	D
1	0	0	0	5
2	20	0	45	10
3	-20	0	90	10
4	0	10	180	10
5	0	-10	270	10
6				

Once the .csv file is in place, press the arrow keys to cycle through these predetermined locations. If the .csv file is removed from the VRSG directory, manual placement of the resolution test pattern is again active. The test pattern plugin reads .csv file every time you press F3 to access the resolution test pattern, which mean you can edit the .csv file to make adjustments without having to restart VRSG. In a shared file system environment, you need to only edit one instance of the .csv file, and it can be done using a system that is not one of the channels.

Running 3D benchmarking tests

When you first install VRSG on one or more of your visual channels, consider running a 3D benchmarking tool to examine the effectiveness and performance of the graphics card on the machine when it runs 3D graphics applications. Consider running these benchmark tests twice and compare the scores. You should confirm that your system has all working visual channels independent of VRSG. If you see significant variation in the two scores, run two more tests and see whether the scores converge. If they do not converge, you might have a problem with the 3D graphics card on one or more channels.

A baseline report can be useful to compare with a later report if you need to troubleshoot VRSG performance problems on your system. This report is also useful for comparing performance benchmarks after installing a new channel or a new graphics card.

You should rerun benchmarking tests periodically to ensure your multi-channel system is performing optimally. Although test results might vary slightly, a large decrease in test results indicates degradation in some part of your 3D system. The performance of an MVRsimulation product scales in direct proportion to this score; the higher the result, the better this product will perform on your system. For more information about 3D benchmarking, see the *MVRsimulation Product Installation Guide*.

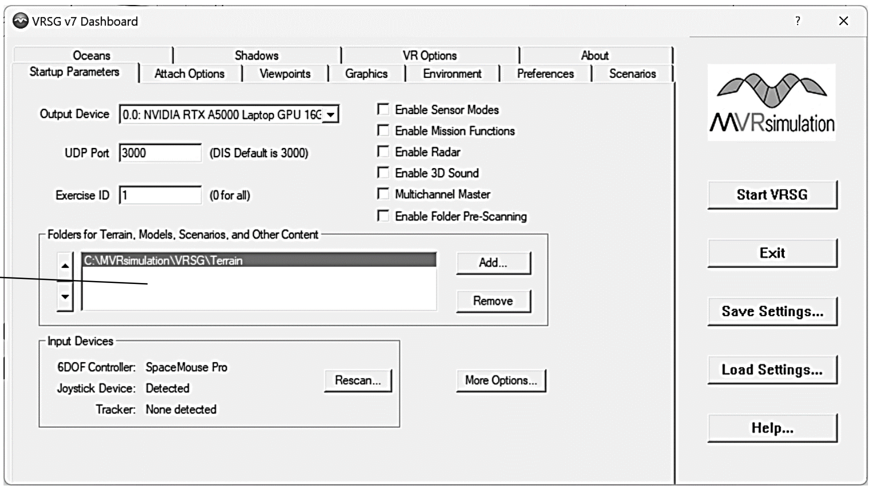
CHAPTER 4

Loading Content into VRSG

VRSG is a geospecific render engine that allows the user direct control over the scope of the 3D and visual content needed for a training session. VRSG uses a search path that is setup on the Dashboard to look for content to load into the visualization window.

This chapter provides information on terrain to help users properly load 3D terrain databases and other related content in VRSG. Review the information and follow the steps in this chapter to load the terrain, models, scenarios, and viewpoints into VRSG.

The set of directory paths VRSG is instructed to load is called its search path.



Output Device 0.0: NVIDIA RTX A5000 Laptop GPU 16G

UDP Port 3000 (DIS Default is 3000)

Exercise ID 1 (0 for all)

☐ Enable Sensor Modes
☐ Enable Mission Functions
☐ Enable Radar
☐ Enable 3D Sound
☐ Multichannel Master
☐ Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content

C:\MVRsimulation\VRSG\Terrain

Input Devices

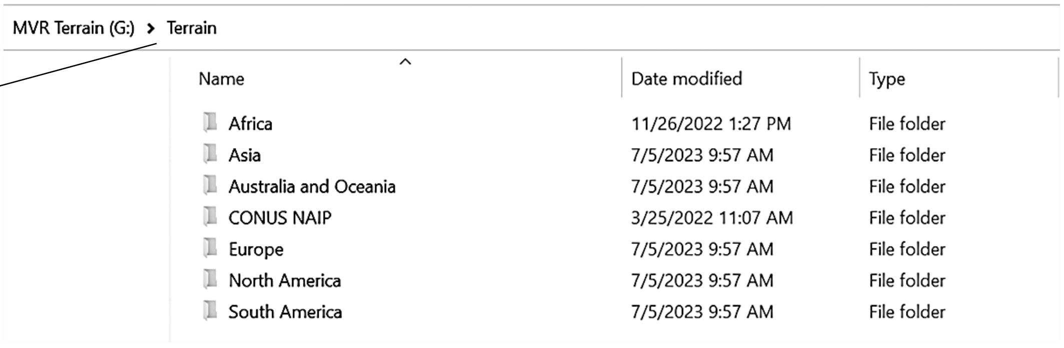
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: None detected

Overview of terrain data organization and storage

Terrain data is organized by region and distributed either on an external drive or as part of a collection encompassing all or portions of the MVRsimulation World Terrain Database. This data can be accessed by using an MVRsimulation regional terrain drive, direct attached large volume (DALV) device, network attached storage (NAS) server, or World Terrain Database server. The CONUS and OCONUS terrain are provided in the CONUS NAIP directory, which is sorted for distribution by geographical region. The rest of the world is distributed by continent and sorted by country.

The complete collection of MVRsimulation's VRSG terrain is saved in regional directories:

Directory that contains the entire collection of MVRsimulation terrain databases.

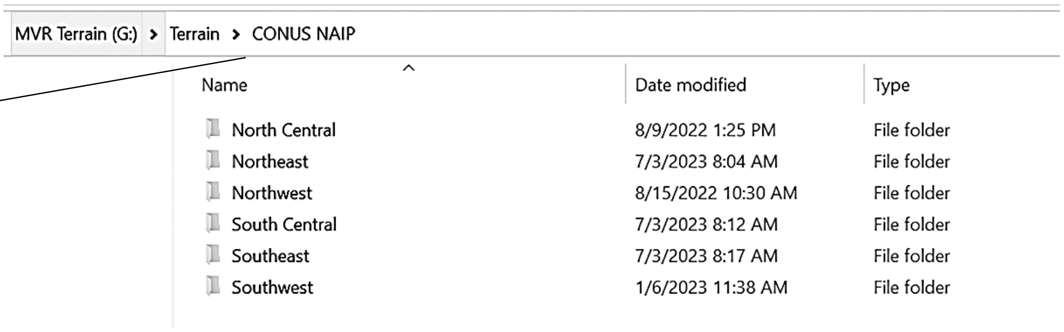


MVR Terrain (G:) > Terrain

Name	Date modified	Type
Africa	11/26/2022 1:27 PM	File folder
Asia	7/5/2023 9:57 AM	File folder
Australia and Oceania	7/5/2023 9:57 AM	File folder
CONUS NAIP	3/25/2022 11:07 AM	File folder
Europe	7/5/2023 9:57 AM	File folder
North America	7/5/2023 9:57 AM	File folder
South America	7/5/2023 9:57 AM	File folder

The CONUS NAIP directory is divided into geographical regions:

Directory that contains the collection of all CONUS NAIP terrain databases.



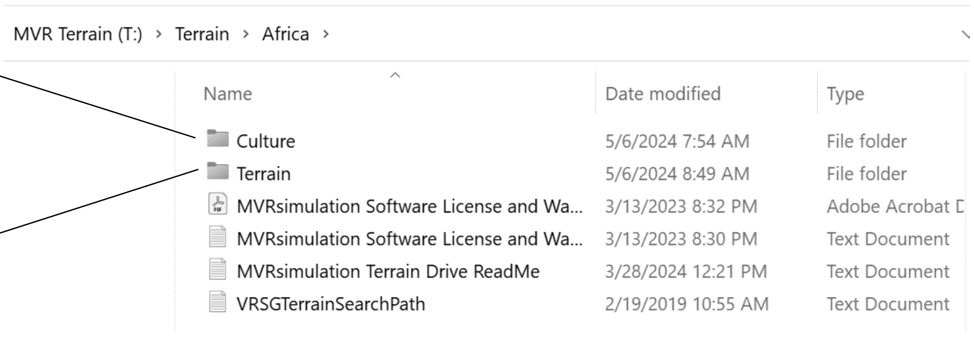
MVR Terrain (G:) > Terrain > CONUS NAIP

Name	Date modified	Type
North Central	8/9/2022 1:25 PM	File folder
Northeast	7/3/2023 8:04 AM	File folder
Northwest	8/15/2022 10:30 AM	File folder
South Central	7/3/2023 8:12 AM	File folder
Southeast	7/3/2023 8:17 AM	File folder
Southwest	1/6/2023 11:38 AM	File folder

Each region of terrain includes both \Culture and \Terrain subdirectories as shown in the following example of the Africa directory:

Contains 3D geospecific databases including vrsg.clt files.

Includes regional MDS terrain tiles, usually without any 3D culture.



MVR Terrain (T:) > Terrain > Africa >

Name	Date modified	Type
Culture	5/6/2024 7:54 AM	File folder
Terrain	5/6/2024 8:49 AM	File folder
MVRsimulation Software License and Wa...	3/13/2023 8:32 PM	Adobe Acrobat D
MVRsimulation Software License and Wa...	3/13/2023 8:30 PM	Text Document
MVRsimulation Terrain Drive ReadMe	3/28/2024 12:21 PM	Text Document
VRSGTerrainSearchPath	2/19/2019 10:55 AM	Text Document

The \Culture subdirectory contains the region's geospecific 3D models that are loaded on the terrain when VRSG initially renders the visualization window. The pre-loaded models can include buildings, fences, trees, light poles, and static characters or vehicles. The \Culture directory contains one or more vrsg.clt files that instruct VRSG where to load the cultural items.

The Terrain subdirectory contains the ortho imagery and elevation data associated with the imagery rendered in the visualization window. The terrain files are saved in MVRsimulation's VRSG round-earth terrain format (.mds).

All 3D models that are loaded before a training session begins are considered culture. All imagery loaded during a VRSG session is considered terrain.

The Culture subdirectory

The \Culture directory contains geospecific databases specifically modeled to replicate detailed environments such as cities or airfields. These databases are designed to be geographically smaller than the entire regional terrain and integrate seamlessly with the broader terrain found in the \Terrain subdirectory.

VRSG pre-loads all culture referenced in the search path upon startup. For this reason, you want to only select the specific subdirectories associated with your training area. Loading an entire \Culture subdirectory will drastically impact startup time and performance. See the section "Configuring terrain in the VRSG Search Path" later in this chapter for more information about how to properly set up the search path.

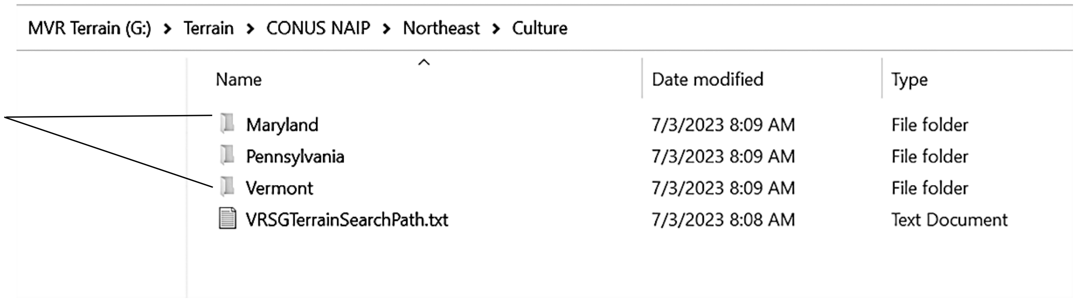
Key characteristics of the culture directories include:

- Geospecific 3D models and MDS tiles crafted specifically for the represented area.
- When loaded with the corresponding \Terrain directory's data, these databases blend smoothly with the underlying terrain.
- These cultural elements enhance the realism of the simulated environment.

Each region of terrain will have a \Culture subdirectory that is organized according to its location: either by U.S. State or by country. Individual areas of interest such as cities and airfields will be further organized within each state and country directory.

The following states of the Northeast CONUS NAIP region have modeled geospecific cultural databases in the \Culture folder:

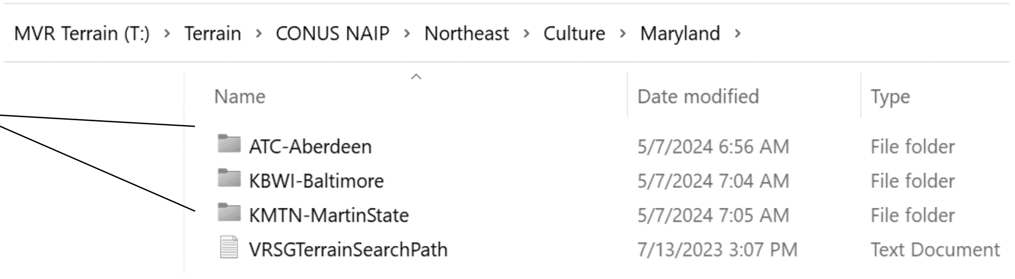
Directory of states with available 3D geospecific cultural databases in the Northeast CONUS NAIP directory.


MVR Terrain (G:) > Terrain > CONUS NAIP > Northeast > Culture

Name	Date modified	Type
Maryland	7/3/2023 8:09 AM	File folder
Pennsylvania	7/3/2023 8:09 AM	File folder
Vermont	7/3/2023 8:09 AM	File folder
VRSGTerrainSearchPath.txt	7/3/2023 8:08 AM	Text Document

Each state or country \Culture subdirectory will contain the specific areas of interest that are modeled as shown in the following example of the airfields in Maryland:

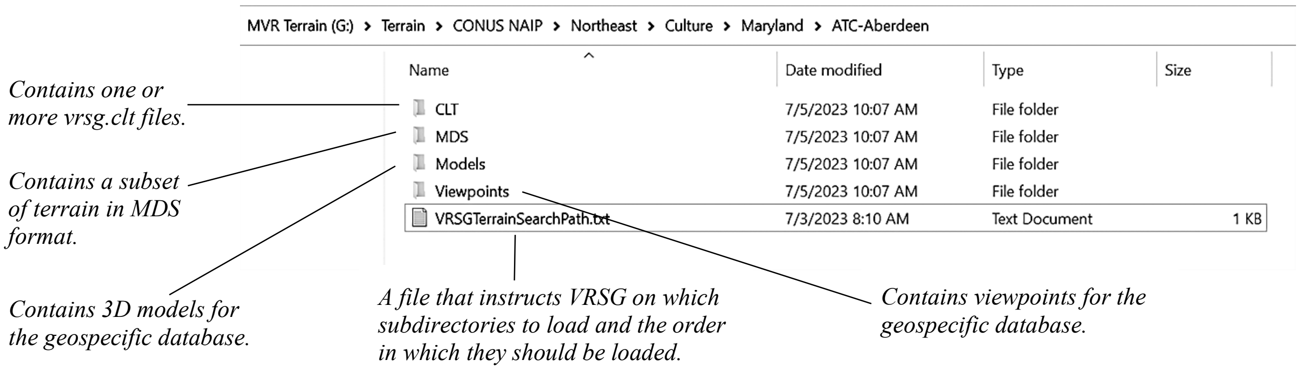
Directory of specific cultural databases in the Maryland subdirectory.


MVR Terrain (T:) › Terrain › CONUS NAIP › Northeast › Culture › Maryland ›

Name	Date modified	Type
ATC-Aberdeen	5/7/2024 6:56 AM	File folder
KBWI-Baltimore	5/7/2024 7:04 AM	File folder
KMTN-MartinState	5/7/2024 7:05 AM	File folder
VRSGTerrainSearchPath	7/13/2023 3:07 PM	Text Document

Structure of subdirectories in \Culture directory

The subdirectories within each \Culture directory usually contain the following folders: CLT, MDS, Models, Viewpoints, and Scenarios. While you shouldn't need to select any of the subdirectories within \Culture, the contents of each cultural terrain database directory generally adhere to the data structure illustrated in the following example:



MVR Terrain (G:) > Terrain > CONUS NAIP > Northeast > Culture > Maryland > ATC-Aberdeen

Name	Date modified	Type	Size
CLT	7/5/2023 10:07 AM	File folder	
MDS	7/5/2023 10:07 AM	File folder	
Models	7/5/2023 10:07 AM	File folder	
Viewpoints	7/5/2023 10:07 AM	File folder	
VRSGTerrainSearchPath.txt	7/3/2023 8:10 AM	Text Document	1 KB

Contains one or more vrsg.clt files.

Contains a subset of terrain in MDS format.

Contains 3D models for the geospecific database.

A file that instructs VRSG on which subdirectories to load and the order in which they should be loaded.

Contains viewpoints for the geospecific database.

Additionally, some cultural databases may include:

- Scenarios: Contains pre-scripted pattern-of-life scenarios provided by MVRsimulation, created using Scenario Editor and tailored to the terrain.
- VectorData: Contains shapefiles of cultural features for use with MVRsimulation's Terrain Tools or Battlespace Simulations' Modern Air Combat Environment (MACE).

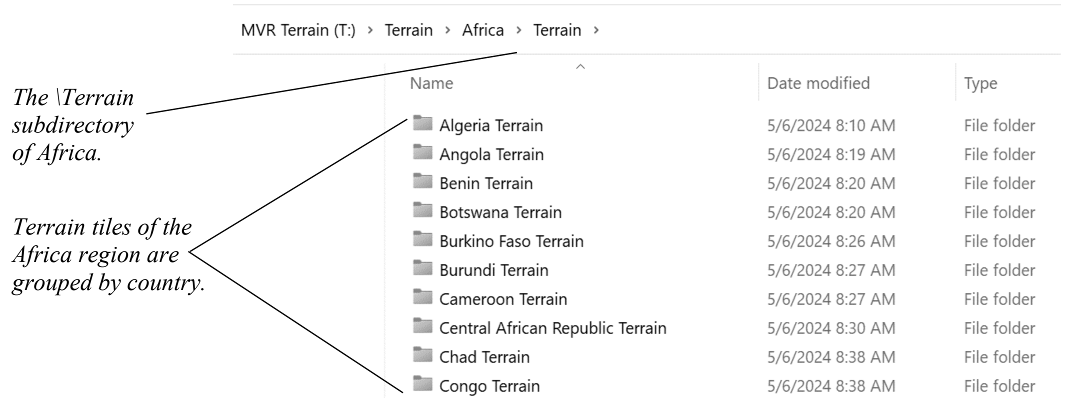
Note: Previously created cultural feature files with the older convention of "metadesic.clt" are respected by VRSG v7.

The Terrain directory

The Terrain directory contains a collection of terrain tiles in VRSG round-earth terrain format (.mds). This directory includes terrain compiled from orthoimagery and elevation source data and typically does not include geospecific 3D culture.

The Terrain directory should be loaded into the VRSG search path beneath any cultural directory. See the section "Configuring terrain in the VRSG Search Path" later in this chapter for more information about how to properly set up the search path.

Terrain directories at the continent level include subdirectories of terrain tiles which are grouped by country name as shown in the Africa terrain directory example below:



MVR Terrain (T:) > Terrain > Africa > Terrain >

Name	Date modified	Type
Algeria Terrain	5/6/2024 8:10 AM	File folder
Angola Terrain	5/6/2024 8:19 AM	File folder
Benin Terrain	5/6/2024 8:20 AM	File folder
Botswana Terrain	5/6/2024 8:20 AM	File folder
Burkino Faso Terrain	5/6/2024 8:26 AM	File folder
Burundi Terrain	5/6/2024 8:27 AM	File folder
Cameroon Terrain	5/6/2024 8:27 AM	File folder
Central African Republic Terrain	5/6/2024 8:30 AM	File folder
Chad Terrain	5/6/2024 8:38 AM	File folder
Congo Terrain	5/6/2024 8:38 AM	File folder

The \Terrain subdirectory of Africa.

Terrain tiles of the Africa region are grouped by country.

Terrain is loaded into VRSG on-demand. This means you can select the root Terrain directory of any region without impacting performance. When loading terrain into VRSG, you have the option to either select the entire Terrain directory to load all terrain data for that region. Alternatively, you can choose to load individual countries or states (for CONUS NAIP) by selecting their respective root directories.

Introduction to VRSG search paths

Upon startup, VRSG loads the default terrain, models, scenarios, and textures directory installed with the software. To direct VRSG to load additional content, you must specify the directories for the additional terrain, models, textures, scenarios, and other files. This set of directory paths is called the *search path*.

Key Points about search paths:

- The search path allows you to customize how and in what order VRSG loads various content.
- The order in which directories are listed is critical because VRSG searches the list from top to bottom. Higher resolution or higher priority content should be listed first.
- To ensure accurate rendering of the 3D cultural content, the Culture directories **must be prioritized at the top** of your VRSG search path.
- Use the arrows on the lefthand side to adjust the priority of the directories.

Configuring terrain in the VRSG search path

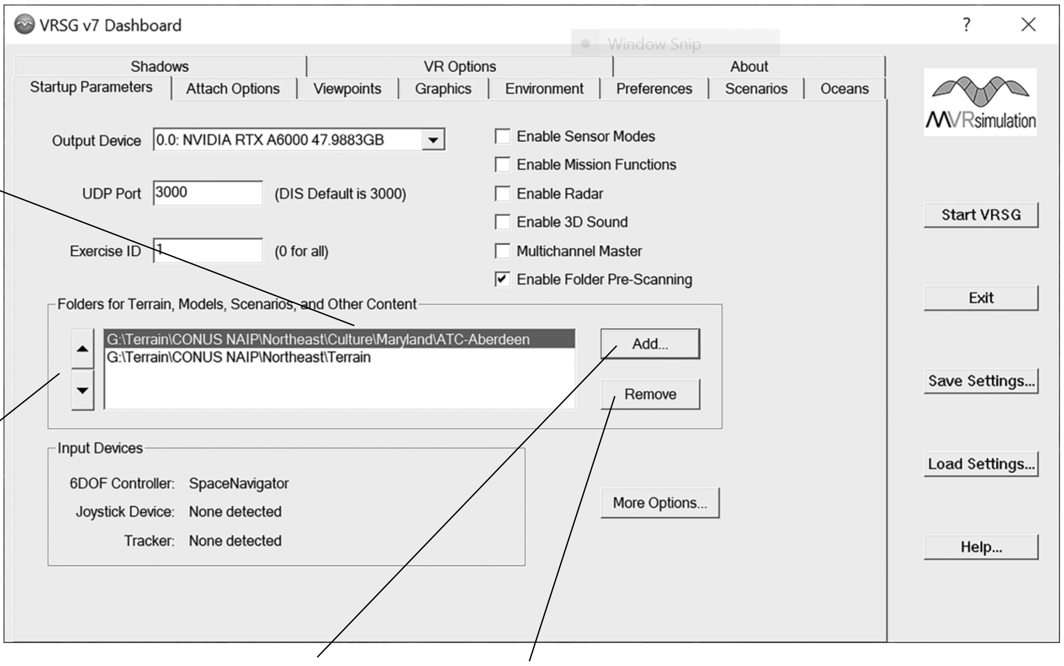
In the "Folders for Terrain, Models, Scenarios, and Other Content" section of the VRSG Dashboard, list all paths to the terrain, models, and other files (such as cultural feature files and scenarios) you want VRSG to load. These directories form the VRSG search path.

VRSG loads the directories in the search path before rendering terrain. Therefore, specify all directories before starting the visualization window. If changes to the search path are needed after starting the visualization window, you must exit and relaunch VRSG.

The order in which directories are listed in the search path is critical, as VRSG searches the list from top to bottom. Cultural files and higher-resolution terrain insets located in the \Culture directory should be listed at the top of the list with imagery and elevation data terrain files from the \Terrain directory at the bottom. If VRSG finds a file for a specific terrain tile in the first folder, it will ignore all subsequently found tiles with the same name. If a directory that contains an .mds terrain tile with no culture is listed above a directory with an .mds terrain tile of the same name that includes culture, the culture will not load since VRSG will ignore the later terrain tile.

Note: MVRsimulation recommends only loading culture for the specific areas you will be training in to improve performance and reduce start times. You can include terrain outside of the training area in the search path without affecting performance since VRSG loads the underlying terrain on-demand.

Examples of loading terrain and culture into the VRSG search path are shown below in the section “Examples of VRSG search paths.”



Output Device: 0.0: NVIDIA RTX A6000 47.9883GB

UDP Port: 3000 (DIS Default is 3000)

Exercise ID: 1 (0 for all)

☐ Enable Sensor Modes
☐ Enable Mission Functions
☐ Enable Radar
☐ Enable 3D Sound
☐ Multichannel Master
☑ Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content

G:\Terrain\CONUS NAIP\Northeast\Culture\Maryland\ATC-Aberdeen
G:\Terrain\CONUS NAIP\Northeast\Terrain

Input Devices

6DOF Controller: SpaceNavigator
Joystick Device: None detected
Tracker: None detected

Be sure to place the Culture directory at the top of the loading order.

Use these arrow buttons to adjust the loading order of the terrain directories.

Click here to add a directory to the VRSG search path.

Highlight a directory and click here to remove it from the VRSG search path.

To load directories in the search path:

1. On the Dashboard's Startup Parameters tab, in the Folders for Terrain, Models, Scenarios, and Other Content section, click the Add button.
2. Browse to the directory you want to add to the search path and click OK.
3. For each additional directory you want to load, click the Add button and browse again to the directory of interest and click OK.
4. Use the arrow buttons on the left of the list of directories (the search path list) to move a directory higher or lower in search priority.
5. Once you have added all the cultural files directories and terrain tile directories to the search path, verify they are in the correct priority order (culture above terrain) and update as needed.
6. Click Start VRSG to load the visualization window.

Note: The order in which directories are listed is critical, as VRSG searches the list from top to bottom. In general, culture directories should always be placed at the top of the VRSG search path with Terrain directories at the bottom.

Dynamic terrain tile updates

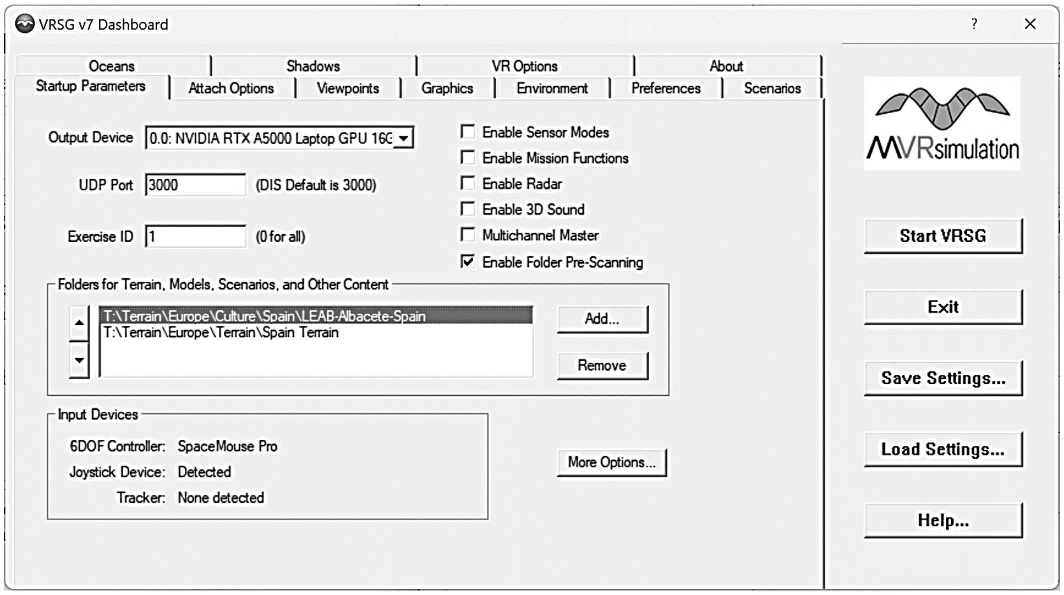
As of version 7, VRSG can accept a UDP message from Terrain Tools or a simulation host notifying it that a new tile has been built. VRSG adds the path of the new tile to the top of the runtime search path. If an older version of the same tile is already loaded, VRSG unloads it and loads the new one. You can also load the newly built or rebuilt tile manually, by dragging it to the visualization window.

Examples of VRSG search paths

This section will describe various VRSG search paths and how they are rendered in the VRSG visualization window, accompanied by images of the associated Dashboard's Startup Parameters tab.

Example 1 - Loading the LEAB-Albacete-Spain terrain database:

In this example, only the LEAB-Albacete-Spain airfield from the Culture directory will be rendered in VRSG, along with the underlying terrain for the entire country of Spain.



Output Device: 0.0: NVIDIA RTX A5000 Laptop GPU 16G

UDP Port: 3000 (DIS Default is 3000)

Exercise ID: 1 (0 for all)

☐ Enable Sensor Modes
☐ Enable Mission Functions
☐ Enable Radar
☐ Enable 3D Sound
☐ Multichannel Master
☑ Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content

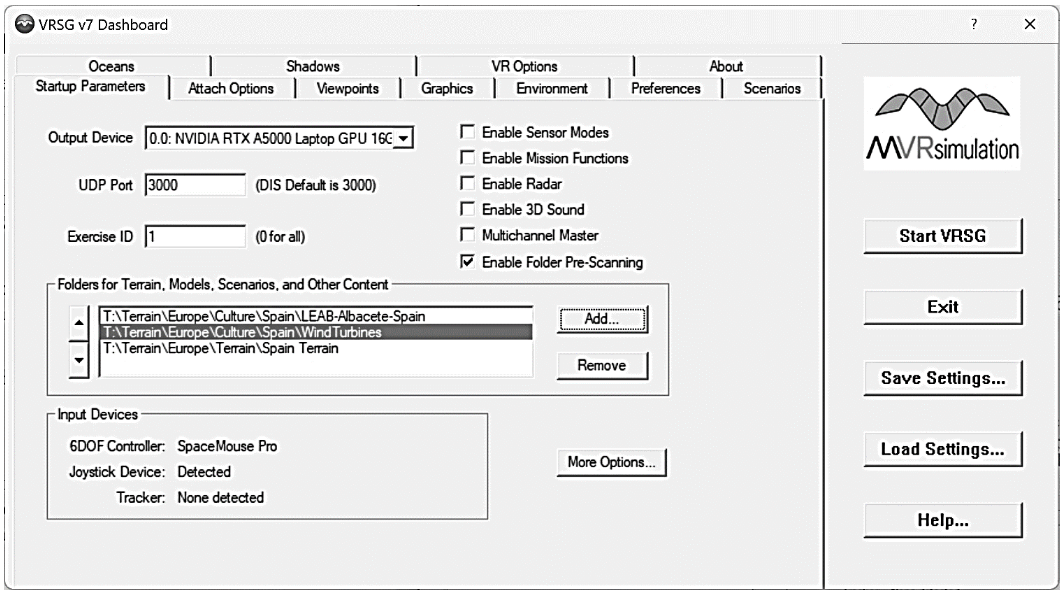
```
T:\Terrain\Europe\Culture\Spain\LEAB-Albacete-Spain
T:\Terrain\Europe\Terrain\Spain Terrain
```

Input Devices

6DOF Controller:	SpaceMouse Pro
Joystick Device:	Detected
Tracker:	None detected

Example 2 - Adding an additional \Culture subdirectory:

In this example, a cultural directory for wind turbines located around southern Spain will be added to the visualization window. The underlying terrain will still display only the country of Spain.



Output Device: 0.0: NVIDIA RTX A5000 Laptop GPU 16G
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

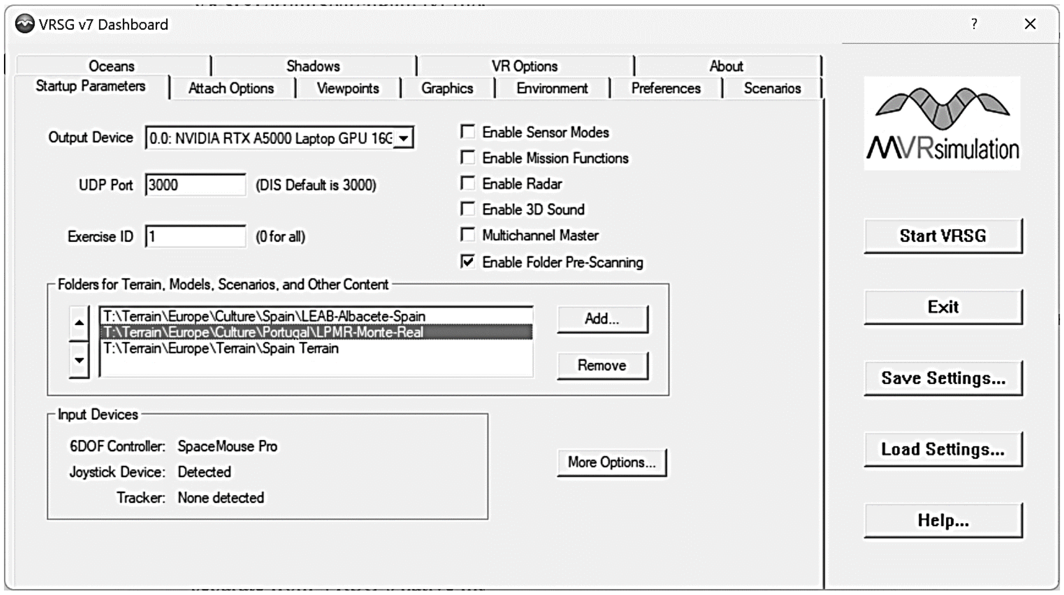
- [ ] Enable Sensor Modes
- [ ] Enable Mission Functions
- [ ] Enable Radar
- [ ] Enable 3D Sound
- [ ] Multichannel Master
- [x] Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content
T:\Terrain\Europe\Culture\Spain\LEAB-Albacete-Spain
T:\Terrain\Europe\Culture\Spain\WindTurbines
T:\Terrain\Europe\Terrain\Spain Terrain

Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: None detected

Example 3 - Removing and adding additional areas of interest:

In this example, the Wind Turbines have been removed and cultural files from LPMR-Monte-Real airfield in Portugal has been added. However, the underlying terrain for Portugal has not been added into the search path, so only the cultural files and high-resolution terrain located in the \Terrain\Europe\Culture\Portugal\LPMR-Monte-Real directory will be seen after leaving the border of Spain.



Output Device: 0.0: NVIDIA RTX A5000 Laptop GPU 16G
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

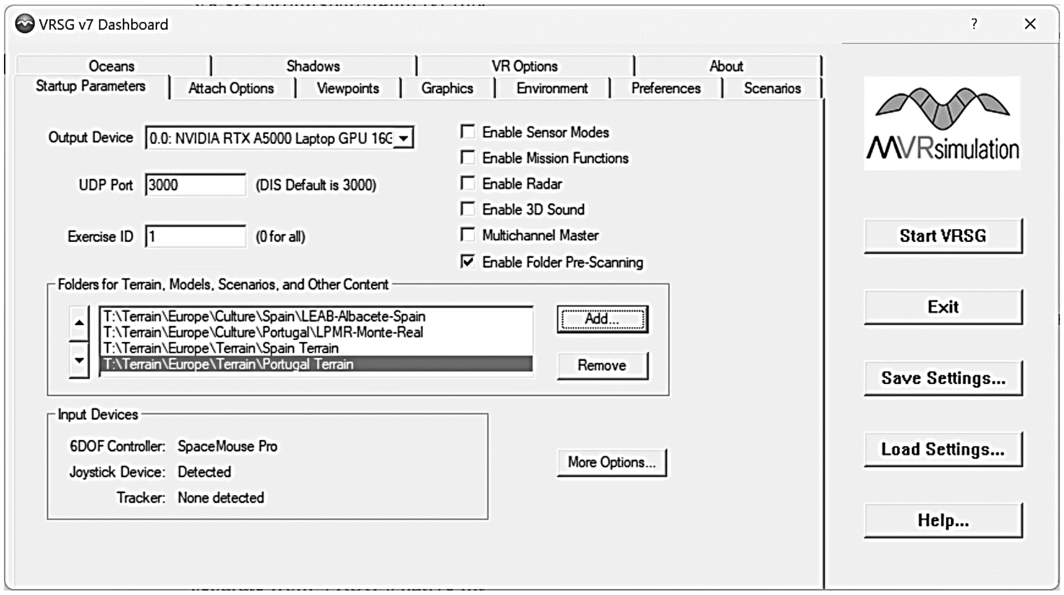
- [ ] Enable Sensor Modes
- [ ] Enable Mission Functions
- [ ] Enable Radar
- [ ] Enable 3D Sound
- [ ] Multichannel Master
- [x] Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content
T:\Terrain\Europe\Culture\Spain\LEAB-Albacete-Spain
T:\Terrain\Europe\Culture\Portugal\LPMR-Monte-Real
T:\Terrain\Europe\Terrain\Spain Terrain

Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: None detected

Example 4 - Adding the underlying terrain for Portugal:

In this example, the underlying terrain for Portugal has been added to the search path. Now the terrain from LEAB-Albacete-Spain to LPMR-Monte-Real will be contiguous through Spain and Portugal.



Output Device: 0.0: NVIDIA RTX A5000 Laptop GPU 16G
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

- [ ] Enable Sensor Modes
- [ ] Enable Mission Functions
- [ ] Enable Radar
- [ ] Enable 3D Sound
- [ ] Multichannel Master
- [✓] Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content

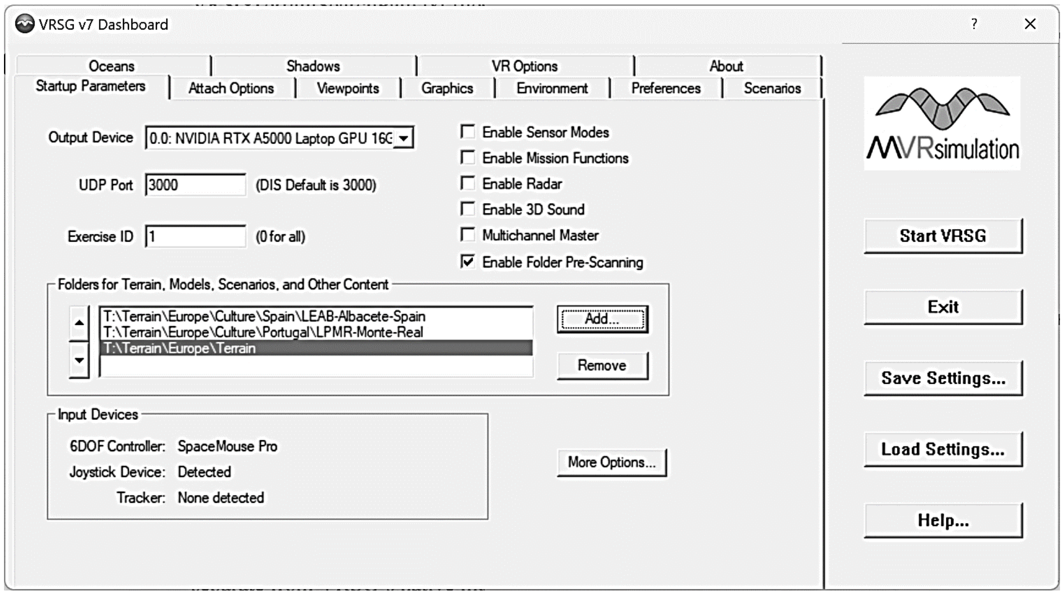
```
T:\Terrain\Europe\Culture\Spain\LEAB-Albacete-Spain
T:\Terrain\Europe\Culture\Portugal\LPMR-Monte-Real
T:\Terrain\Europe\Terrain\Spain Terrain
T:\Terrain\Europe\Terrain\Portugal Terrain
```

Input Devices

6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: None detected

Example 5 - Using the region’s root terrain directory.

In this example, the root \Europe\Terrain directory has been loaded. This means VRSG will render 3D cultural entities of only LEAB-Albacete-Spain and LPMR-Monte-Real. You will also see all of Europe’s underlying terrain. This approach allows you to load only selected cultural areas to improve performance while including the imagery and elevation data for the entire region. Using a geographically expansive \Terrain directory does not impact performance, as VRSG will dynamically load only the subset of terrain tiles that are visible from the current viewpoint.



Output Device: 0.0: NVIDIA RTX A5000 Laptop GPU 16G
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

- [ ] Enable Sensor Modes
- [ ] Enable Mission Functions
- [ ] Enable Radar
- [ ] Enable 3D Sound
- [ ] Multichannel Master
- [✓] Enable Folder Pre-Scanning

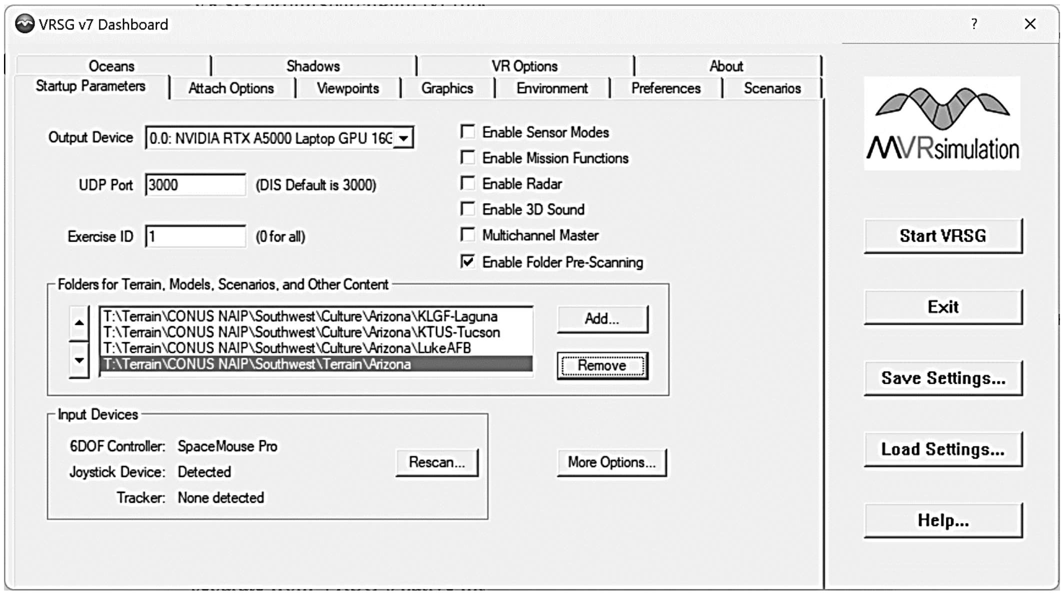
Folders for Terrain, Models, Scenarios, and Other Content

```
T:\Terrain\Europe\Culture\Spain\LEAB-Albacete-Spain
T:\Terrain\Europe\Culture\Portugal\LPMR-Monte-Real
T:\Terrain\Europe\Terrain
```

Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: None detected

Example 6 – Culture and terrain from Arizona in the CONUS NAIP Southwest directory:

Here is an example featuring three airfields in Arizona from the Culture directory, along with the underlying terrain data of Arizona. The culture folders are listed above the terrain directory since they are higher fidelity.



Output Device: 0.0: NVIDIA RTX A5000 Laptop GPU 16G
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

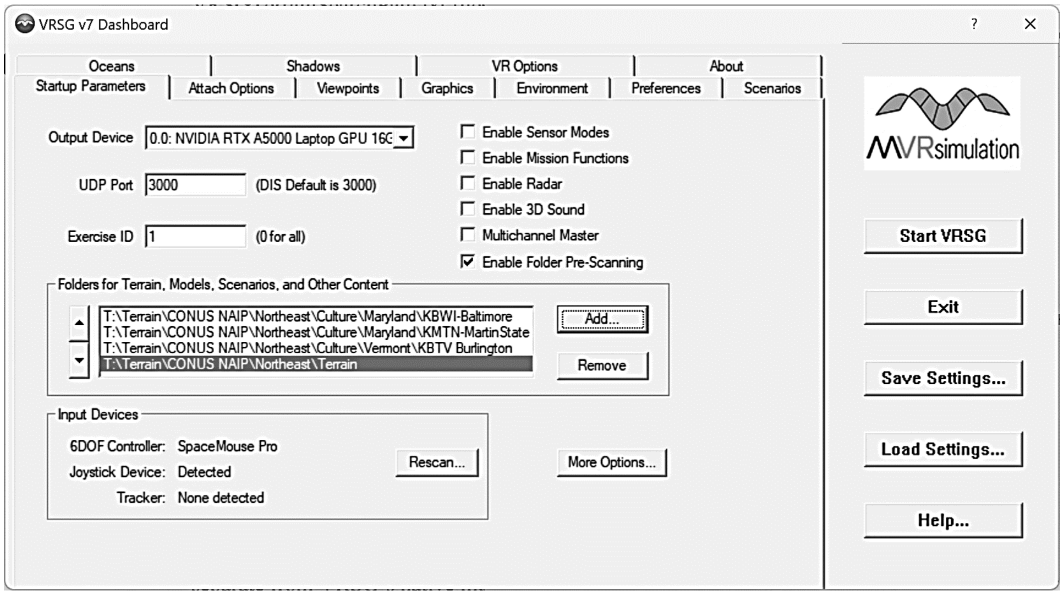
- [ ] Enable Sensor Modes
- [ ] Enable Mission Functions
- [ ] Enable Radar
- [ ] Enable 3D Sound
- [ ] Multichannel Master
- [✓] Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content

```
T:\Terrain\CONUS NAIP\Southwest\Culture\Arizona\KLGF-Laguna
T:\Terrain\CONUS NAIP\Southwest\Culture\Arizona\KTUS-Tucson
T:\Terrain\CONUS NAIP\Southwest\Culture\Arizona\LukeAFB
T:\Terrain\CONUS NAIP\Southwest\Terrain\Arizona
```

Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: None detected

Example 7 – In this example, two airfields in Maryland and one in Vermont are loaded from the Culture directory. The entire Northeast Terrain database is loaded as well.


VRSG v7 Dashboard

Output Device: 0.0: NVIDIA RTX A5000 Laptop GPU 16G
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

- [ ] Enable Sensor Modes
- [ ] Enable Mission Functions
- [ ] Enable Radar
- [ ] Enable 3D Sound
- [ ] Multichannel Master
- [x] Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content

```
T:\Terrain\CONUS NAIP\Northeast\Culture\Maryland\KBWI-Baltimore
T:\Terrain\CONUS NAIP\Northeast\Culture\Maryland\KMTN-MartinState
T:\Terrain\CONUS NAIP\Northeast\Culture\Vermont\KBTV Burlington
T:\Terrain\CONUS NAIP\Northeast\Terrain
```

Input Devices

6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: None detected

Organizing custom culture and terrain files in VRSG

As described throughout this user’s guide, culture, textures, viewpoints, and scenarios play crucial roles in enhancing the realism of the virtual 3D world rendered by VRSG. VRSG supports the use of multiple files for cultural features, viewpoints, and textures, offering organizational flexibility. This capability is particularly useful when managing multiple scenarios that utilize the same underlying terrain but require specific cultural details and textures.

The terrain directories included with VRSG serve as examples of how to organize these files effectively. For instance, the Somalia terrain directory demonstrates a structured approach to managing a complex terrain project, located at \MVRsimulation\VRSG\Terrain\Somalia.

Managing Custom 3D Content

MVRsimulation advises storing your own culture and terrain files separately from VRSG's installed files. Before upgrading to a newer version of VRSG, always back up any custom files added to installed terrain directories to prevent loss of data during the upgrade process.

For your site-specific models, create a subdirectory named \User within the installed \MVRsimulation\VRSG\Models path. This keeps your custom models separate from VRSG’s native models:

```
\MVRsimulation\VRSG\Models\User
```

By default, VRSG searches the \MVRsimulation\VRSG\Models\User directory, so you do not need to add it to the search path. [I had no idea VRSG did this, but I confirmed in code that it does indeed look for Models/User by default, and will traverse that subdirectory recursively if it exists. Cool!] However, if you store models in a different location outside \MVRsimulation\VRSG\Models, you must add the directory to the top of the search path to ensure VRSG will find the models.

Ground clamping of models in VRSG depends on the subdirectory where the model is stored. MVRsimulation recommends organizing your site’s models into specific subdirectories:

```
\User\Military
\User\Commercial
\User\Characters
\User\Buildings
\User\Trees
\User\Signs
\User\Other
```

Additionally, create a \Textures subdirectory under each \User directory to store textures associated with models. For example, textures for user-built building models would be located at \MVRsimulation\VRSG\Models\User\Buildings\Textures.

Note: Before removing an older version of VRSG during an upgrade, remember to back up the \MVRsimulation\VRSG\Models\User directory to preserve your custom content.

For more detailed information on managing models, refer to the chapters: "Configuring Models and Events," "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats," and "3D Model Content."

Managing textures

Textures associated with terrain tiles should be placed in a subdirectory named \Textures within the directory of the tiles. These textures are used for models or applied directly to the terrain (e.g., road textures or microtextures). Terrain tiles created with MVRsimulation Terrain Tools may reference external textures such as for roads or fences, and these textures must be stored in the \Textures subdirectory.

An example of a terrain directory with a Textures subdirectory can be seen at \MVRsimulation\VRSG\Terrain\Afghanistan\MDS\Textures

For more information on managing textures, refer to the chapter "Manipulating Textures."

Microtextures

Microtextures are high-resolution, geotypical ground textures used to enhance terrain detail at ground level. VRSG blends one or more microtextures with geospecific terrain imagery during runtime.

Each directory of terrain tiles can have its own unique microtexture stored in the appropriate \Textures subdirectory. VRSG applies the microtexture present in this subdirectory to all terrain tiles within that directory.

An example of a directory containing microtextures can be found in the VRSG installation directory at \MVRsimulation\VRSG\Terrain\Somalia\Kismayo\MDS\Textures.

Water Textures to Override VRSG's Default:

You can specify a custom water texture on a per-tile directory basis to replace VRSG's default water texture during runtime. This customization is useful when representing bodies of water that differ significantly in color from VRSG's default water.

The custom water texture file must be named "water_override.tex" and reside in a \Textures subdirectory within the directory containing the associated terrain tiles. Additionally, provide a lower resolution version named "water_override_low_detail.tex" for blending at greater distances. Ensure the image file meets VRSG's texture requirements as outlined in the chapter "Manipulating Textures."

An example of a custom brown water texture can be found in the \MVRsimulation\VRSG\Terrain\Afghanistan\MDS\textures directory.

Texture-material attribution in VRSG

Utilizing VRSG's physics-based sensor capability involves attributing material codes to scene elements through texture mapping. One approach is creating a vrsg.irm table to map textures to materials. This file can be placed in any directory within VRSG's search path. The default vrsg.irm file is located in the \MVRsimulation\VRSG\Textures directory.

For example, each geospecific terrain may have its unique vrsg.irm file, such as the one located in the \MVRsimulation\VRSG\Terrain\Afghanistan\vrsg.irm directory.

For detailed guidance on mapping textures to material codes for sensor simulation, refer to the chapter "Working with Sensor-View Modes and Physics-Based IR."

Multiple vrsg.irm Support

VRSG supports multiple vrsg.irm files, which can be stored in subdirectories of models they apply to. If multiple instances of vrgs.irm files reference the same texture, the first one will be used. In this case, VRSG will output conflicts to the VrsgError_MachineName.txt file located in the VRSG root directory, typically C:\MVRsimulation\VRSG.

Wind Sensitivity Metric

VRSG v7 allows you to assign a wind sensitivity metric to textures within the vrsg.irm file, as demonstrated in the default \VRSG\Textures vrsg.irm file. This feature enables vegetation textures to dynamically respond to wind direction and velocity.

To assign a wind metric to a vegetation texture, use the `-windSensitivity=` command with a sensitivity value and the texture name. This command controls how vegetation models using the specified texture respond to wind. In the default vrsg.irm file, several vegetation textures utilize this command:

```
grass13-field 4019 -windSensitivity=0.3

poppy01_field 4210 -windSensitivity=0.3 -treeShader

poppy02_field 4210 -windSensitivity=0.3 -treeShader

poppy03_field 4210 -windSensitivity=0.3 -treeShader
```

Ensure that the model's base is at the origin (not underground) for the wind sensitivity command to function correctly. During simulation, wind direction typically originates from an external system. To test the vegetation texture in VRSG, press the W key to cycle through wind directions.

Managing viewpoints

VRSG supports multiple viewpoint files located in various subdirectories, such as directories organizing terrain tiles by geographic area or within specific scenario subdirectories. During visualization, VRSG loads all vrsg.viewpoint files found within its search path. Viewpoints created during a VRSG session are automatically saved in the default vrsg.viewpoint file located in \MVRsimulation\VRSG\Viewpoints directory.

The vrsg.viewpoint file is formatted as an editable ASCII text file. This format allows you to easily copy and paste new viewpoints from the default vrsg.viewpoint file to create custom terrain-specific viewpoint files. This capability enables you to manage specific viewpoints by saving them in a geospecific \Culture folder, ensuring they are loaded only when that \Culture folder is included in the search path.

For example, terrain-specific viewpoints can be found at:
`\MVRsimulation\VRSG\Terrain\Somalia\Kismayo\vrsg.viewpoint.`

Additional viewpoint files are stored within the scenarios for Kismayo, located in
`\MVRsimulation\VRSG\Terrain\Somalia\Kismayo\Scenarios\vrsg.viewpoint.`

Creating VRSGTerrainSearchPath.txt

Managing search paths for VRSG content can be complex, especially when dealing with multiple drives and directories containing various cultural features, microtextures, scenarios, and more. To streamline this process, you can utilize a text file named VRSGTerrainSearchPath.txt within a high-level directory listed in the Folders for Terrain, Models, Scenarios, and Other Content section.

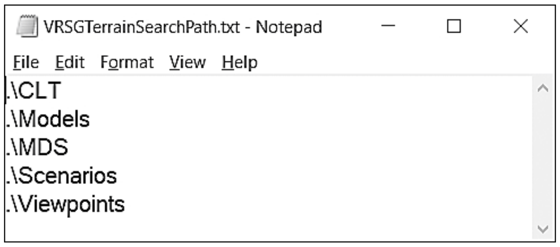
In VRSGTerrainSearchPath.txt, list the subdirectories in the order of priority that VRSG should search. This ASCII text file functions similarly to the search path settings on the Dashboard's Startup Parameters tab, with VRSG sequentially searching from top to bottom.

This method is particularly useful in configurations where directory structures frequently change or are intricate. Instead of managing these changes within VRSG, you simply update the VRSGTerrainSearchPath.txt file. It also facilitates setting up complex search paths for extensive collections of terrain tiles and cultural files by pointing to a single directory, which then expands into a comprehensive list of directories and subdirectories.

Examples of VRSGTerrainSearchPath.txt setups can be found in the directories of the terrain provided with VRSG installations, such as:

`MVRsimulation\VRSG\Terrain\Syria\Hajin\VRSGTerrainSearchPath.txt`

This file explicitly lists all \Hajin subdirectories that VRSG should search to render the corresponding terrain tiles, models, and scenarios delivered with VRSG. Note that this file does not include the underlying terrain for Syria located in the \Terrain folder. To incorporate this, you would need to add it explicitly to VRSGTerrainSearchPath.txt or include the directory manually in the search path.


VRSGTerrainSearchPath.txt - Notepad
File Edit Format View Help

```
.\CLT
.\Models
.\MDS
.\Scenarios
.\Viewpoints
```

The VRSGTerrainSearchPath.txt file is designed to store relative paths to directories using Windows notation for relative path conventions, excluding absolute paths. Each line in the file must begin with "." to indicate the root directory where the VRSGTerrainSearchPath.txt file resides. You can specify parallel subdirectories branching from another subdirectory by adding ".." after the initial ".".

For example:

```
.\..\..\..\New Terrain Directory\MDS
```

In Windows relative path conventions, a single dot denotes the current directory, and double dots signify moving up one level in the directory hierarchy.

After creating the VRSGTerrainSearchPath.txt file, place it within the directory paths where you want VRSG to search for 3D content to display. On the VRSG Dashboard's Startup Parameters tab, you only need to specify the directory containing the VRSGTerrainSearchPath.txt file. VRSG will then search the folders listed in the file accordingly.

In cases involving shared network drives, ensure to specify a path in the VRSG Dashboard to at least one VRSGTerrainSearchPath.txt file on each shared drive. Note that paths stored in the VRSGTerrainSearchPath.txt file cannot point to different shared network drives; they are restricted to pointing to relative folders on the same drive.

Copying terrain from an MVRsimulation terrain drive to a file storage server

When copying terrain from an MVRsimulation terrain drive to a permanent file storage server, it's crucial to maintain the original file structure. MVRsimulation strongly recommends preserving the directory structure as it exists on the MVRsimulation terrain drive.

Certain directories, such as those containing high-resolution terrain tiles (\Culture directory) or lower resolution tiles with broader coverage (\Terrain directory), serve specific purposes. Retaining both directories separately ensures that VRSG can access the appropriate data without confusion.

MVRsimulation terrain drives include default VRSGTerrainSearchPath.txt files that list all necessary subdirectories implicitly included in VRSG's search path. If you opt for a different organizational structure on your file storage server, ensure these points are considered to prevent potential loading issues when reorganizing terrain data.

Adding new databases

MVRsimulation frequently adds new and updated culture and terrain databases to MVRsimulation's Downloads Server. Users can download a subset of cultural or terrain databases from downloads.mvrsimulation.com.

These databases should be added to the appropriate directories in your VRSG terrain storage.

To add any new or updated databases:

1. Visit downloads.mvrsimulation.com and log in.
2. Download the desired databases.
3. Unzip the downloaded files.
4. Copy the new databases into the respective \Culture folders within your VRSG terrain directory structure.

Additional terrain support

For further assistance with loading terrain, or any other terrain-related support questions, send an email to terrain@mvrsimulation.com.

For new or updated terrain drives, send your request to sales@mvrsimulation.com.

Customers on active maintenance who need an account on MVRsimulation's Downloads Server can request an account by sending an email to downloads@mvrsimulation.com.

CHAPTER 5

Configuring Models and Events

VRSG gives you complete control over the mapping of entity appearance in the virtual world. An entity is any moving model in the virtual world, such as a truck, an aircraft, or an individual combatant. In addition to controlling the appearance of entities you can associate events in the virtual world with animation effects by editing the mapping information in VRSG configuration files. To effectively manipulate the mapping information for entities and events, you should become familiar with the DIS enumerations for entities.

Information about DIS is available at the MVRsimulation website and from these sources:

- IEEE 1278.1-1995, Standard for Distributed Interactive Simulation - Application Protocols.
- IEEE 1278.2-1995, Standard for Distributed Interactive Simulation - Communication Services and Profiles.

In the directory \MVRsimulation\VRSG\Models, MVRsimulation provides the latest published DIS standards document for your convenience: SISO-REF-010-2023 Enumerations v31.pdf.

VRSG has no explicit limit on the number of entities it can handle in a scene while maintaining real-time performance. MVRsimulation customers have run exercises with upwards of 15,000 entities while maintaining real-time performance. The only limitations might be performance considerations such as the bandwidth of the communications channel, model complexity, terrain complexity, viewing range, and how near the entities are in your field of view.

Mapping virtual world entities

The \MVRsimulation\VRSG\Models directory contains MVRsimulation's 3D content libraries. The 3D model files are in MVRsimulation's HPY or HPX model format; they contain vertex data for the polygonal wireframe models that are rendered in the virtual world. The file header information in a model file in turn references the appropriate texture map files that will be applied to the polygonal model. (See the chapter "MVRsimulation 3D Model Format" for information about MVRsimulation's model file format.)

The mapping information that controls which model is mapped to the appropriate DIS entity enumeration is contained in the ModelMap.ini file. This initialization file is located in the \MVRsimulation\VRSG\Models directory. At runtime, VRSG loads one ModelMap.ini, the first one it encounters in the search path. If no ModelMap.ini exists in the search path, then VRSG loads the one in the \Models directory.

You can determine which ModelMap.ini file VRSG is using by inspecting the VrsgInfo_*.txt file, located in the \MVRsimulation\VRSG directory.

Modelmap.ini is installed with entries commented out. For entities you plan to use in VRSG, edit the file to remove the comment character, change the defaults, and add your entity-mapping information. The mapped entities will be displayed in the All Entities list on the VRSG Dashboard's Attach tab.

Note: VRSG uses the file \Models\LegacyModelNames.txt to convert between a legacy model name and a new model name. This allows MVRsimulation to rename a model for a VRSG release, to give the model a more tactically-appropriate name, or to replace a deprecated model with a new one. This LegacyModelNames.txt enables legacy ModelMap and cultural feature files to continue to work in VRSG, without generating any errors about missing models, while providing continuous improvements to the MVRsimulation model libraries.

ModelMap syntax

The ModelMap.ini initialization file must contain one mapping entry per line. Blank lines are ignored, and lines that begin with a comment marker '!' are also ignored. The general syntax of a ModelMap.ini entry is:

```
<DIS enumeration> <Model-class> <visual-model> [model load options…]
```

The DIS enumeration is a 7 number identifier that collectively identifies a particular type of entity. The DIS terminology uses these labels for the 7 number identifier:

```
Kind Domain Country Category Sub-Category Specific Extra
```

Typical values for Kind are 1 (platform), or 3 (lifeform). Most entities that are associated with a vehicle will use a Kind value of 1. Typical values for Domain include 1 (land), 2 (air), or 3 (surface). Typical values for Country are 225 (USA), 1 (Afghanistan), or 222 (Russia). For a complete reference of the DIS enumeration values, see the documents referenced on the previous page.

For a DIS enumeration in the ModelMap.ini file, you can enter an asterisk '*' for any of the 7 numbers. An asterisk will match any value from an incoming entity. When VRSG sees a new entity on the network, it scans ModelMap.ini from top to bottom, searching for an enumeration match. The values of the enumeration will match if they are an asterisk, or if the given number matches that of the new entity. You should place more specific entries towards the top of the file, followed by more general entries towards the bottom of the file.

To make an entity invisible (hidden), map its enumeration to the model file empty.hpx.

Each model in the ModelMap.ini file is loaded only once, and only one copy is stored in memory. Multiple entries using the same model name in the ModelMap file (as well as multiple instances of the model in the scene) will all share a single instance of the model's resources (textures and geometry) in system memory. This means that if you have multiple DIS entities in a VRSG scene that use the same model, VRSG will load that model only once.

The model class informs VRSG how to display the particular entity. Possible values for class are the keywords `Vehicle`, `Human`, `Bubble` and `Cylinder`. These model classes and their application are described in the sections below.

Displaying vehicle models

The `Vehicle` model class is used for entities that are associated with a vehicle or other rigid-body model. In ModelMap.ini, such models begin with the class `Vehicle`. For example:

```
1  1  225   6  1  3  *  Vehicle  M1046.M-220.US.desert.hpy
1  1  225   2  5  2  *  Vehicle  M1128.M2.US.green.hpy
```

Displaying 3D animated human character models

The `Human` model class is used for entities that are associated with human life form entities. A visual model that you associate with a Human class should be a 3D character model from the MVRsimulation character model library. In ModelMap.ini, such models begin with the prefix “human-” as in human-us_soldier-033.hpy. For example:

```
3 1   1 * * * * Human human-afghan-001.hpy
3 1 225 * * * * Human human-us_soldier-033.hpy
```

Human entities can have an optional weapon associated with them. To associate a weapon with a human character, simply add the name of the weapon model after the name of the character model in the ModelMap.ini entry. Weapon models in the MVRsimulation character model library begin with the prefix “weapon-” as in weapon-m16.hpy. For example:

```
3 1   1 * * * * Human human-afghan-001.hpy weapon-ak47.hpy
3 1 225 * * * * Human human-us_soldier-003.hpy weapon-m16.hpy
```

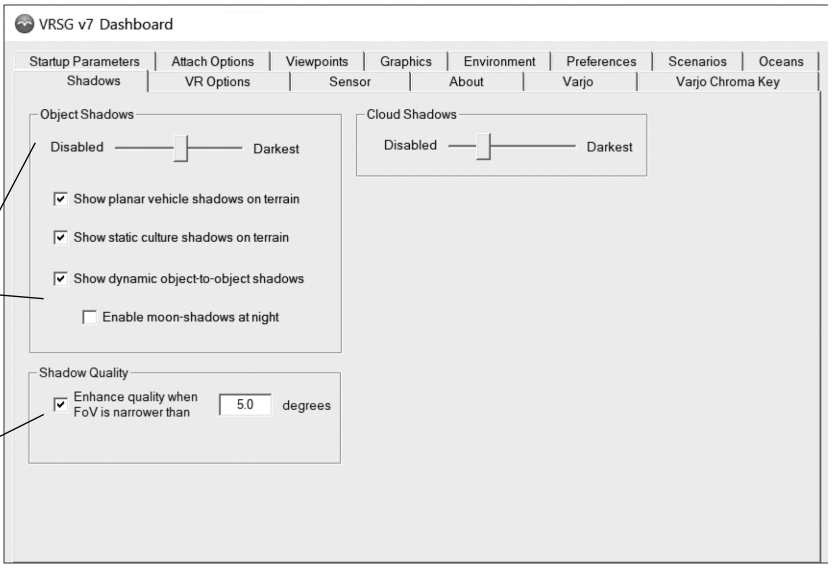
Other objects carried by human entities (such as a shovel, cell phone, laser designator, and so on) can be attached to characters as weapons in the same manner. MVRsimulation’s model library contains over 200 animated human character models that can be referenced by ModelMap.ini. The library includes animations for these models to support the most common life form states used by the DIS protocol. In addition, animations are provided for the primary weapon state of the DIS protocol. Distinct animations are provided for the weapon in the deployed position or in the firing position.

Displaying shadows cast by models and clouds

To display shadows cast by all objects in a scene (both dynamic entities and static cultural models), and/or volumetric clouds, activate the appropriate option(s) on the VRSG Dashboard’s Shadows tab.

Use the sliders and checkboxes to activate and control the intensity of the cast shadows.

Click this checkbox and enter a narrow FOV threshold at which shadow enhancement will take effect.



Object Shadows: Disabled — Darkest

[✓] Show planar vehicle shadows on terrain

[✓] Show static culture shadows on terrain

[✓] Show dynamic object-to-object shadows

[ ] Enable moon-shadows at night

Cloud Shadows: Disabled — Darkest

Shadow Quality: [✓] Enhance quality when FoV is narrower than 5.0 degrees

To activate shadows for clouds and/or objects (models), move the appropriate slider from the left (shadows disabled) to the right to control the intensity of the cast shadows. Controlling the intensity of each kind of shadow can be useful in scenes where there is a union of shadows cast from multiple sources. The Object Shadows slider operates on the type(s) of shadow selected. Minimize the performance impact of object shadows on a culture-dense scene by unselecting one or more shadow options to optimize for performance.

VRSG has the ability to improve the appearance of dynamic cast shadows when viewing the scene through a narrow field-of-view (FOV), such as those typically used for UAV sensors. A narrow FOV induces high magnification and greater standoff distances, which can cause shadows to become washed out or to disappear entirely. Click the Shadow Quality checkbox to direct VRSG to render dynamic shadows at a higher resolution, for FOV angles below a given threshold. (The FOV in use is shown on the Dashboard's Graphics tab.)

Using normal maps with vehicle entity models

Most entity models in the MVRsimulation military vehicle library have normal maps. Because normal maps use a great deal of video memory, you must explicitly enable their use, on a per-model basis, in the ModelMap.ini file. Add the `-normalMap` command to an entry to enable normal maps. For example:

```
1 2 225 20   5   *   * Vehicle A-10C.US.grey.hpy  -normalMap
```

Suppressing a model from loading at startup

You can direct VRSG to not load an entity model at startup, and only load it if the entity model appears in a simulation, by adding the command `-delayedLoad` to the entity's line in the ModelMap.ini file.

When a model in ModelMap.ini is marked with `-delayedLoad`, VRSG will not pre-load it at startup, thus the model will not consume any resources. If the model presents itself in the simulation, VRSG will momentarily pause while it loads the model. This command is useful if a model is not likely to appear in a simulation, and when the model does appear, a delay in the visualization is acceptable. If new models are added to a simulation at scenario start time, again there will be a short delay while VRSG loads the models.

Adding contrails, dust trails, or exhaust effects

You can add contrails to missiles or aircraft by adding additional keywords to the ModelMap.ini entry. You can also add dust trails to ground-based vehicles in a similar manner.

To assign a particle-based contrail to an entry in ModelMap.ini, add the command:

`-contrailEffect=<`*`effect_name`*`>`

For example:

```
1 2 225 1 9 * * Vehicle F18_RAAF.hpy -contrailEffect=contrail.par
```

In the example above, the default contrail contrail.par will be added to the F18_RAAF.hpy model. MVRsimulation provides the file \Effects\contrail.par as a default contrail model. You can edit this file or create different versions of it for different types of aircraft or missiles.

To add a dust trail to a ground-based entity, use the command `-dustEffect=<effect_name>`. MVRsimulation provides the file Effects\dust.par as the default dust trail model.

Entities can have multiple contrails or dust trails. To add multiple contrails or dust trails to a model, simply repeat the command `-contrailEffect=<effect_name>` multiple times. The particle system name may be followed by an optional origin offset, to specify the origin of particle emission.

The following example attaches two dust effects to the M1A2 model, one for the right track and one for the left track:

```
1 1 225 1 1 * * Vehicle M1A2.hpy -dustEffect=dust.par -3.2,-1.3,0
-dustEffect=dust.par -3.2,1.3,0
```

Note the `X,Y,` and `Z` offsets for the particle system must follow the particle system filename, separated by commas with no spaces as shown above.

In order for the contrail or dust trail to emit particles, the following two conditions must hold:

- The entity is moving.
- The DIS trailing effects bits are set in the appearance mask of the EntityStatePDU.

The CIGI protocol has its own mechanisms to set the appearance mask. See the appendix "CIGI Version 4.0 Support" for details.

To add an exhaust particle effect to an entity, use a command of the following form:

```
-exhaustEffect=<some_particle_system>
```

The following example is an entry for the frontmost vehicle shown in the scene above:

```
1  1 225  0  0  0  26 Vehicle MTVR-MK23.M2.US.desert.hpy
-exhaustEffect=exhaust-MTVR-MK3.par -tracks(0.000000,0.000000)
-dustEffect=dust.par 0.000000,0.000000,0.000000
```

An exhaust particle system will generate particle emissions when the entity's power-plant appearance bit is turned on.

In cases where you need to use a single exhaust effect for multiple entities, the .par file need not have the origin defined. Instead, you can use a single effect and define the origin right after the effect listed in ModelMap.ini entry. For example, the following entry produced the example that follows it:

```
Sarvatra.IN.desert.hpy -exhaustEffect=exhaust_generic.par
3.17,0.57,-3.29 -exhaustEffect=exhaust_generic.par 3.17,-0.57,-3.29
```

You can add two exhaust particle effects to an entity. For example:

```
1  1 225  0  0  0  26 Vehicle modelfile.hpy modelname -
exhaustEffect=exhaust-LCVP-left.par -exhaustEffect=exhaust-LCVP-
right.par -tracks
```

Adding wakes

VRSG's 3D ocean simulation automatically generates wakes for entities that are assigned DIS domain 3 when the Enable Wakes option is selected on the Oceans tab on the VRSG Dashboard.

For modeling the wakes of sea vessels on non-ocean bodies of water (lakes, rivers, and so on) 85 meters and higher above sea level, using VRSG's 2D legacy water you can add a polygonal wake effect. To assign the default polygonal wake effect to a model, add the `+wake` command to the ModelMap.ini entry. For example:

```
1 3 225 * * * * Vehicle Fishing-Vessel.US.grey.hpy +wake
```

To customize the wake effect for a particular vessel, you use the `+trail` command instead, with the following syntax.

```
+trail( numPts, sampleInterval, color, initialWidth, incrWidth,
offset
```

- `numPts` defines the number of data points (vertices) to use to construct the wake.
- `sampleInterval` defines how often the model's position is sampled, in seconds. `sampleInterval` and `numPts` combined define the length of the wake, and its fidelity (degree of quantization). The values for the default wake are 100 and 1.0 respectively. Slower moving entities can be sampled at a larger `sampleInterval`.

- `color` defines the color of the wake. You provide this value in packed hexadecimal notation in the form AARRGGBB. For example, the value ff0000ff creates a blue trail. The value of the default VRSG wake is c8c8c8c8 which means 200 for alpha, red, green, and blue.
- `initialWidth` defines the width in meters of the wake at its head. The default wake uses a value of 5.0 meters.
- `incrWidth` defines how fast a wake grows in width. The width of a wake at `L` meters linear distance from the head is given by `initialWidth + L*incrWidth`.
- `offset` defines where the wake begins along the model's forward (X) axis. For sea vessels a wake typically begins at the center of the entity's mass.

For example:

```
1 3 225 * * * * Vehicle LCAC.US.grey.hpy
+trail(100,1.0,fac8c8dc,5.0,0.5,10)
```

Although the +wake command is ignored when the 3D oceans Enable Wakes option is turned on, you might want to use this command for all sea entities to ensure your scenario can run with wakes trailing sea vessels if 3D oceans are turned off. (An example of this can be found in the KismayoAmphibious scenario that is installed with VRSG.)

Adding track or wheel impressions

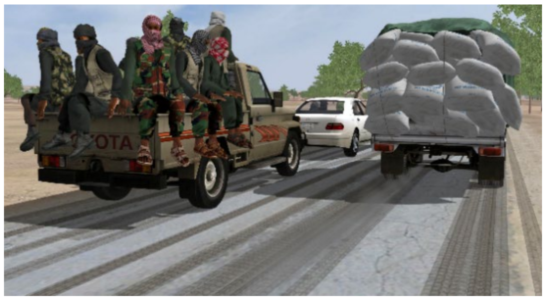
You can direct VRSG to simulate a track or wheel impression to appear behind a tracked or wheeled vehicle entity -- or a footprint impression to appear behind a character entity – as those entities are in motion. To enable the impression, add the command "-tracks" to the model's entry in ModelMap.ini.

For example:

```
1 1 225 1 1 0 0 Vehicle M1A2.US.desert.hpy -tracks
```

For most tracked and wheeled vehicles, VRSG can automatically determine the width and offset of the tracks by inspecting the model's geometry, and will use a generic track or wheel texture for the impression. You can override the default texture and specify another texture by adding the command `-trackTexture=` in the ModelMap entry.

```
1 1 225 1 1 0 0 Vehicle M1A2.US.desert.hpy -tracks
       -trackTexture=my_tracks.rgb
```

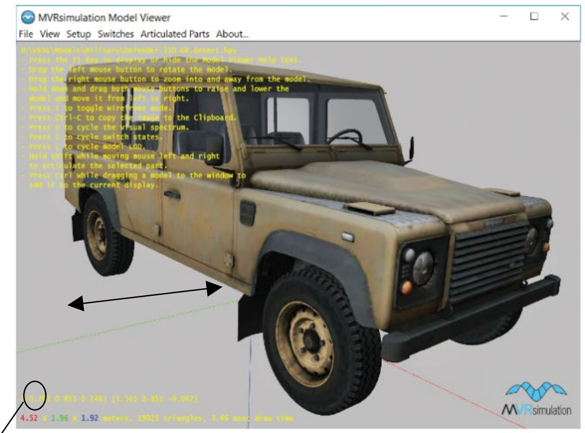


Several textures are delivered with VRSG that can be used for track, wheel, and footprint impressions; the textures are installed in the directory \MVRsimulation\VRSG\Textures. Examples of these textures include treadmark_wheeled.tex, track-human-foot.tex, and track-animal-hoof.tex.

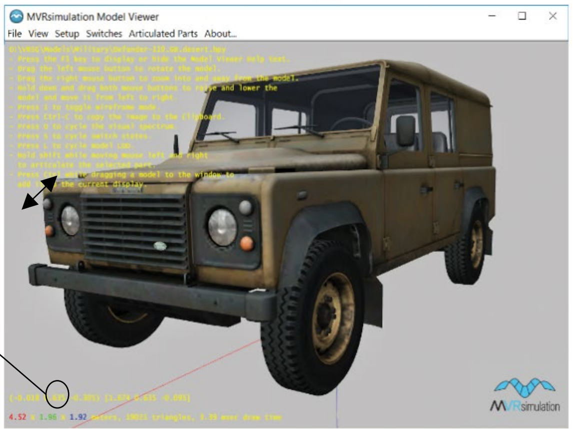
If you encounter an issue with the width and offset that VRSG computes for the track or wheel impression, you can explicitly specify the outer and inner extents of the tire with `-tracks(innerEdge,outerEdge)`. This example defines tracks for the right front tire:

```
1 1 225 6 1 0 0 Vehicle M1035.US.desert.hpy -tracks(1.075,0.75)
```

You can measure extents of the wheels or tracks of the model in Model Viewer by positioning the cursor at the outer edge of the right front tire, noting the value and then positioning it at the inner edge of the tire.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

```
D:\VRSG\Models\Military\Defender-110.GB.desert.hpy
- Press the F1 key to display or hide the Model Viewer help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
```


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

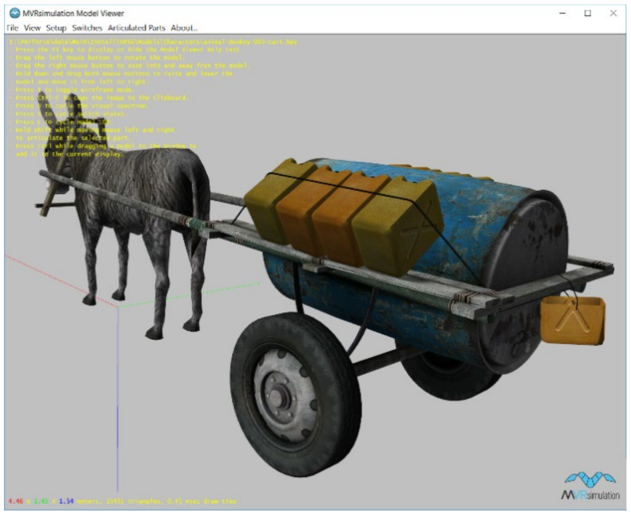
Positioning the cursor at the outside edge of the wheel and then at the inside edge to obtain the values for the tire impression.

Use the `-trackOffset` command to control the track's origin along an entity's x-axis. This command is useful in cases where you want to explicitly control the start position for rendering tracks, such as when an entity is attached to another entity and you want to show the tracks of the trailing entity. An example of this can be seen in scenarios that are delivered with VRSG that take place on the Afghanistan terrain, where pattern of life activities on the street show donkeys pulling carts. The donkey and cart are built as a single model, and VRSG must be directed where to begin the wheel tracks.

The following ModelMap.ini entry controls the tracks made by the wheels of the cart:

```
3   1   0   0  64   0   5 Human animal-donkey-001-cart.hpy
-tracks(1.400000) -trackTexture=track-donkey-cart.tex
-trackAlpha=0.700000 -trackOffset=-2.500000 -delayedLoad
```

In this example, the offset begins 2.5 meters behind the model's x-axis origin, which again you can determine in the Model Viewer, as shown earlier.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

For entities that create a single impression, such as motorcycles or human or animal footprints, use the following single-track syntax:

```
-tracks(width)
```

For example:

```
3  1 225  0  0  0  56 Human human-somalia-002.hpy -tracks(0.300000)
       -trackTexture=track-human-boot2.tex
```

A single track impression of the given width is created behind the entity, and centered on the entity.

The intensity of a track impression can be modulated with:

```
-trackAlpha=N
```

Where N should be a value greater than zero and less than 1. For example, a value of 0.5 would reduce the track intensity by 50%.

Track impressions persist for an hour, or until they consume a certain amount of resources.

Attaching a model to another model

You can attach a dynamic moving model to another dynamic moving model via a ModelMap entry, with the `-attachModel` command. This command is useful for cases such as attaching radars or vehicles to ships, or drivers or pilots to vehicle entities. Currently one attached model is supported per ModelMap entry.

This attachment method improves on the older method of attachment via an entry in the cultural feature file (which is still useful for attaching a model to a model that is in the cultural feature file and described later in this chapter).

To place a human character inside a vehicle, specify the model and the intended animation (driver, passenger, gunner). The \MVRsimulation\VRSG\Animations directory contains a set of driver, passenger, and weapon-holding animations for characters added to vehicles. The placement of the character inside the vehicle is handled by the BVH animation. For example:

```
* * * * * * * vehicle M1163.US.desert.hpy -attachModel= human-us-
soldier-040.hpy -bvh=M1163_Driver.bvh
```

To attach a vehicle to another vehicle (such as a towed howitzer as shown below), you must specify an attachment offset. For example:

```
* * * * * * * vehicle M998A1.US.desert.hpy -attachModel=
M777.US.desert.hpy -attachOffset=-8.2,0,0 -appearance=100000
```

The attachment offset is expressed as:

`-attachOffset=` *`x,y,z`*

where *x, y, z* are the intended coordinates of the attachment point on the attached/parent model. To obtain the coordinates of the attachment point, open the model in the Model Viewer and place the cursor at the intended attachment point. Press the "P" key on the keyboard. This action copies the x, y, z coordinates of the cursor position to the Windows Clipboard. (See the chapter "Previewing Models, Effects, and Terrain" for information about the Model Viewer.)

You can also specify an appearance via the -appearance command in the entry. (Many artillery pieces require an appearance to put them into the towed state.) In the ModelMap entry, when "-appearance=" is placed before "-attachModel", the appearance applies to the parent model; otherwise the appearance applies to the child model.

When the parent entity is destroyed, the attached entity (vehicle or human) is no longer displayed.

Ground clamping and ocean clamping

You can force certain models to be ground-clamped by adding the `-groundClamp` command to their entries in the ModelMap.ini file. If ground clamping is enabled on the Dashboard's Preferences tab, all enumerations are ground clamped, and this command is not necessary. By using the `-groundClamp` command in the ModelMap.ini, and leaving ground-clamping unselected on the Preferences tab, you can restrict ground clamping to particular enumerations. For example:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.desert.hpy –groundClamp
```

VRSG automatically ocean-clamps any entity whose DIS-enum is in the surface domain. Use the `-oceanClamp` command in the ModelMap.ini for entities in other domains to have them float/bob in the ocean waves.

Highest-elevation clamping

You can clamp an entity to the highest elevation at a given X-Y location by adding the `-roofClamp` command to its entry in the ModelMap.ini file. This command is useful for forcing characters to clamp to rooftops in scenarios that involve JTAC or sniper activity.

Disabling a model's orientation clamping

To disable orientation clamping on a per-model basis, use the `-noorientclamp` command. This command selectively overrides the global Ground Clamp Orientation setting on the Dashboard's Preference tab. An example use case might be a wind sock model, where it is desired that the model is aligned vertically with the gravity vector, not with the terrain surface normal. For example:

```
5 1 225 47 1 2 0 Vehicle windsock-001-white.hpy –noorientclamp
```

In the above example, VRSG would honor the simulation's provided pitch and roll for the model, and not override to make them conformal to the local terrain surface.

Specifying a model's timeout period

A global DIS timeout period is specified on the Advanced Startup Parameters dialog box. If an EntityStatePDU is not received from the entity in the given period, that entity is removed from the simulation. This global timeout period can be overridden on a per-entity basis using the "`–timeout`" command. For example:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.desert.hpy –timeout=20
```

In this example, the M1A2 model will timeout after 20 seconds, overriding the global timeout settings for all other entities.

Scaling models

You can change the rendered size of a model by providing the `-scale` command. The following example would render the M1A2 model at 120% its normal size:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.desert.hpy -scale=1.2
```

The `-scale=` command applies a scale factor regardless of range. To scale a model as a function of range, you use the `-scaleFunction=` command. Defining the scale of a model as a function of range is useful in compensating for the limited acuity associated with computer-generated graphics. The `-scaleFunction=` command takes four parameters in the following form:

```
-scaleFunction=r1,s1,r2,s2
```

where r1 is the range where the scale factor s1 is applied; for ranges below r1, the scale factor of s1 is used; for ranges beyond r2, the scale factor s2 is used. For ranges between r1 and r2, the scale factor is linearly interpolated between s1 and s2. Note r1 must be less than r2. For example, if the model is acceptable for viewing out to 1km, but you want the model to scale slowly in size until it reaches two times its normal size at 4km, you would use the `-scaleFunction` command as shown:

```
-scaleFunction=1000,1,4000,2
```

Scaling a model's LOD switch points

You can scale the ranges of a model's LOD switch points using the `-lod_scale` command. The following example would scale the model's switch points by 200%:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.desert.hpy -lod_scale=2.0
```

Note that this command can be used in the ModelMap.ini or in a cultural feature file (vrsg.clt) as described later in this chapter.

Forcing a model to display at a lower-detail LOD

In some cases you might want to limit the highest level of detail displayed for a given model. Doing so may be useful if you are using older hardware not capable of rendering a given model at sufficiently high frame rates. The `–minLODRange` command prevents a model's computed LOD range from going below a minimum value, which keeps the model from using the (highest) levels of detail associated with LOD ranges below that minimum. For example:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.desert.hpy –minLODRange=2000
```

In the above example, the model would never be displayed at a finer level of detail than the 2km or greater representation. You can use MVRsimulation's Model Viewer utility to determine a model's LOD switch points.

Scaling a model's RADAR return intensity

The RADAR return intensity for a given model can be uniformly scaled using the `-radarReturnScale` command. For example:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.desert.hpy –radarReturnScale=0.5
```

In the above example, the model's RADAR return intensity would be reduced by 50%.

Specifying the model type for CIGI

CIGI does not use the 7-tuple DIS enumerations to identify a model; instead, CIGI uses a scalar integer namespace to identify the type of model. You can specify the entity type identifier by using the `-type=` command. For example:

```
1 1 225 2 1 0 0 Vehicle M1A2.M2.US.desert.hpy -type=12
```

In the above example, a CIGI host would use 12 in the entity type field of the Entity Control packet to refer to the M2A3 model.

When you use CIGI, consider providing a valid DIS enumeration for the model. Doing so provides additional documentation and enables VRSG to determine the model's domain (land, sea, air).

Adding additional appearance bits

You can add additional appearance bits to what an entity is providing in the DIS Entity State PDU by using the `-appearance=` command. The value provided will be logically OR'ed with the appearance bits being transmitted by the entity. This feature is useful for turning on certain switch states for a model. An example use-case would be the power-plant status bits (0x400000). Most MVRsimulation aircraft models use this bit to animate props or rotors. If a simulation is not stimulating this bit when the vehicle's engine is on, you could force this bit on with the following command:

```
1 2 225 6 1 0 0 Vehicle AH-64A.US.green.hpy -appearance=400000
```

The value provided must be in hexadecimal, without the preceding "0x" notation.

VRSG supports 64-bit appearance masks, which enables advanced features of newer MVRsimulation models to be assigned to bits that are not used by SISO/DIS.

You can find and identify the appearance bits for an MVRsimulation model's switch states when you load the model in the Model Viewer, as described in the chapter, "Previewing Models, Effects, and Terrain."

Specifying the initial state of an articulated part

Specify the initial state of an entity's articulated part with the `-setPartValue` command. This command enables you to configure the degree-of-freedom (DOF) for an articulated part.

The syntax of the command is:

```
-setPartValue(partId, value)
```

partId is the ID from the set of DIS protocol enumerations, which are described in the document available in the \MVRsimulation\VRSG\Models directory (SISO-REF-010-2023 Enumerations v31.pdf). For example, part ID 4096 is the primary turret, 4416 is the main gun, and so on.

partId identifies the part and the degree-of-freedom being controlled. The value you provide for *partId* is the part code plus the DOF identifier taken from the following table:

Degree of freedom	**Code**
X translation	5
X translation rate	6
Y translation	7
Y translation rate	8
Z translation	9
Z translation rate	10
Azimuth	11
Azimuth rate	12
Elevation	13
Elevation rate	14
Roll	15
Roll rate	16

For example, to initially set a model's turret (4096) azimuth angle (11) to 1.5 radians, add 4096+11=4107, then use the command `-setPartValue(4107,1.5)`.

You can use multiple `-setPartValue` commands in one entry in ModelMap.ini to set the values for multiple parts of a given entity.

The `-setPartValue` command is useful for initializing an articulated part to some reasonable behavior before the simulation controls the part (or in case the simulation does not control the part). As soon as the simulation updates the part, it takes over control. By using this command you can get radar dishes spinning, guns pointed in a certain direction, turrets initially positioned, and so on.

Replacing a destroyed culture model with another model

A cultural feature model can be replaced with an alternate model to show its destroyed state. For example, you could replace a destroyed building model with a model of a pile of rubble. The alternate model will be automatically scaled to approximate the mass of the original model it replaces. This feature can effectively make any building destroyable.

The DIS EntityStatePDU has a field for *alternate entity type*. To use this feature, specify in the alternate entity type field the enumeration of the model to use when the entity becomes destroyed. This alternate entity type enumeration should be mapped to the desired model in ModelMap.ini (such as the model rubble-005.hpy).

This feature applies only to entities of the culture kind (kind = 5). When VRSG sees a culture entity with appearance bits indicating it is destroyed, VRSG will use the alternate entity type to choose the model to display. The alternate model automatically scales appropriately so that its mass is similar to the mass of the original model it is replacing (that is, a large building will be replaced with a large pile of rubble).

Adding a rotor wash effect

You can add a rotor wash particle system to a helicopter entity by specifying the command `–rotorWashEffect= <effect_name>`. This command identifies a particular rotor wash particle system. VRSG will use this particle system to create the rotor wash effect when the aircraft is close to the ground or ocean. MVRsimulation provides the file \Effects\helo_rotor.par as the default particle system for rotor wash. You can create customized versions of this file and map it to different enumerations for different rotor wash effects on different airframes.

For example:

```
1   2 225   0   0   0  26 Vehicle UH-1Y.US.grey.hpy
-rotorWashEffect=helo_rotor.par
```

This next example shows a rotor wash effect on VRSG's 3D oceans.

```
1   2 225   0   0   0  13 Vehicle MH-60R.US.grey.hpy
-rotorWashEffect=helo_rotor.par -delayedLoad
```

Adding light points to a model

You can add light points to a model with the `-lightPoint=` command using the following syntax:

```
lightPoint=X,Y,Z,size,minPixelSize,maxRange,color,appearanceMask,
period,timeOn,azimuth,elevation,horizontalLobeAngle,verticalLobeAngle
```

- The `X, Y,` and `Z` fields identify the light's position in model-space in meters, using the DIS and CIGI convention of `X` forward, `Y` right, `Z` down.
- `size` conveys the radius of the light in meters, and the `minPixelSize` controls the smallest size the light will display as, given in pixels.
- `color` is specified as a hexadecimal value in AARRGGBB format, an 8-digit hex number, where the valid range for each component is 00..FF. The AA byte contains the alpha value, or intensity, of the color. A value of 00 would be fully transparent, and a value of FF would be the color at full intensity. If you use just 6 digits, the alpha (intensity) will be zero.
- `appearanceMask` indicates which bit of the EntityStatePDU appearance field is used to turn on the light. If zero, the light is always on.
- `period` and `timeOn` specify the frequency and duty cycle of flashing lights. For example, for a light to flash at 2 Hz, use a period of 0.5. A `timeOn` of 0.25 of would have the light be on half of its flash cycle. For non-flashing lights, use equal values for `period` and `timeOn`.

For directional lights, azimuth and elevation control the light's direction, given in degrees relative to the entity they are attached to. The `horizontalLobeAngle` and `verticalLobeAngle` fields control the lobe size of a directional light. For omnidirectional lights, all fields past `appearanceMask` may be omitted, and will default to non-flashing omni-directional lights.

This example puts the green light on the tip of the F16 tail:

```
-lightPoint=-5.6,0,-3.4,0.1,6.0,5000,ff00ff00,0,1,1,0,0,180,180
```

To add multiple light points to entities, consider adding the light points via a model attribute JSON file. This method is described later in this chapter in the section "Adding and editing model metadata."

Adding light lobes to a model

A light lobe illuminates anything that falls inside its cone of influence. Light lobes illuminate on entities that have the SISO DIS appearance bit set for headlights on a ground domain vehicle or a flashlight on a human character entity.

When the headlights bit is set for a ground-domain entity, VRSG automatically attaches two light lobes to the front of the vehicle, using the bounding dimensions to determine approximate location of the light origins. By default the lights have an illumination range of 30 meters, meaning their effect will be attenuated to zero at a range of 30 meters from the light origin. Lights will be rendered out to 60 times their illumination range. There are two parts to headlights in MVRsimulation ground vehicle models. The actual headlight bulb on the model is something built into the model. This is emissive geometry so it glows in the dark. If a model supports these, you will see the Headlights option in the Switch menu when viewing the model in Model Viewer. The simulation setting the headlight bit causes the emissive geometry to be displayed for the model. The other part is VRSG's automatic attachment of light lobes to ground-domain entities that have their headlight bit set. (A model need not have emissive headlights built into it for the light lobes to work, but it looks better when it does. Otherwise the scene becomes illuminated from no clear source of light.)

For a human character, the light lobe is placed at the handheld-item's "weapon" location, and oriented along the forward axis of the weapon. The weapon type is not enforced. If the character holds a flashlight and has the flashlight bit on, a light lobe will be attached to it, (independent of what the actual "weapon" the character is holding).

Note: A light lobe can be attached to an entity via CIGI, using MVRsimulation's component control for this purpose. A light lobe can also be attached to a static model or an entity via a model attribute JSON file, as described later in this chapter in the section "Adding and editing model metadata."

Summary of available commands for ModelMap.ini

The following commands can be added to a ModelMap.ini entry to modify the behavior of the loaded model:

Command	**Description**
-appearance=*x*	Specifies additional appearance bits to be OR'ed with the model's provided appearance.
-attachModel=	Attaches a dynamic moving model (entity) to another, via an attachment point.
+contrail	Adds a contrail (EntityType *must* be domain 2, air)
-attachOffset=*x,y,z*	Specifies the coordinates of the attachment point, measured in the coordinate system of the entity being attached to. Values are in meters, with the DIS/CIGI axis convention of *x*=forward, *y*=right, *z*=down.
-contrailEffect= *effect_name*	Specifies a contrail effect to associate with an aircraft or missile model.
-darkenRange=	Darkens the color of air-to-air targets as a function of range, to make models detect-able against the sky background. Range given specifies the range at which the model will become completely black. Up to this range, the model is incrementally darkened using a non-linear function of range.
-delayedLoad	Directs VRSG to not pre-load the model at startup; only load the model if it appears in the VRSG session.
-dustEffect= effect_name	Specifies a dust effect to associate with a ground-based model.

Command	**Description** *(continued)*
-flameEffect= *effect_name*	Assigns a custom effect for a flaming vehicle.
-forceShadow	Produces a ground shadow for aircraft similar to the planar shadows for ground vehicles.
-groundClamp	Ground clamps all instances of this model type.
-launchFlash=*radius*	Automatically attaches an omni-directional light lobe to a munition entity when the entity has the launch flash appearance bit set (bit 16, 0x10000). A light lobe is created to illuminate the scene around the entity by the given radius. Useful for night time scenarios where it is desired that missile launches illuminate the surrounding scene.
-lightPoint=	Adds a light point to a model; see above description for details.
-lod_scale=*N*	Scales a model's LOD switch point ranges by the ratio given by *N*.
-material=*N*	Assigns an explicit material code to all polygons in a model; for physics-based IR simulation.
-minLODRange=	Places an upper-bound on the highest LOD displayed for a given model.
-nocolldet	Makes models of this type invisible to collision detection. VRSG will not consider the model when testing collision segments against the scene. Useful to apply on bullets, tracers, or other types of entities that VRSG should not waste cycles determining collisions against.
-noorientclamp	Disables orientation clamping of models on a per-enumeration basis.
-noSpecular	Disables specular lighting effects for a given model.

Command	**Description** *(continued)*
-oceanClampOffsetZ=Z.zz	Offsets the position of an ocean-clamped entity upward by this amount, in meters. Positive values cause a sea vessel to "ride high" as if it were lighter than usual, and negative values will cause it to "ride low" as if it were heavily loaded.
-oceanDynamicsScale=S.ss	Scales the responsiveness of an oceanClamped entity's dynamics. Values larger than 1.0 will make the entity's dynamics more responsive; values smaller than 1.0 will dampen the entity's dynamic response. Small vessels have larger values than large ones.
-oceanDisableBowParticleEffect	Disables the bow wake splash effect.
-radarReturnScale=	Modulates a model’s Radar return intensity.
-roofClamp	Elevation clamps a model to the highest point at a given location, which can include culture models. Useful for clamping human characters to rooftops.
-rotate=yaw,pitch,roll	Applies an additional rotation to a model before rendering. Angles are given in degrees.
-rotorWashEffect=*name*	Specifies a rotor wash effect to apply to a helicopter model.
-rubble=	Specifies an alternate model to use for the destroyed state. Useful for buildings as most do not have an explicit destroyed state. They can specify a rubble model from the model library. The rubble model will be automatically scaled to have similar mass as the model it is replacing.
-scale=*N*	Scales the model by the ratio given by *N*. For example, a value of 2 will double the model's size.

Command	**Description** *(continued)*
-scaleFunction=r1,s1, r2,s2	Defines scale for a model as a function of range; parameter r1 is the range where the scale factor s1 is applied. For ranges below r1, the scale factor of s1 is used. r2 is the range where the scale factor s2 is applied. For ranges beyond r2, the scale factor is limited to s2. For ranges between r1 and r2, the scale factor is linearly interpolated between s1 and s2 (r1 must be less than r2).
-setPartValue(*partId*, *value*)	Sets the initial DOF rate or angle for an articulated part of a model.
-shadowOffset=	Specifies a vertical offset for projected planar shadows. Normally they project to the model's Z=0 plane. This setting can be used to force shadows to a different point along the model's Z axis.
-skinnedGloves	Associates the specified character model with the gloved hand models with articulated fingers (skinnedGloveLeft-01.hpy and skinnedGloveRight-01.hpy).
-skinnedHands01, -skinnedHands02	Associates the specified character model with the ungloved hand models with articulated fingers (skinnedHandLeft-01.hpy, skinnedHandRight-01.hpy, skinnedHandLeft-02.hpy, and skinnedHandRight-02.hpy).
-smokeEffect= *effect_name*	Assigns a custom effect for a smoking vehicle.
-terrainModel	Marks a model to be considered for elevation lookup queries.
-timeout=	Overrides the DIS timeout period for an entity.
-trackAlpha=*N*	Modulates the track intensity by the given value (e.g. 0.5 for 50%).
-trackOffset=*N*	Controls the track's origin along an entity's x-axis.
-tracks	Creates a track or wheel impression behind a moving ground entity.
-trackTexture=t	Used with –tracks, specifies a custom texture to use for a track or wheel impression.

Command	**Description** *(continued)*
-translate= *x,y,z*	Moves a model's origin to a new location.
-type=	Specifies an entity model type identifier for the CIGI protocol.
-uvw=	Assigns explicit thermal properties to a model.
+wake	Adds a wake to a sea vessel model on VRSG's 2D legacy water surface. (EntityType *must be* domain 3, surface vessels)
-wakeAmplitudeScale= S.ss	Scales the amplitude of wakes behind the entity in 3D oceans. An S value of 2.0 makes wake-waves twice as tall, and 0.75 makes them 75% as tall.

Examples of using the commands for handling entities on VRSG's 3D oceans can be found in the Kismayo Amphibious scenario that is installed with VRSG.

Troubleshooting a missing entity

You can capture a snapshot (written to an ASCII file) of all dynamic entities in the current scenario rendering in VRSG. This file is useful for debugging purposes and for identifying any issues in your Modelmap.ini file. Press Shift-F4 to capture all the entities with their enumeration and model ID, at their current position and orientation. The ASCII file is saved in VRSG's installation directory as dynamic_models_snapshot.clt. Each dynamic model is enumerated with its DIS ID, 7-digit DIS entity type, position, orientation, name of 3D model in use, and DIS appearance mask.

If an entity is not mapped to any entry in the Models\ModelMap.ini file, it will default to the beach ball model designation described at the end of the file, and display the beach ball model:

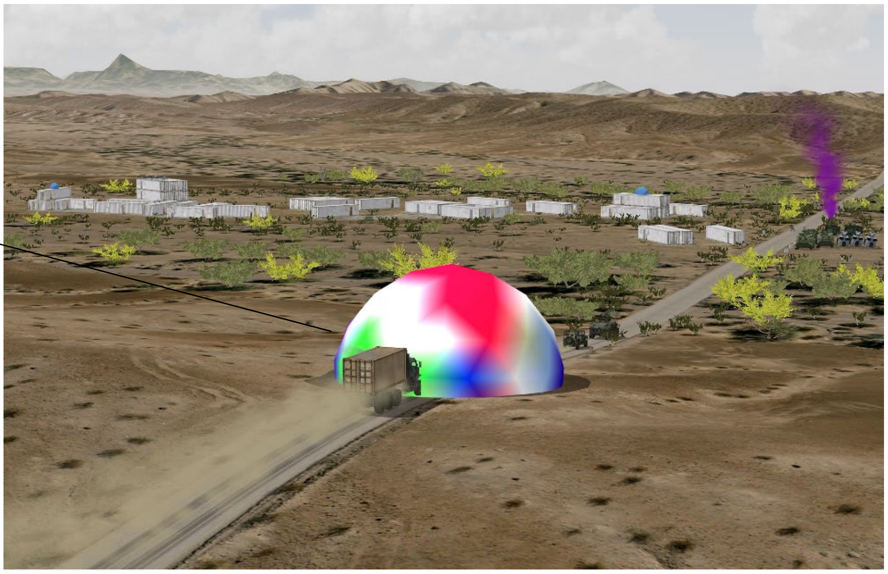
```
* * * * * * * Vehicle beach_ball.hpy
```

The beach ball default model (shown in the examples on the next page) is helpful for resolving entity enumeration issues. For example, suppose you identified in the onscreen Help the enumeration of an unmapped entity. You could edit the \Models\ModelMap.ini file to make an association for the enumeration 1:1:0:0:0:0:0 that maps to either a vehicle in the list or to a new model you have added.

You can edit the ModelMap.ini file while VRSG is running; you do not need to terminate your VRSG session to edit the file. Edit the file while VRSG is running and then, press Ctrl-M to have VRSG reload the ModelMap.ini file.

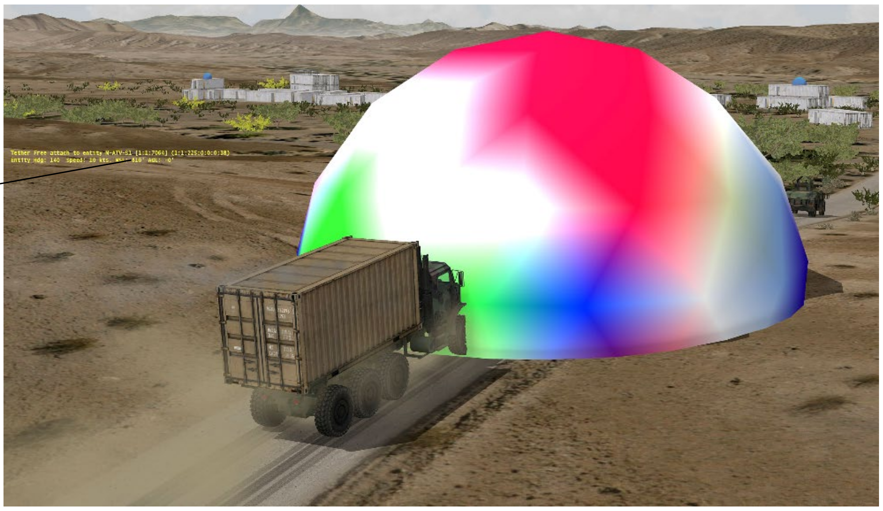
MVRsimulation strongly recommends that you do not remove the beach ball mapping from the end of the ModelMap.ini file. If VRSG cannot resolve an enumeration to a visual model, the entity will be ignored. Having the beach ball entry ensures that a visual indication of the missing entity will be rendered, as shown below.

Indicates an unresolved enumeration / missing entity.



The beach ball in the scene above indicates that a vehicle entity is missing from the convoy. When you attach to the beach ball, the attachment message indicates the entity mapping in the Modelmap.ini:

Enumeration of unresolved entity.



```
Tether Free attach to entity M-ATV-S1 [1:1:7064] (1:1:225:0:0:0:38)
Entity Hdg: 140  Speed: 10 kts. MSL: 810' AGL: -0'
```

Checking the VRSG error log (VrsgError_<*machine_name*>.txt) is another way to identify the missing model:

```
Failed to locate model file M-ATV-SXB-CAGE.M153.US.desert.hpy
```

If you do not see DIS entities in the VRSG scene, first check the All Entities list box on the Dashboard’s Attach Options tab. If this list box is empty, VRSG is not seeing your DIS entities on the network. This could be due to a variety of possible reasons:

- You are using BSI MACE and your MACE mission has not been started.
- VRSG and the simulation are not in agreement on exercise ID. In VRSG, you can set the exercise to 0 to allow all exercises.

- The DIS UDP ports are mismatched. The VRSG UDP setting (default setting is 3000) must match the UDP setting used by the simulation.
- The Site and Host (Host is referred to as “App” in MACE) values must not be the same in VRSG and the DIS simulation. At least one of those values must be unique. For example, VRSG can be Site 1 / Host 101 and MACE can be Site 1 / App 100, or VRSG can be Site 1 / App 100 and MACE can be Site 2 / App 100 and. But the values of both settings cannot be identical.
- The networked machines are not physically reachable. Try pinging one machine from the other.
- The Windows firewall is blocking VRSG, the DIS simulation, or both applications.

Creating threat domes

In the ModelMap.ini file, you can use the model class `Bubble` or `Cylinder` to create a wire-frame threat dome that represents the detection and lethal ranges of a Surface to Air Missile (SAM) or similar threat system.

In the ModelMap.ini file, map the enumeration using the model class `Bubble` or `Cylinder` and five parameters that describe the horizontal and vertical radius of the dome in meters, and the color of the dome in terms of the red, green, and blue components of the intended color. The expected ranges of the color values are 0 (least intensity) to 255 (maximum intensity).

```
Bubble horizontalRadius verticalRadius red green blue
```

The following example creates a yellow threat dome with a horizontal radius of 1500 meters, and vertical radius of 1000 meters:

```
9 1 222 * * * 1 Bubble 1500 1000 255 255 0
```

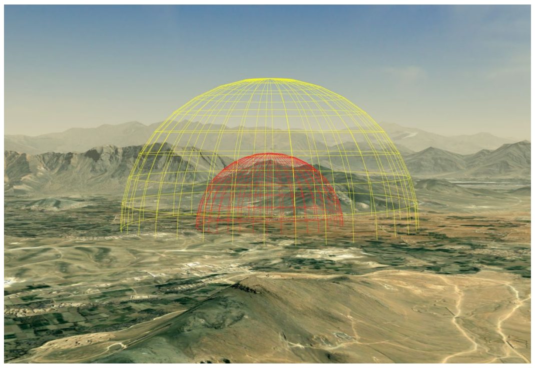
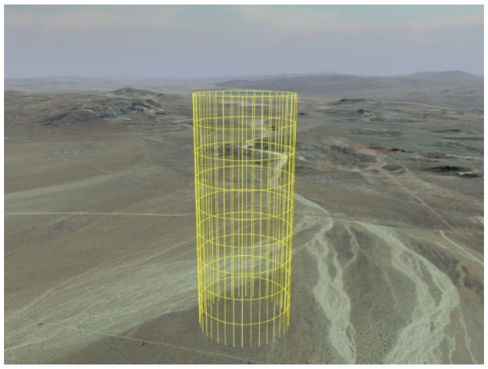
This next example creates a red threat dome with a horizontal radius of 500 meters:

```
9 1 222 * * * 2 Bubble 500 1000 255 0 0
```

To create a cylinder variation of a threat dome, use the keyword `Cylinder` instead of `Bubble`. For example:

```
! Crotale_MEZ
5    0    0    6   7   0    22 cylinder 10193 5854 150 100 0
! SA11_MEZ
5    0    0    6   7   0    20 cylinder 30390 21340 200 100 50
! ZSU234_MEZ
5    0    0    6   7   0    24 cylinder 2052 3659 200 0 150
```

A threat dome does not have to be associated with a DIS entity. You can also place a dome as a static model in a cultural feature file (vrsg.clt) as described later in this chapter.

Mapping animations to 3D animated characters

This section describes how animations are associated with VRSG's 3D animated characters. The VRSG release provides a large set of pre-built animations which should accommodate the most common uses, so it is not likely that you will need to edit the animation configurations.

VRSG's animation files, which are in the BVH motion capture format, are located in the Animations subdirectory. BVH animation files are ASCII, but are generally not edited by hand. A third-party tool, such as Alias MotionBuilder, is used to author BVH animations.

About entries in the Animation.ini file

The mapping file Animations\AnimationMap.ini is used to associate DIS lifeform entity appearances with animation clips. Each animation clip is stored in a separate file in the Animations folder with the .bvh extension. The syntax of an AnimationMap.ini entry is:

```
<mask> <value> <bvh-file-name>
```

The mask is a 32-bit hexadecimal number that VRSG uses to select which bits of the DIS EntityStatePDU appearance are relevant for choosing the animation. The value is a 32-bit hexadecimal number that matches the result of the bitwise AND of the mask and the appearance mask. For example:

```
0x00f0000 0x0010000 standby.bvh
```

In this example, the mask selects the 4 bits of the appearance mask that correspond to the life form state of the entity. If the lifeform state bits are 1, the entity is indicating it is in a standing posture. The above entry maps this result to an animation for standing.

Entries in AnimationMap.ini are processed top-to-bottom, so when a match is found that animation is used. Therefore more specific masks should be placed towards the top of the file, and more general ones towards the bottom of the file. For example, an entry that looked at both life form state bits and primary weapon state bits should precede an entry that looks at life form state only, for example:

```
0x30f0000 0x2010000 standby_rifle_deployed.bvh
0x30f0000 0x3010000 standby_rifle_firing.bvh
0x00f0000 0x0010000 standby.bvh
```

This example shows a correct order. If the third entry were first instead, then VRSG would catch all primary weapon states since the mask does not consider these bits. This would cause the deployed and firing animations to never be engaged.

This example loads standby_rifle_deployed.bvh, standby_rpg_deployed.bvh, and standby_pistol_deployed.bvh. Any other animation files in the Animations directory that matched the regular expression would be loaded for that single entry.

When VRSG finds an entry in AnimationMap.ini that is appropriate for the given appearance mask, VRSG scans the set of animations given for that entry looking for one that matches the type of weapon the character entity is using. If the entity in the above example were using a weapon of type 'rifle', the animation standby_rifle_deployed.bvh would be chosen. Similarly, if the entity used a weapon of type 'pistol', standby_pistol_deployed.bvh would be chosen.

Different types of weapons generally require unique animation clips to support them. For example, firing a rifle requires different hand positioning than firing a pistol or an RPG. VRSG addresses this requirement by allowing multiple animations to be loaded for a given entry in AnimationMap.ini. The animation names given in AnimationMap.ini can contain an asterisk (*) as a wildcard to represent multiple weapon types. For example:

```
0x30f0000 0x2010000 standby_*_deployed.bvh
```

The type of weapon that is used for selecting animations is stored in the weapon model file. For example, the first line of an HPX file contains the command *–weaponType=<name>* to identify the type for the weapon model. If this command is absent, a type of 'rifle' is assumed. As an example, the header line of an RPG model would appear as follows:

```
HPXV002 –weaponType=rpg
```

VRSG does not actually enforce the weapon type that is stored in the HPX file, although the types 'rifle, 'rpg, and 'pistol' are used by convention. Instead, VRSG uses this label to find the appropriate animation by substituting the weapon type in the animation file names.

This information is important if you plan to add weapons and/or animations to VRSG using models you have converted to HPX models. The weapon models provided with VRSG have already been correctly assigned their weapon type and require no user intervention.

You can have VRSG randomly select an animation from a group of animations by listing them all in one appearance mapping entry in the Animation.ini file. This can be useful if you have a set of character animations performing essentially the same action but slightly differently. For example, you could create an appearance mapping entry in the file that mapped an appearance code to standby_talk-001.bvh, standby_talk-002.bvh, standby_talk-003.bvh, standby_talk-004.bvh, standby_talk-005.bvh (all in MVRsimulation's library of animations) and then associate the appearance to several characters standing around talking with each other.

An example of a similar setup can be found in the AnimationMap.ini file itself. VRSG's animation library contains 10 total dead animations of varying body positions on the ground that result from a range of actions like collapsing quickly or falling over from a blast. The AnimationMap.ini lists them all with the deceased appearance code, as shown in the following entry:

```
0x0000018 0x0000018 dead.bvh dead-0.bvh dead-45.bvh dead-90.bvh dead-
135.bvh dead-180.bvh dead-225.bvh dead-270.bvh dead-315.bvh dead-
360.bvh
```

(The animations are named by entity-relative degree heading.) VRSG will pick a .BVH from the list at random. VRSG uses the detonation blast information to determine which direction entities fall over and then maps the corresponding animations. You can also associate an explicit animation by name to a character, to have it fall over in a direction consistent with impact.

Mapping virtual world events

VRSG is delivered with over 175 effects files, located in the \MVRsimulation\VRSG\Effects directory. The \Effects directory contains several kinds of effects:

- Billboard-based effects (.eff) for rotating 2D textures such as simple explosions. VRSG also contains an effect called flames.flm.
- Particle-based (.par) effects for smoke, dust trails, rotor wash, explosions, and so on. VRSG also contains 10 solid particle (.spf) effects.
- Dynamic craters, using VRSG's capability to dynamically deform terrain surfaces to represent craters resulting from munitions impacts.
- Cloud files (named cloud*.cld) for different levels of density and sparseness of legacy clouds.

Effects can be free-standing or attached to an entity. Free standing effects are for explosions, smoke grenades, and so on. With these effects, the particle emitter source stays where it was established; you cannot move it. As the particles are generated over time, they can be affected by changes in wind conditions. With an attached effect, the particle emitter moves with the entity it is attached to. Common use of these effects are contrails, dust effects, and helicopter rotor wash.

To move an effect that is normally free-standing, you must attach it to an entity. The entity need not be associated with a visible 3D model; you could map the model to "empty.hpy" which would display no geometry. But you would gain addressability of the attached effect's location through the entity's position.

Billboard, particle, and solid particle effects (.eff, .par, and .spf) can be previewed in MVRsimulation's Model Viewer as described in the chapter "Previewing Models, Effects, and Terrain." They can be triggered by a detonation DIS PDU, added to the ClientMap.ini file (described below) for use with a CIGI event or a scenario created in Scenario Editor, or dragged directly onto the terrain in the VRSG visualization window.

The mapping information that in turn controls which effect is mapped to the appropriate DIS enumerated event is contained in the initialization files FireMap.ini and DetMap.ini, located in the \MVRsimulation\VRSG\Effects directory.

You can edit these two initialization files to change the defaults. To examine how a DIS event such as a fire or detonation effect is generated in the virtual world through the relationship of the FireMap.ini, DetMap.ini and the *.eff files, consider the contents of the following two mapping initialization files:

FireMap.ini

```
! FireMap.ini
! Copyright 1997 - 2025 MVRsimulation, Inc. All rights reserved.
! URL: www.mvrsimulation.com  Email: support@mvrsimulation.com
! This file can be used only with MVRsimulation's Virtual Reality
! Scene Generator (VRSG). The receipt of this file and your use of
! the information contained herein is subject in all cases to your
! agreement to the provisions governing "Additional Materials"
! of the current MVRsimulation, Inc. license agreement found at:
! https://www.mvrsimulation.com/howtobuy/license_agreement_policy.htm
!
! This file allows you to associate munition firings with animation
! sequences. A given warhead type may be mapped to a particular
! special effect file. When a weapon is fired, this mapping file
! is read top to bottom looking for a match, when one is found,
! the associated special effect is rendered.
!
! For any of the numeric fields, an asterisk '*' matches anything.
!
! Syntax of entries:
! kind domain country category subcategory specific extra effect-file
!
! GAU-4 20mm   20 mm
2    2    225   2   2   0    *  muzzle.eff a10-fire.par -muzzleTip
!
! 120mm Tank   120 mm
2    2    225   2   13  0    *  muzzle.eff muzzleBlast.par -muzzleTip
!
!SA-2 Missile  SA-2f Guideline Mod5
2    1    222   1   14  4    *  launchSmoke-medium.par
!
! Default catch-all, should be last entry in file
2 8 225 2 1 5 0 null.eff
!* * * * * * * muzzle_blast.par –muzzleTip
```

The `-muzzleTip` command in the FireMap.ini file directs VRSG to automatically compute the position of the effect at the end of the barrel, and redirect the effect along the gun. This effect is shown in the following image. The particle file used (muzzle_blast.par) has the entry "use_dis_velocity 1" which allows the built-in direction to be overridden.

If a particle system has the "use_dis_velocity 1" entry, the velocity vector in the Fire or Detonation PDU will be used to steer the particles, unless `-muzzleTip` is used. If `-muzzleTip` is used, the computed direction from the model's articulated parts overrides the FirePDU velocity.

DetMap.ini

```
! DetMap.ini
! Copyright 1997 - 2025 MVRsimulation, Inc. All rights reserved.
! URL: www.mvrsimulation.com  Email: support@mvrsimulation.com
!
! This file can be used only with MVRsimulation's Virtual Reality 
Scene
! Generator (VRSG. The receipt of this file and your use of the
! information contained herein is subject in all cases to your
! agreement to the provisions governing "Additional Materials"
! of the current MVRsimulation, Inc. license agreement found at:
!
https://www.mvrsimulation.com/howtobuy/license_agreement_policy.html.
!
! This file allows you to associate munition detonations with
! animation sequences. A given warhead type and detonation
! result may be mapped to a particular special effect file.
! When a detonation occurs, this mapping file is read top to bottom
! looking for a match, when one is found, the associated special
! effect is rendered.
!
! The result column corresponds to the detonation result field of the
! Detonation PDU (See the DIS protocol for details).
! For any of the numeric fields, an asterisk '*' matches anything.
! Syntax of entries:
```

```
! kind domain country category subcategory specific extra result
effect-
! file
!
!gun_round 5.56
2    8    225  2   1  *    *  *   dustPuff-verySmall.par
! 2  8    225  2   1  5    0  *   m16_impact.eff
!
!Smoke Grenade (Hand) M243 smoke grenade
2    9    225  2   41 1    1  *   smokeRed.par
!
!Mk-82
2    9    225  1   14 0    0  *   explosionSmokeDonut-large.par
          explosionSmokeCenter-large.par explosionFlash-large.par
!
!155mm WP
2    9    225  2   14 13   0  *   smokeWhitePhosphorus.par
                          smokeWhitePhosphorus-mushroom.par
!
!SA-14 Missile
2    1    222  1   26 0    0  *   explosionFlash-small.par
                           explosionSmokeAirburst-small.par
!
! Chaff
8    2    *    1   *  *    *  *   chaffPuff.par
!
! entity impact
!* * * * * * * 1 explosion1.eff
! entity proximte impact
!* * * * * * * 2 explosion2.eff
! ground impact
!* * * * * * * 3 explosion_gnd.eff
! air impact
!* * * * * * * 17 explosion1.eff
! water impact
!* * * * * * * 14 bullet.eff
!
! * * * * * * * * explosion_large.eff
!
! Default catch-all, should be last entry in file
*    *    *    *   *  *    *  *   explosionFlash-small.par
                                    explosionSmokeDonut-small.par
```

You can direct VRSG to ground clamp certain detonation effects on a per-enumeration basis. To do this, simply add the `-groundClamp` flag to the DetMap.ini entry following the effect name. For example:

```
2 8 225 2 1 5 0 * explosion_ground.eff -groundClamp
```

The following options can be appended to entries in ClientMap.ini or DetMap.ini to control the size of a detonation's impact on the 3D ocean surface.

Name	**Default value**	**Unit**	**Description**
-oceanImpactMass=	0.0	kilograms	Mass of the item impacting the ocean surface
-oceanImpactVelocity=	0.0	meters/second	Velocity at time of impact on the ocean surface
-oceanImpactDiameter=	0.0	meters	Diameter of the item impacting the ocean surface

ClientMap.ini

The ClientMap.ini, another initialization file also located in \MVRsimulation\Effects, lists all the effects that are available for a user to specify within a network packet. Available effects are indexed in the order they appear in the file. In this case (as with CIGI or scenarios built with Scenario Editor) instead of sending a DIS enumeration, the network packets specify an integer for the effect to be triggered and that integer is used as an index into the list of effects listed in the file. The ClientMap.ini file must match at both the sending and receiving end.

Creating billboard effects

The rendering information for a billboard-based special effect is described in an *.eff file. You can fully control animation effects simply by placing the proper DIS enumeration hierarchy with the required effect. You can also create your own effects.

The following flames.eff file illustrates the format of an effects (.eff) file:

```
16
-3 0 3 -6 0.0667 flame0.tex
-3 0 3 -6 0.0667 flame1.tex
-3 0 3 -6 0.0667 flame2.tex
-3 0 3 -6 0.0667 flame3.tex
-3 0 3 -6 0.0667 flame4.tex
-3 0 3 -6 0.0667 flame5.tex
-3 0 3 -6 0.0667 flame6.tex
-3 0 3 -6 0.0667 flame7.tex
-3 0 3 -6 0.0667 flame8.tex
-3 0 3 -6 0.0667 flame9.tex
-3 0 3 -6 0.0667 flame10.tex
-3 0 3 -6 0.0667 flame11.tex
-3 0 3 -6 0.0667 flame12.tex
-3 0 3 -6 0.0667 flame13.tex
-3 0 3 -6 0.0667 flame14.tex
-3 0 3 -6 0.0667 flame15.tex
```

The first line in the file indicates the number of frames, shown as 16 in the above example. The remaining lines describe each frame of the animation. For each frame, there are typically 6 parameters, with an optional 7th parameter. These parameters are:

- `LLY` is the lower-left Y coordinate of the frame, in meters.
- `LLZ` is the lower-left Z coordinate of the frame, in meters.

- `RY` is the upper-right Y coordinate of the frame, in meters.
- `URZ` is the upper-right Z coordinate of the frame, in meters.
- `Time` is the display duration for the frame, in seconds.
- `Texture` is the texture map to display for the frame.
- `Alpha` is the translucency of the frame. (This parameter is optional.) The value of 0 is transparent, and 255 (the default) is fully opaque. This parameter is useful for fading out an effect as it completes by using a transparency gradient, by assigning successively decreasing alpha to the final few frames.

Coordinates are given in the DIS and CIGI convention of X forward, Y right, and Z down. Because the billboard is rotated to appear normal to the viewer's direction of gaze and placed at the effect location, the X is implicitly zero and is therefore not needed.

Included with VRSG is a built-in flame effect called flames.flm, shown in the screen capture above. This .flm effect can be used the same way that other billboard (.eff) and particle (.par) effects are used.

Adding particle-based effects

You can use VRSG's particle-based effects (.par) for smoke plumes, contrails, blowing sand, blowing dust, dust trails, rotor wash, exhaust emissions, tactical smoke, and other special effects. Some particle effect files mimic exhaust emissions for specific vehicles.

Particle effects provide a greater degree of realism, as they are volumetric and interact with wind. VRSG includes many solid particle effects, which are described later in this section.

You can also use particle-based effects for one-time animations such as explosions, muzzle flash, or tactical smoke. To do so, you would map the intended particle effect(s) into the file FireMap.ini, DetMap.ini, or ClientMap.ini the same way you would with a billboard-based (.eff) effect file. You can specify several effects for one entry as shown in this example:

```
2 9 225 2 73 5 * * explosion_large.eff dustPuff.par dustBillow.par
```

Tactical smoke is generally evoked via a DetonationPDU. You map the enumeration of the burst descriptor into Effects\DetMap.ini, associating the enumeration with a particular smoke model. Tactical smoke models have a finite system lifetime parameter, as they eventually stop smoking. VRSG automatically removes them when they are done smoking.

The following parameters are in the smoke.par file:

- `max_verts` – controls the maximum number of active particles in a given system. This parameter provides an upper bound on computational load of the system, as well as the perceived density of the system.
- `new_verts_per_second` – controls the rate at which new particles are generated; thus a particle system is fully populated after (max_verts / new_verts_per_second) seconds.
- `new_verts_per_meter` – ties the rate at which new particles are generated to the velocity of a moving entity, ensuring that particles are emitted more often as the entity's speed increases. This is useful for particle effects attached to fast moving objects such as missiles or aircraft, to emit enough particles such that the contrail or smoke trail appears visually connected.
- `particle_lifetime` – specifies the lifetime, given in seconds, of a single particle.
- `particle_lifetime_ir` – specifies the lifetime, given in seconds, of a single particle, when viewed in the IR spectrum.
- `particle_lifetime_nvg` – specifies the lifetime, given in seconds, of a single particle, when viewed in the NVG spectrum.
- `system_lifetime` – amount of time the system will continue to generate particles. After system_lifetime is exceeded, the system will cease generating new particles. Existing particles will continue to propagate until they expire (controlled by particle_lifetime). When all particles have expired and there are no more active particles, the entire particle system is removed.
- `particle_lifetime_variance` – adds a degree of randomness to the lifetime of a particle. The lifetime of a particle will vary randomly between particle_lifetime – 0.5 * particle lifetime_variance and particle_lifetime + 0.5 * particle_lifetime_variance.
- `start_color` and `end_color` specify the initial and final color of each smoke particle, as four component values (red, green, blue, alpha), with each value between 0 and 1.
- `start_color_variance` and `end_color_variance` add a degree of randomness to the color of each particle.
- `start_size` – specifies the starting size of a particle, given in meters.
- `end_size` – specifies the terminal size of a particle, given in meters.
- `size_variance` – introduces variability in the start size and end size of particles. The value given is a percentage of the given start and end sizes. The default size variance is zero percent.
- `ir_scale` – scales particles by a given value when rendered in the IR spectrum.
- `nvg_scale` – scales particles by a given value when rendered in the NVG spectrum.
- `azimuth` and `elevation` parameters control the direction vector in which the particle will be ejected. The variance of these parameters allows you to have something between a fine spray to a full omnidirectional eruption.

- `speed` – controls the velocity of the particle in meters per second. Combining this speed with azimuth and elevation angles defines the velocity vector of a particle.
- `speed_variance` – provides variability in the velocity.
- `acceleration` – adds an optional acceleration vector to the particle. You can use an external force vector, wind, which applies to each particle as well.
- `delay` – specifies a delay, in seconds, before the first particle is emitted.
- `lod_range` – specifies the maximum range the particle system should be displayed. You can use this parameter to limit how far away particle systems can be rendered, which can improve performance.
- `radius` – normally particles are emitted from a single point, at the prescribed origin. If the radius keyword is present, particles are randomly distributed across a circle surrounding the origin. The radius is specified in meters. This feature is useful for creating rotor wash effects.
- `origin` – specifies the X Y Z origin of particle emission. By default, particles are emitted from the origin (X=Y=Z=0).
- `emissive` – causes the particle system to be self-luminous, emitting the specified amount of light independent of time of day.
- `refractive` – causes the effect to refract light. A useful application is a jet engine exhaust effect.
- `sticky` – causes the particles to move with the entity they are attached to. Normally particles are emitted and decay in place at their point of attachment. Their motion comes from their given velocity, direction, and acceleration properties, but are not affected by the movement of the entity that emitted the particle. The sticky keyword causes the particles to move with the entity. An example application is a tracer effect, being attached to a munition round.
- `min_pixel_size` – specifies the minimum size a particle will display as, given in screen pixels. If the effect of perspective projection on the size of the particle caused it to be smaller than this size, the displayed size would be held at this value.
- `wind_scale` – modulates the degree to which the wind vector influences particle motion. To make particles less influenced by wind, use a value less than 1.0.

Use the particlePixelScalePercent DWORD registry variable to scale the minimum pixel size of particles that have a min_pixel_size defined for them. For example, a value of 150 would increase the min_pixel_size by 50 percent. This registry variable is useful for distributing one set of effects files, and then easily tweak them to improve their visibility on a specific display.

Particle systems support separate color parameters for IR and OTW modes. Because IR is monochrome, these parameters are scalar values in the range 0..1. The alpha channel is taken from the OTW colors. For example:

```
start_color_ir 0.6  ; initial IR intensity of particle
end_color_ir 0.4    ; terminal IR intensity of particle
```

By creating copies of and editing the particle description files in the \Effects directory (such as smoke.par, flames-1m.par, dustBillow-small.par, and helo_rotor.par, and so on) you can create or customize new particle-based effects. Examine the commented syntax in these .par files for more information. When creating particle systems for tactical smoke, you can use \Effects\smoke.par as a starting point. Modify the system_lifetime parameter to control how long you want the system to emit smoke. Edit the start_color and end_color fields to achieve the desired color for the smoke. Give your new smoke effect a different filename. Finally, add an entry in DetMap.ini to associate your new effect with a DIS enumeration for a munition. In this way, you can customize per-model effects.

The following examples show a scene that initially (on the left) uses smoke.par, thus they all have the same start time and expansion duration.

A scene using smoke.par. All the instances have the same start time and duration.

Using smoke.par variants with different start times and duration (and a few smokeBlack.par instances).

You can vary the size of a particle effect with the `size_variance` <*percent*> parameter. The variability affects the starting size and the ending size of the particles created.

Use the `boxDims` parameter for a box-shaped particle emitter for particle system. With this parameter you use dimensions in place of a point source or radial-source emitter, in the following format:

```
box_dims <width> <height>
```

The following four parameters control the corkscrewing shape of contrails:

- `corkscrew_angle` <*degrees*> Angle which the particles are emitted off-axis.
- `corkscrew_angle_rate` <*degrees_per_second*> Rate to diminish the effect of the corkscrew_angle. This parameter simulates the effect of the missile refining its track as its flight progresses.
- `corkscrew_spin` <*degrees_per_second*> Rate of rotation of the corkscrew pattern.
- `corkscrew_delay` <*seconds*> Time into the flight at which the corkscrew pattern begins.

Among the particle effects (.par) that are delivered with VRSG are several contrail effect files. You can examine these files, and edit a copy of them in a text editor to create your own contrail effects.

You can preview .par effects in the Model Viewer to see how the effect appears and dissipates, but the Model Viewer does not give any performance related information.

If your site is developing its own custom effects, MVRsimulation *strongly recommends* you preview these effects directly in VRSG, by dragging the .par file from Windows File Explorer and dropping it on the rendered VRSG scene. Doing this kind of inspection will help validate the effect's frames-per-second performance, scale, and particle_lifetime (duration).

Adding solid particle-based effects

VRSG's ballistic solid-particle effects (.spf), located in VRSG's \Effects directory, model projectiles with dust trails that are cast from detonation events. Some throw solid particles in a given direction denoted in the filename (debrisEast.spf, debrisNorth.spf, and so on).

Solid particle effects are "fire-and-forget"; they are not addressable in flight, which means you cannot change their behavior dynamically. They expire and remove themselves automatically when they are complete. The solid particles are not terrain-collision aware; after the 3D debris models are hurled, they end up below the terrain and are no longer visible. You can optionally attach a particle-based contrail effect to each 3D debris object.

Like other effects, they can be listed in the \Effects\DetMap.ini file and can be detonated from a Detonation PDU and layered with other effects. They are listed in the \Effects\ClientMap.ini file that is delivered with VRSG, so you can trigger them from an event in a scenario created in Scenario Editor, or drag-drop the effects file directly onto the terrain in the VRSG visualization window.

Also like the other effects, an .spf file can be copied and edited in text editor, which means you can customize the parameter values or create your own solid particle effect. By copying and/or editing the solid particle description files in the \Effects directory (debrisLarge.spf, debrisCrater.spf, and so on) you can create or customize new particle-based effects. The commented syntax in these .spf files provide more information. Dragging and dropping the effects in the VRSG visualization is useful for previewing your results.

The following parameters are in the debrisCrater.spf file:

- `filename =` debris-small.hpy – model name of the "solid particle" object that will be cast by this particle effect.
- `maxParticles =` 150 – the maximum number of solid particle objects emitted by an instance of this effect during its lifetime.
- `particlesPerSecond =` 100 – the maximum number of objects emitted per second.
- `emitterLifetime =` 0.4 – the duration, in seconds, that an instance of this effect will emit new solid particle objects (instances of the model from above).
- `lodRange =` 2000 – the distance from the eyepoint the particle system will be visible/rendered.
- `particleLifetime =` 4 – the duration, in seconds, each individual solid particle object will be updated before it is removed from the queue of active particles.
- `particleLifetimeVariance =` 1 – adds a degree of randomness to the lifetime of each solid particle object (value in seconds).
- `velocityDir =` 0.0, 0.0, 1.0 – the direction in which particles will be emitted, in North-East-Up (NEU) coordinates, with +Z being upward. In this example, the particle system is shooting particles straight upward.
- `velocityVariance =` 50.0, 50.0, 50.0 – adds random variation to the direction in which particles will be emitted. The values specify an allowable deviation (in degrees) in heading, pitch, and roll of each individual particle's initial velocity vector.
- `velocityMag =` 20.0 – defines the velocity of each emitted particle, in meters-per-second.
- `velocityMagVariance =` 10 – applies a degree of randomness to the velocity of each particle.
- `rotationRate =` 360.0 – controls the speed at which each particle rotates about its axis (in degrees-per-second).
- `rotationRateVariance =` 60.0 – applies randomness to the rotation rate (in degrees-per-second).
- `rotationAxis =` 1.0, 0.0, 0.0 - defines the axis of rotation around which each particle will rotate (in NED coordinates).
- `rotationAxisVariance =` 20.0, 20.0, 20.0 – defines random rotation (heading, pitch, roll) to offset the above axis of rotation to add a degree of randomness to individual particle rotations.
- `scale =` 0.8 – applies scale to the model representing each particle.
- `scaleVariance =` 0.5 – applies randomness to the above scale value.
- `position =` 0.0, 0.0, -1.0 – the initial starting position of each particle, in NEU coordinates. This system is starting the particles 1 meter down.
- `positionVariance =` 1.0, 1.0, 0.0 – applies randomness to the particle starting position.

- `orientation =` 0.0, 0.0, 0.0 – the initial starting orientation of particles (heading, pitch, roll) in degrees.
- `orientationVariance =` 130.0, 130.0, 130.0 – applies randomness to each particles' starting orientation, in degrees.
- `particleEffect =` debrisTrail.par – the particle effect that will trail each solid particle as its being simulated.

Performance considerations

MVRsimulation's particle effects system has multi-dimensional performance implications. Particles are powerful with an open architecture; they can easily be configured such that unknowingly you could create effects that adversely affect performance within a VRSG scene. Your site could create an effect with millions of particles which runs fine in VRSG, as long as the footprint on the screen is small, that is, far from the eyepoint. (The more pixels covered by the effect in the scene, the more time it takes to render the scene.) Conversely, you could create an effect that has a modest number of particles but adversely impacts the frame rate when it is rendered very close to the eyepoint, especially in a scene using other rendering-intensive features such as 3D oceans and shadows.

MVRsimulation *strongly recommends* you preview your effects directly in VRSG prior to fielding them. To preview an effect, drag the effect (.par, .eff, or .spf file) from the Windows File Explorer and drop it onto the rendered scene in the VRSG visualization window. Inspect the effect up close and far away in Desktop Cover mode and press the H key to see the frame rate. This way, you can see the frames-per-second performance of the effect, the particle generation rate, and the particle_lifetime duration of the effect. (For example, MVRsimulation's support team has seen problematic customer effects that were 8km wide and 16km high, accidentally configured for a 17-year window with a particle generation rate of 500 particles per second.)

You can also preview effects in the Model Viewer, but doing so will help only with visual verification, not with performance-related characteristics.

Adding dynamic craters

VRSG can dynamically deform terrain surfaces to represent craters resulting from munitions impact. You can associate crater radius and depths with differing munition types. Upon receipt of a detonation event, VRSG dynamically hyper-tessellates the terrain surface to the degree needed to capture the crater's shape. The newly formed crater supports mission functions such as elevation lookups and intervisibility queries. In physics-based IR rendering, the crater appears hot, with incremental cooling over time, from the outer radius inward.

In the \Effects\DetMap.ini or \Effects\ClientMap.ini file, at the end of the entry for the effect file, add:

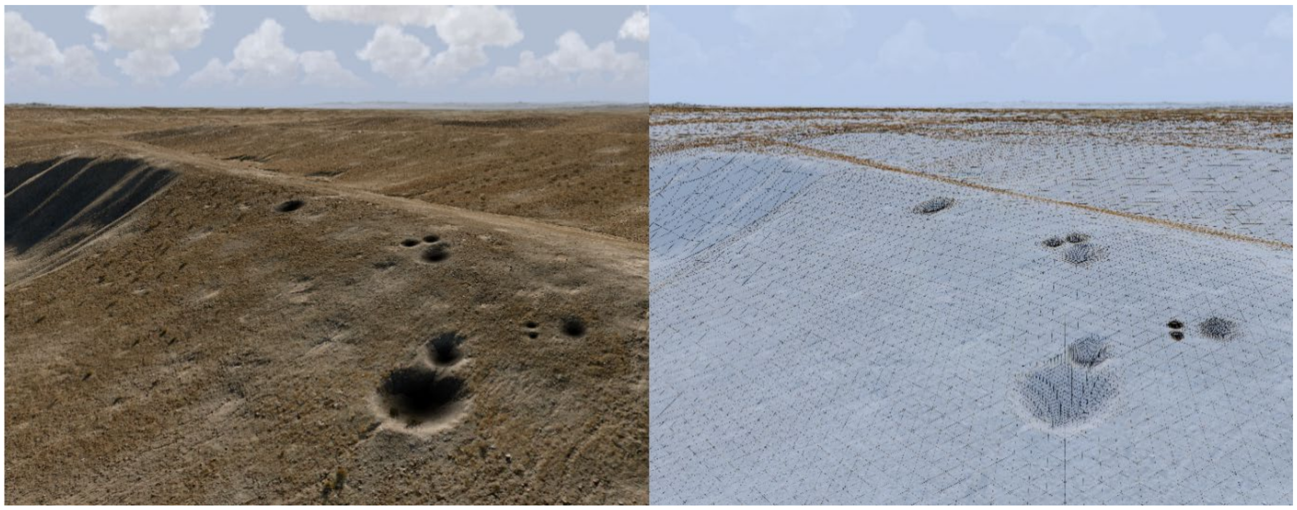
```
-craterRadius=N
-craterDepth=M
```

Where the radius and depth are in meters. CraterDepth is optional. If a depth value is not provided, VRSG will use the radius for the crater depth.

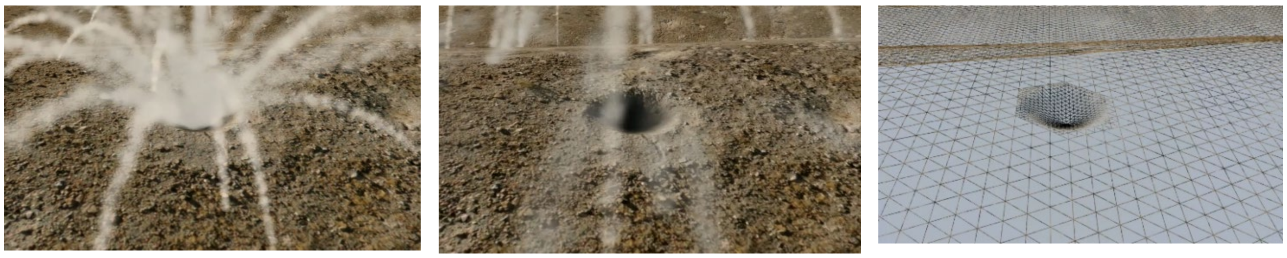
For example:

```
explosion_large.eff -craterRadius=10 -craterDepth=4
```

The following example shows textured and wireframe views of a set of craters. The geospecific round-earth terrain of the Prospect Square area at the U.S. Army Yuma Proving Ground was built with 2 cm imagery collected by MVRsimulation’s small data collection UAV.



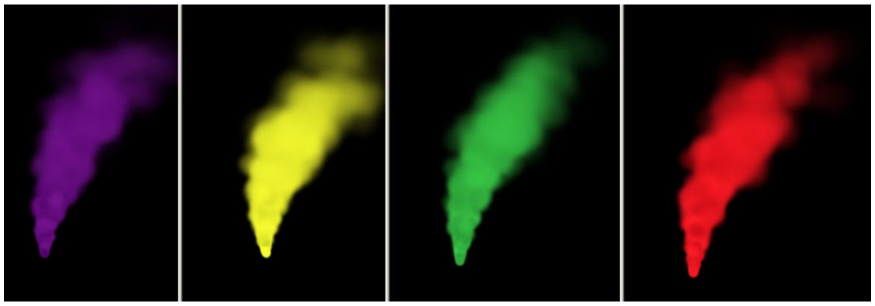
Real-time VRSG screen capture of textured and wireframe views of craters rendered on 2 cm per-pixel resolution virtual terrain of the Prospect Square area at the U.S. Army Yuma Proving Ground, AZ. The terrain was built with 2 cm imagery collected by MVRsimulation’s data collection small UAV.



Real-time VRSG screen capture of textured and wireframe views of craters and a crater particle effect triggered by a detonation event rendered on 2 cm per-pixel resolution virtual terrain of the Prospect Square area at the U.S. Army Yuma Proving Ground, AZ. The terrain was built with 2 cm imagery collected by MVRsimulation’s data collection small UAV.

Adding signal grenade smoke

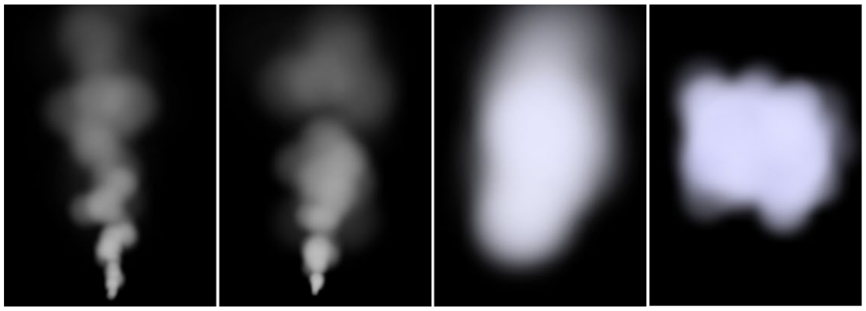
VRSG contains a set of colored smoke effects for use as a signal grenade.



These smoke flares are located in the \VRSG\Effects directory:

smokeGreen.par	smokeWhitePhosphorus.par
smokeRed.par	smokeWhitePhosphorus-mushroom.par
smokeViolet.par	smokeWhitePhosphorus-mushroom-small.par
smokeWhite-0m.par	smokeWhitePhosphorus-small.par
smokeWhite-0mTall.par	smokeYellow.par

The following examples show the white and white-phosphorus effects:



These effects can be used with smoke grenade models in the MVRsimulation military model library: M18.US.green.hpy, M18.US.red.hpy, M18.US.violet.hpy, M18.US.yellow.hpy, M67.US.green.hpy, and M83.US.white.hpy mapped to detonations in DetMap.ini. As well, each smoke grenade model has an analogous "weapon-" model, which can be used with the throwing grenade animation for human character models. (See the chapter "Using 3D Characters in VRSG" for information about character entities and animations.)

The following image shows a character model (human-us-soldier-sof-030.hpy) throwing a green smoke grenade with the throwing-grenade.bvh animation.

Physics-based destruction of buildings

VRSG supports Applied Research Associates' (ARA's) damage server, which computes physics-based destruction of buildings in near real-time. The damage server can replace a building in VRSG by sending a damaged version that takes into account weapon type, point of impact, incidence of impact, and building physical properties. The new damaged version of the building is transmitted to VRSG using a secure HTTPS connection. Building models used with ARA's damage server must be pre-constructed in some conformal way to be used by their server.

This damage server is a component of Integrated Weapons of Mass Destruction Toolset (IWMDT) Simulation, and property of the US government. Distribution is authorized to US government agencies and their contractors only; critical technology and administrative or operational use. Contact David Pyle at ARA at dpyle@ara.com for more information.

Adding and editing model metadata

VRSG offers the ability to add attributes, or *metadata*, to 3D models. Model metadata is stored in the popular human-readable/editable JSON file format. You can edit JSON files directly with an ASCII text editor such as Notepad.

Overview

For any given model in MVRsimulation's model format (HPY or HPX), VRSG checks for the existence of a file of the same model name, in the same directory, with the ".json" extension. If such a file exists, VRSG will load and parse the JSON file to extract the metadata for that model. For example, to modify the metadata for the model AC-130H.US.grey.hpy, you would create and edit a file named AC-130H.US.grey.json, located in the same directory as the model itself. Many of MVRsimulation's newer military models already have a JSON metadata file embedded within the HPY model format, which you can extract and edit. You can check for the presence of an embedded JSON file and extract it by loading the model of interest in Model Viewer and choosing File > Export Model Metadata. This action exports the JSON metadata file in the same directory as the model. If that menu item is unavailable, no metadata file exists within the model, and you will need to create one from scratch.

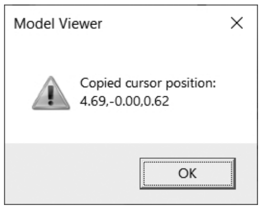
You can modify the following attributes of a model in the model metadata JSON file:

- How the model is ground-clamped: to include a model's origin above-ground altitude, and resting pitch angle.
- How animated movements of multiple articulated parts of a model are coordinated.
- How DIS burst descriptors in FirePDUs select which muzzle flash animations to play.
- How muzzle blast particle effects are attached to a model and associated with DIS burst descriptors in FirePDUs.
- How a light point is attached to a model and its color, intensity, and angle attributes.
- How a static, persistent light lobe is attached to a model and its color, intensity, and angle attributes.
- For models that need to be cut into the terrain (e.g. fighting positions), specify the width and height of the rectangle the model requires.
- Assign a weapon type to a weapon model.
- Control how a model is scaled as a function of range, overriding the default behavior of perspective projection.
- Remap a model's articulated part codes.
    - The expected value is an array of integers, consisting of the current part ID followed by the new part ID.
    - Example JSON file entry would look like this:
```
"artPartRemaps": [
   7715, 37,
   7716, 38
],
```

MVRsimulation will create more attributes for model metadata JSON files in future VRSG releases.

For a given model displayed in the Model Viewer, the display shows (on the lower left part of the display) the x, y, z values of the cursor's position on the model; these values are needed for model attributes that require attachment points. The x, y, z values identify a position in model-space in meters, using the DIS and CIGI convention of x= forward, y= right, z= down.

To copy the x, y, z offset position of a given model in the Model Viewer, place the cursor over the location of interest on the model and press the letter "P" key on the keyboard. This action copies the x, y, z coordinates of the cursor position to the Clipboard.


Model Viewer

Copied cursor position:
4.69,-0.00,0.62

Paste the x, y, and z values into the syntax in use in the JSON file.

For a model's selected articulated part, the part's relative position under the cursor is also displayed in the lower left corner of the Model Viewer. This can provide the number needed for specifying where particle effects are attached to the given model.

For more information about the Model Viewer, see the chapter "Previewing Models, Effects and Terrain."

Syntax example

A JSON file is a collection of key/value pairs. A key defines the attribute by name and is always in double-quotation marks. Following the key is its corresponding value. The value can be a number, a quoted string, an array of discrete values, or an array of compound objects. You can find full descriptions of the JSON syntax on many websites. For the purpose of understanding what model metadata you can control, consider the set of possible key/value items in the following example JSON file:

```
{
     "version": 1,
     "groundPlaneZ": 2.4,
     "muzzleTip": [
          {
            "enum": "2:1:225:1:1:0:0",
            "muzzleFlashIndex": 0
            "partId":  6048,
            "offsetX": 0.0,
            "offsetY": -1.0,
            "offsetZ": 0.0,
            "azimuth":   -90.0,
            "elevation": -10.0,
          },
          {
            "enum": "2:1:225:1:2:0:0",
            "muzzleFlashIndex": 1
            "partId":  6080,
            "offsetX": -0.2,
            "offsetY": -1.5,
            "offsetZ": 0.0,
            "azimuth": -90.0,
            "elevation": -10.0,
          }
     ]
}
```

Ground clamping

The `groundPlaneZ` key/value pair specifies the Z (vertical) position in the model's coordinate system that should rest at ground level when the model is being ground clamped. By default, a model is ground-clamped at its Z coordinate value of zero. For some models, such as aircraft, this default is not desirable. Aircraft models typically have their Z origin at the center of mass, but you would want the model to ground clamp to the bottom of the landing gear.

Resting pitch

The `restingPitch` key/value pair specifies a pitch angle to apply to a model when in a ground clamped state. This is useful for models that do not sit flat when resting on the ground, such as many helicopters.

Terrain cutouts

Models that have the `terrainCutout` attribute will cause VRSG to cut a rectangular-shaped hole into the terrain to accommodate the model. This is useful for models such as fighting positions and tank defilades. The value is a compound structure consisting of width and height attributes, as illustrated by the following example:

```
"terrainCutout":
  {
       "width":  7.0,
       "height": 19.6
  }
```

The VRSG model library contains several pre-built fighting position models which are stored under Models\Other\fighting-position-*.hpy.

Weapon type

The `weaponType` field indicates to the animation system how the weapon model is placed into the hands of a human character. When a human holds a weapon model, the weapon type is used to select the appropriate animation given the human's posture and weapon state attributes. Possible values for weapon type include "rifle", "rpg", "pistol", etc. To support a new weapon type, BVH animation files for the new weapon type must be created and added to the Animations folder.

Scale function

The `scaleFunction` field is a compound structure that controls how a model is scaled as a function of range from the viewer. Without this entry, a model is scaled consistent with the perspective fore-shortening of camera projection. Models can be made to be scaled artificially larger as a function of range to enhance identification/classification and account for the reduced visual acuity associated with computer-generated graphics. This feature is best illustrated by an example:

```
"scaleFunction":
  {
     "startScale":1.0,
     "endScale":  2.0,
     "startRange": 1000.0,
     "endRange": 10000.0
  }
```

In the above example, the model's scale will increase between 1X and 2X as the distance to the model increases between 1 kilometer and 10 kilometers. For distances beyond 10 kilometers, the scale will remain at 2X, and for distances below 1 kilometer, the distance will be 1X.

Articulated parts animation

In a JSON file you can modify or define the articulated part animation data. In this way you can control the start time, position, and duration of multiple articulated parts in a coordinated manner.

The articulated part animation data provides two purposes:

- It simplifies part movement for a controlling simulation. A simple simulation can just toggle a switch state to move a part from position A to position B, or from position B to position A. This relieves the simulation of having to provide explicit part position data over time.
- It enables the coordinated movement of multiple articulated parts. For example, an aircraft model may have a complex object to move, such as the landing gear, which requires the coordinated movement of multiple parts. The gear may be modeled with multiple parts, such as the door, wheel brace angle, hydraulic arms, and so on. The simulation could simply control the door angle, and the position of everything else would derive from the door angle. Simple simulations can just toggle the switch state, but more advanced simulations might need to take explicit control over, to represent a condition such as the gear getting stuck half-way down.

A simulation could use the simple switch state interface to animate this coordinated group of multiple parts. The simulation can also take explicit control of a single part by sending articulated part positions; in this case the positions of the other parts in the group are implied by the position of the single part that the simulation is controlling.

An articulated part animation is identified by the `partAnimations` variable name. The value section is an array of structured elements. Each of these elements contains the following members:

- appearance - this optional field provides the switch state bit used to play the animation. The value is expected in hexadecimal, with the "0x" prefix expected. If this field is not present, the simulation must provide articulated part data for one of the parts in the group, the other part positions will be implied by the provided part positions for the other part.
- duration - the length in seconds of the animation.
- name - provides a label which is used in the Model Viewer’s Switches menu and serves as an identifier for the part being controlled.
- table - this is an array of arrays. One array is provided for each part in the animation group. At minimum one array must be provided for one part. If additional parts are included in the group, they will be additional elements of the array. The elements of the array are as follows:

```
[ partId, time0, position0, ... timeN, positionN ]
```

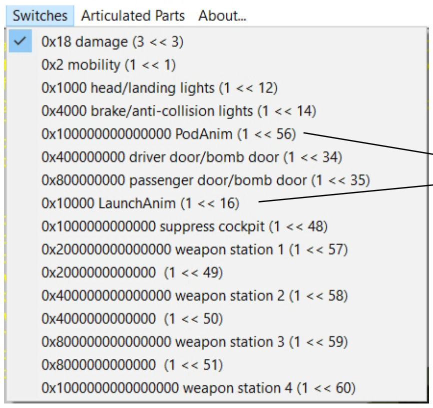
Where:

`partid` is the DIS articulated part code plus the degree of freedom identifier. This is the same value that would be used in a DIS Entity State PDU articulated part record.

Following the `partId`, is a set of time/position pairs, indicating the position of the part at the given time into the animation sequence. For part translations, these would be values given in meters. For angular movements, these would be given in radians.

In the following example, all the models of this Sky Dragon DK-10 air defense system have one or more animated articulated parts and an embedded JSON metadata file.

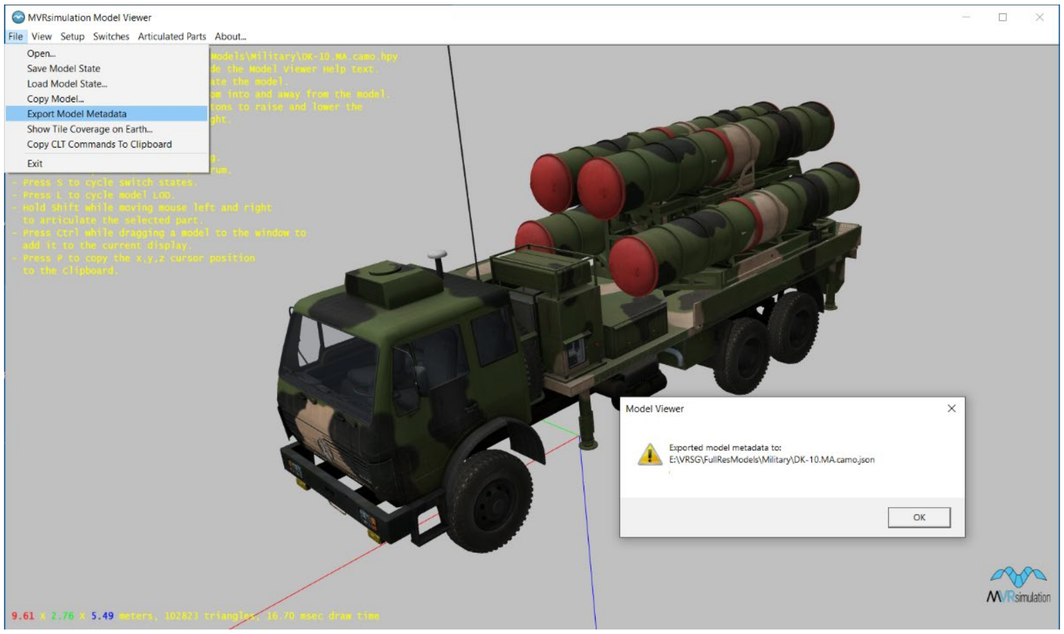
Loading the leftmost model, DK-10.MA.camo, the scene in the Model Viewer, you can see that it has two animations, one for the launcher and another for the articulating ground support pods.



✓ 0x18 damage (3 << 3)
0x2 mobility (1 << 1)
0x1000 head/landing lights (1 << 12)
0x4000 brake/anti-collision lights (1 << 14)
0x100000000000000 PodAnim (1 << 56)
0x400000000 driver door/bomb door (1 << 34)
0x800000000 passenger door/bomb door (1 << 35)
0x10000 LaunchAnim (1 << 16)
0x1000000000000 suppress cockpit (1 << 48)
0x200000000000000 weapon station 1 (1 << 57)
0x2000000000000 (1 << 49)
0x400000000000000 weapon station 2 (1 << 58)
0x4000000000000 (1 << 50)
0x800000000000000 weapon station 3 (1 << 59)
0x8000000000000 (1 << 51)
0x1000000000000000 weapon station 4 (1 << 60)

The two animations of articulated parts within the DK-10.MA.camo model.

To edit these animations, first extract the model metadata from the HPY model file by choosing File > Export Model Metadata. This action extracts a JSON metadata file located outside the model and in the same director which the model is located. The JSON metadata file has the same name as the model and a .json extension.



```
Models\Military\DK-10.MA.camo.hpy
de the Model Viewer Help text.
ate the model.
om into and away from the model.
tons to raise and lower the
ght.
g.
rum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the Clipboard.
```

Exported model metadata to:
E:\VRSG\FullResModels\Military\DK-10.MA.camo.json

```
9.61 X 2.76 X 5.49 meters, 102823 triangles, 16.70 msec draw time
```

The following is an excerpt from the JSON file for the DK-10.MA model, which describes the two animations that are delivered with this model. (You could add more animations to the model by editing this file.)

```
    "partAnimations": [
        {
            "appearance": "0x100000000000000",
            "duration": 2.5,
            "name": "PodAnim",
            "table": [
                [
                    5705,
                    0.0,
                    -0.0,
                    2.5,
                    0.5966072082519531
                ],
                [
                    5741,
                    0.0,
                    0.0,
                    1.6666666269302368,
                    2.96705961227417
                ],
                [
                    5775,
                    0.0,
                    0.0,
                    1.25,
                    2.96705961227417
                ],
                [
                    5807,
                    0.0,
                    0.0,
                    1.25,
                    -2.96705961227417
                ]
            ]
        },
        {
            "appearance": "0x10000",
            "duration": 5.0,
            "name": "LaunchAnim",
            "table": [
                [
                    5837,
                    0.0,
                    0.0,
                    5.0,
                    0.7853981852531433
                ],
                [
                    5869,
                    2.5,
                    0.0,
                    5.0,
                    0.7853981852531433
                ],
                [
                    5895,
                    2.5,
                    0.0,
                    5.0,
                    -0.30000001192092896
                ],
                ]
            ]
        }
    ],
    "version": 1
}
```

Muzzle flash animations

Many models have muzzle flash animations that are played in response to an entity issuing a DIS FirePDU. If a model has a single muzzle flash animation, and there is no JSON metadata file provided for the model, any FirePDU issued by the entity will trigger the playback of the muzzle flash animation. If a model has multiple muzzle flash animations, they will each be assigned an index, starting at zero and ending at the number of animations minus 1. If there is no JSON metadata file provided for the model, then a FirePDU will play the first muzzle flash animation in the model (index 0).

By adding JSON metadata for the model, you can control how FirePDUs select which muzzle flash animation to play. A JSON file can contain a `muzzleTip` object, which is defined by an array of compound objects in the JSON syntax. Each array object has two members, associating the DIS enumeration from the FirePDU's burst descriptor with the index of a muzzle flash animation. The `enum` member contains the DIS enumeration, and the `muzzleFlashIndex` field contains the index of the muzzle flash animation to play for that enumeration.

For models with built-in flip-book animations for muzzle flash effects, only the `enum` and `muzzleFlashIndex` members are needed. These two members are sufficient to allow VRSG to choose the proper built-in muzzle flash animation from the FirePDU burst descriptor.

A muzzle flash animation can also elicit the creation of particle-based effects. By adding additional fields to the `muzzleTip` object, you can control how particle effects are attached to guns or other articulated parts in response to FirePDUs. When you add a particle effect to FireMap.ini and use the `-muzzleTip` option, VRSG will attach the effect to the main gun, given by DIS articulated part code 4416. VRSG also estimates the end of the gun barrel by analyzing the model's geometry. With additional metadata, you can control which FirePDUs are associated with which articulated parts (such as secondary guns or grenade launchers), and control how and where the effects are attached. The following fields allow you to customize this behavior:

```
  "partId":  6080,
```

The `partId` field indicates which articulated part to attach to. The default is 4416, for the primary gun.

```
  "offsetX": -0.2,
  "offsetY": -1.5,
  "offsetZ": 0.0,
```

The `offset` triplet specifies the origin of the effect relative to the part's coordinate system. By default, VRSG will place the effect at the X extent of the part, which typically corresponds to the end of the gun barrel. Note these values are in meters and follow the DIS local coordinate system conventions of X forward, Y right, Z down.

```
  "azimuth": -90.0,
  "elevation": -10.0,
```

The `azimuth` and `elevation` fields specify the direction of the particle emission. By default, the particles will emit along the X (forward) axis of the part. In some cases you might need to override this default direction, such as a rear-facing gun in the bed of a technical pickup truck, or a side-facing gun in an AC-130 gunship.

The following complete example is taken from the M777 artillery model.

```
  "muzzleTip": [
    {
      "label" :   "cannon",
      "partId":    4032,
      "offsetX":   4.9,
      "offsetY":   0.0,
      "offsetZ":   0.0,
      "azimuth":   0.0,
      "elevation": 0.0,
      "recoilDistance" : 0.5,
      "effectCommand": "muzzleBlast.par muzzleBlast_right.par
muzzleBlast_left.par",
      "muzzleFlashIndex": 0
    }
  ]
```

Light points

In a JSON file you can define light points for a model, such as an aircraft taillight. A light point is a light source that emits rays of light in all directions from a specific location, but does not cast illumination onto other surfaces. In the model’s JSON file you define the light point’s location, size, and color and intensity of the light.

The following JSON file is an example of the syntax for defining a light point taillight for MVRsimulation’s aircraft model f-16c.us.grey.457fs-85-1457.hpy:

```
{

     "version": 1,
     "groundPlaneZ": 1.82,

    "lightPoint": [
          {
            "X": -5.6,
            "Y": 0.0,
            "Z": -3.4,
            "size": 0.1,
            "minPixelSize": 6.0,
            "maxRange": 5000.0,
            "red": 0,
            "green": 255,
            "blue": 0,
            "intensity": 255,
            "mask": 0,
            "period": 1.0,
            "dutyCycle": 1.0,
            "azimuth": 0.0,
            "elevation": 0.0,
            "horzLobeAngle": 180.0,
            "vertLobeAngle": 180.0
          }
  ]
}
```

The attributes for a light point are:

- `X, Y, Z` identify the light’s position in model-space in meters, using the DIS and CIGI convention of `X` forward, `Y` right, `Z` down.

- `size` – The radius of the light in meters.
- `minPixelSize` – The smallest size the light will display as, given in pixels.
- `red, green, blue` – The color values of the light.
- `intensity` – The intensity, or alpha value, of the color. A value of 00 would be a very dim light, and a value of 255 would be the color at full intensity.
- `Mask` – Which bit of the EntityStatePDU appearance field is used to turn on the light. If zero, the light is always on.
- `period` and `dutyCycle` – Frequency and duty cycle of flashing lights. For example, for a light to flash at 2 Hz, use a period of 0.5. A `dutyCycle` of 0.25 would have that light be on for half of its flash cycle. For non-flashing lights, use equal values for `period` and `dutyCycle`.
- `azimuth` and `elevation` – Direction of directional lights, given in degrees relative to the entity they are attached to.
- `horzLobeAngle` and `vertLobeAngle` – Lobe size of a directional light.

Non-flashing omnidirectional lights only require the attributes for position, sizes, and color.

Light lobes

In a JSON file you can define static, persistent light lobe attributes for static models, such as streetlights or for entities. A light lobe illuminates any 3D surface that is inside its cone of influence. The model's JSON file is where you define the location, orientation, and color of the light lobe.

The following example shows the JSON syntax for adding a light lobe to the metadata file of a streetlight model, in this case MVRsimulation's model streetlight-003.hpy:

```
{
  "version": 1,
  "lightLobe": [
    {
      "offsetX":   2.25,
      "offsetY":   0.0,
      "offsetZ":   -9.42,
      "azimuth":   10.0,
      "elevation": -90.0,
      "halfAngle": 60,
      "fadeRange": 150,
      "red":       255,
      "green":     255,
      "blue":      255
    }
  ],
}
```

The attributes for a light lobe are:

- `offsetX, offsetY, offsetZ` identify the light's position in model-space in meters, using the DIS and CIGI convention of `X` forward, `Y` right, `Z` down.

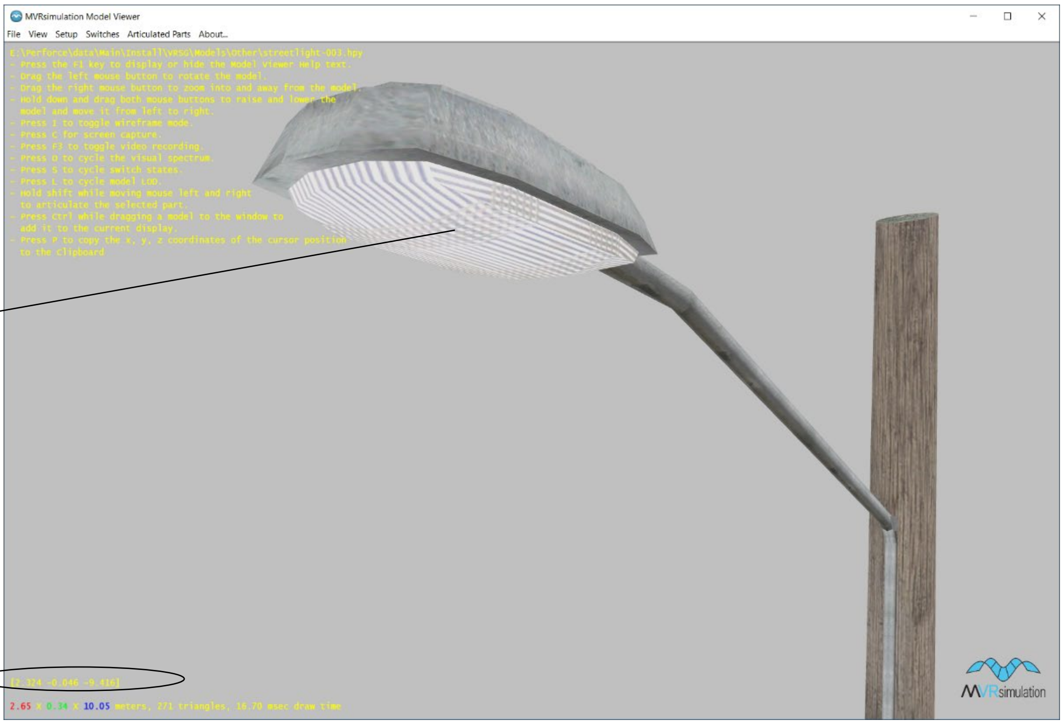
- `azimuth` – Rotate right/left from the XYZ origin. (Use 0 for straight down)
- `elevation` – Pitch up/down from the XYZ origin. (Straight down is -90.)
- `halfAngle` – Size of light pool cast on the ground. A good sized light pool is 100; a value of 10 yields a smaller light pool.
- `fadeRange` – Brightness of the light cast on the ground. (100+ is a bright light; a value of 10 is a soft light.)
- `red, green, blue` – The color values of the light each in the range 0-255.

To increase the brightness of the light lobe, for example for stadium lighting, increase the values of the `halfAngle` and `fadeRange` attributes.

To easily obtain the x, y, and z offset position and other attributes needed for adding a light lobe to a model:

1. In Model Viewer, place the cursor over the exact location on the model you want the light lobe to be.
2. Press Shift-P on the keyboard.

   This action copies to the Windows Clipboard not only the x, y, z offset values of the cursor position, but also the orientation and lobe color attributes.

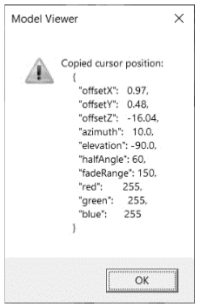

MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

```
E:\Perforce\data\Main\Install\VRSG\Models\Other\streetlight-003.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press 1 to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x, y, z coordinates of the cursor position
  to the Clipboard

[2.324 -0.046 -9.416]
2.65 x 0.34 x 10.05 meters, 271 triangles, 16.70 msec draw time
```

The location for attaching a light lobe to this streetlight model.

The x,y,z offset position over which the cursor is resting. Use these values to specify the attachment point in the JSON file.

3. Paste the captured attributes from the Clipboard into the model’s JSON file.



```
Copied cursor position:
 {
  "offsetX":   0.97,
  "offsetY":   0.48,
  "offsetZ":   -16.04,
  "azimuth":   10.0,
  "elevation": -90.0,
  "halfAngle": 60,
  "fadeRange": 150,
  "red":       255,
  "green":     255,
  "blue":      255
 }
```

4. In the JSON file, be sure to append the heading information “"version": 1” and “"lightLobe":” to the beginning of the light lobe syntax as shown in the example above.

To enable the light lobes to illuminate in VRSG, set the Environment tab settings on the Dashboard to a dark sky or night time.

Note: All streetlight and stadium light models in MVRsimulation’s culture model library have built-in light lobe attributes compiled into the HPY model.

Adding cultural features to the terrain

Cultural features include any static model such as a tree, a building, a street sign, or a non-moving character or vehicle. Cultural features for a given terrain are described in a cultural feature file (vrsg.clt), an ASCII text file that specifies models and their locations on the database. VRSG reads the vrsg.clt at runtime and loads the models described in the file.

To add culture to a given 3D terrain, use one or more of the following methods:

- Build up substantial content on the terrain with the VRSG Scenario Editor application, which is installed with VRSG. This application provides a graphical interface with tools you can use to build dense 3D scenes on your 3D terrain with models from MVRsimulation’s content libraries. In addition to adding realistic visual content you can create pattern-of-life scenarios with Scenario Editor. For more information, see the *MVRsimulation Scenario Editor User’s Guide*.
- Place one or two cultural features directly on the terrain by dragging a given 3D model file (in MVRsimulation’s HPY or HPX model format) from Windows Explorer and dropping it in the VRSG visualization window at a specific location on the terrain. To refine the placement of the model, attach to the model and then move it or reorient it. Save the edit to a cultural feature file by pressing the J key on the keyboard. This process is described further in this chapter.
- Place one or more models on the terrain by editing a cultural feature file in an ASCII text editor to add an entry for the model with its coordinates, orientation, and directory path, as described in this chapter. Models that can be placed this way include large inset models of multiple buildings generated by CityEngine and converted to MVRsimulation’s model format (HPX).

The best way to build up a lot of culture on 3D terrain is with VRSG Scenario Editor. While this application’s main purpose is to provide a means for creating scenarios to run in VRSG, it provides an easy way to place and manipulate culture quickly on terrain tiles, and to produce a cultural feature file. For more information about VRSG Scenario Editor, see the *MVRsimulation Scenario Editor User Guide*.

Although you can add a culture model to the terrain while it is rendered in VRSG, if you need to add a substantial number of static models, it is more efficient and substantially less time-consuming to do so in VRSG Scenario Editor. Moreover, MVRsimulation *strongly* advises against using these steps to make edits to culture on a terrain that you have already edited in Scenario Editor and exported a scenario to run in VRSG. It is best to use the method below to modify a few culture models that are not associated with scenarios created in Scenario Editor.

A cultural feature file is always named "vrsg.clt" as in \MVRsimulation\VRSG\Models\vrsg.clt. Although VRSG always writes updates to the file \MVRsimulation\VRSG\Models\vrsg.clt, you can have multiple vrsg.clt files located in different subdirectories, for example stored with terrain tiles organized by geographic area.

During visualization, VRSG loads all vrsg.clt files it finds in its search path (that is, in the directories listed in the Alternate Folders for Terrain, Models, and Textures section of the Advanced Parameters dialog box) as well as the one in the \MVRsimulation\VRSG\Models directory. In effect, the content is appended as if it were all in a single vrsg.clt file. If necessary, you can examine the VrsgInfo.txt to see which vrsg.clt files were loaded for the VRSG session.

Using cultural feature files can increase the flexibility of your 3D terrain. For example, if your terrain is used by one set of users that require a forest in a specific area while another set of users wants the area populated with some buildings for a different scenario, instead of compiling the trees or the buildings into the terrain, you could reference them in separate cultural feature files and provide each set of users different cultural feature files. An example of having different types of culture in separate cultural feature files can be found in the Kismayo, Somalia, 3D terrain delivered with VRSG. The cultural feature files are located in \MVRsimulation\VRSG\Terrain\Somalia\Kismayo\CLT.

Parts of a cultural feature file

The cultural feature file (vrsg.clt) directs VRSG where to place static models on the 3D terrain.

The file begins with a coordinate system header at the top of the file, such as:

```
! LL Coordinates
```

or

```
! UTM Meters in zone <zone>
```

Each model is specified by a one-line entry in the file listing its position, angle, and filename, followed by optional commands (described in the table "Summary of available CLT commands" later in this section). A model entry has the following syntax:

```
X Y Z   Yaw Pitch Roll    Model-Filename
```

where X Y Z specifies the location of the model in geodetic or UTM coordinates in meters.

The elevation may be specified in meters above sea level (MSL), or above ground level (AGL). Use the control comment `!agl=on` to instruct VRSG to interpret elevations as AGL, or ! `agl=off` to interpret elevations as absolute values relative to sea level. For example, the following control comment will cause VRSG to interpret the elevation of all subsequent models as AGL:

```
! agl=on
```

If you are working with models that are not situated on hilly terrain properly, you can explicitly clamp them to the terrain as shown in this vrsg.clt entry:

```
! LL Coordinates
! agl=on
N32 08 29.5596    W111 10 30.6152     0.0     45.00  1.00  2.00
afghan-building-140.hpy
! agl=off
```

Notice `! agl=on` and the elevation value of "0.0".

"Yaw Pitch Roll" specifies the orientation of the model. These values are given in degrees (degrees, minutes, and seconds). The degrees portion can be of arbitrary precision; it is scanned as a double and you can leave minutes and seconds zero.

"Model-Filename" is the name of the HPY or HPX model file to use for the static entity.

Exclamation marks (!) placed at the beginning of a line begin a commented line.

Consider the following vrsg.clt example:

```
!
! LL coordinates
! clamp_lp=on
N35 16 25.59 W116 37 21.09 -1.0 320.5 0.0 0.0 lt_ap_bikelake.hpy 0 1
!
N35 16 25.59 W116 37 21.09 -1.0 320.5 0.0 0.0 lt_rw_bikelake.hpy 0 1
!
N35 16 34.28 W116 37 24.96 -1.0 320.5 0.0 0.0 lt_right_vasibar.hpy 0
1
!
! clamp_lp=off
```

The drawing order for models listed in a vrsg.clt file is the order in which they are listed.

Assigning model IDs

By default, VRSG does not assign IDs to individual static models listed in a cultural feature file (either static cultural features or static vehicles and characters). However, they can optionally be assigned unique IDs much like DIS entities are given IDs. You can explicitly set the ID of a particular model in the vrsg.clt file to make that model addressable.

Some reasons you might need a static model to be addressable are:

- You want to know the ID of a particular model so that you can attach to it (by using the Attach tab in the VRSG Dashboard).
- The model is a light point model and your simulation needs to send a message to VRSG to control the model’s displayed intensity.
- You need a model to be addressable by your CIGI host, or by DIS, or some external system.
- You want to send a SetDataPDU message to VRSG to change a model’s appearance mask (for example, switch a model from an undamaged state to a damaged state).

To assign an ID to a static model, insert the control comment `! id=N` into the vrsg.clt file. For example, the following control comment assigns the ID of 127 to the model that follows it:

```
! id=127
```

A static model that has been assigned an ID in the vrsg.clt file is stored in VRSG's internal database using the DIS convention of a site of zero, host of zero, and the assigned unique entity ID number.

The only case in which VRSG assigns IDs to static models automatically is for a cluster of models that VRSG has aggregated for efficiency/performance purposes. In those cases, the assigned IDs of such clusters start at ID number 65536.

Coordinate systems for model placement

VRSG supports a variety of coordinate systems for specifying model locations in the vrsg.clt file. Simple XY coordinates can specify the location of the model if prefixed by:

```
! UTM Meters in zone NN
```

For example:

```
! UTM Meters in zone 38M
! agl=off
227488.439 9959488.751 3.178 0.00 0.00 0.00 kismayo-fence-0004.hpy
227440.775 9959506.484 2.425 0.00 0.00 0.00 kismayo-fence-0001.hpy
227370.607 9959559.630 1.404 0.00 0.00 0.00 kismayo-fence-0003.hpy
```

When VRSG encounters the UTM zone directive while parsing the vrsg.clt file, further XY locations will be interpreted as being relative to the given UTM zone (such as 38M in this example).

Geodetic coordinates can be used in the vrsg.clt file using the following example syntax:

```
! LL Coordinates
N35 13 0 W117 32 12 720 168 0 0 FV103.GB.desert.hpy
```

The above example would place the model at 35 degrees, 13 minutes, and 0 seconds north, and 117 degrees, 32 minutes, and 12 seconds west, at an altitude of 720 meters MSL. You can enter decimal degrees instead of deg/min/sec using the same syntax. Simply include the fractional part of the degrees, and leave minutes and seconds zero, as shown in this example:

```
N35.2166 0 0 W117.5366 0 0 168 0 0 FV103.GB.desert.hpy
```

Military Grid Reference System (MGRS) coordinates can be used in the vrsg.clt file using the following example syntax:

```
16S FQ 4521 6603 720 168 0 0 FV103.GB.desert.hpy
```

MGRS eastings and northings can be entered with 1, 2, 3, 4, or 5 digits of accuracy.

Ground clamping

By default, elevations are interpreted as being above mean-sea-level (or height above ellipsoid). You can instruct VRSG to interpret elevations as above-ground-level (AGL) by adding the `! agl=on` command to the vrsg.clt file as described earlier in this section.

Clustering model instances

By default, VRSG clusters culture model instances and neighboring models at runtime as a means of optimizing performance. To turn off clustering, use the `! cluster=off` command, and `! cluster=on` to resume clustering.

Orientation

By default, a model is given the orientation specified by the yaw, pitch, and roll parameters in the vrsg.clt file. Orientation clamping orients a model such that it sits naturally on the terrain surface, conforming to the roll and pitch of the local terrain surface. This means you can force VRSG to orient the model such that it observes the intended heading (yaw), but ignores the pitch and roll values in the vrsg.clt file, using instead values that are consistent with the local terrain surface. To enable orientation clamping, add the following command:

```
! orient=on
```

Orientation clamping remains enabled for all subsequent models in the vrsg.clt file until it is explicitly turned off. To instruct VRSG to return to using the given roll and pitch parameters directly, add this command:

```
! orient=off
```

Damage state and other appearance modifiers

To add dynamic appearance attributes such as a damage state or fire or smoke effects to a model, add the `-appearance=X` command to the line following the model. The appearance value is the DIS appearance mask given in hexadecimal without the "0x" prefix.

The following are settings for common appearance changes:

- Destroyed is 0018.
- Burning is 0020.
- Flaming is 8000.

The appearance is a bitwise OR of all the specified settings; for example to specify a destroyed, smoking, and flaming T-72 model:

```
12051 45722 -1 0 0 0 T-72-M4.CZ.camo.hpy -appearance=8038
```

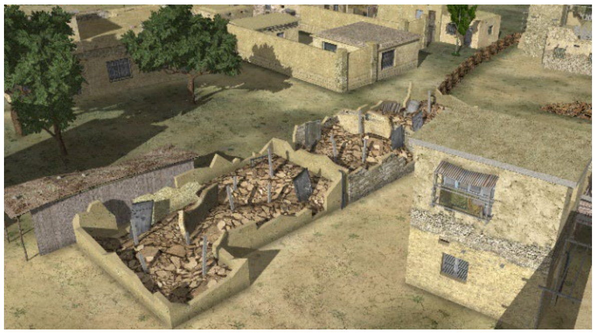
The complete set of DIS appearance masks is documented at: www.sisostds.org.

All of MVRsimulation's entity models have damage states. MVRsimulation is now standardizing on building damage states into its culture models. Thus far, VRSG is delivered with over 300 culture models that contain three damage states. Most of the models are buildings, a few are tree models. The following example shows one building (with the orange awning) in various damage states.

No damage.

Minimal damage (doorway and roof behind the building).

Partial damage.


Full Damage.

By lightly rolling the middle mouse button (wheel) over a building with damage states in the VRSG scene, you can preview the three damage states. If you roll the middle mouse wheel over several buildings and trees, you can see them all in damages states, as shown next:

Damaged area.

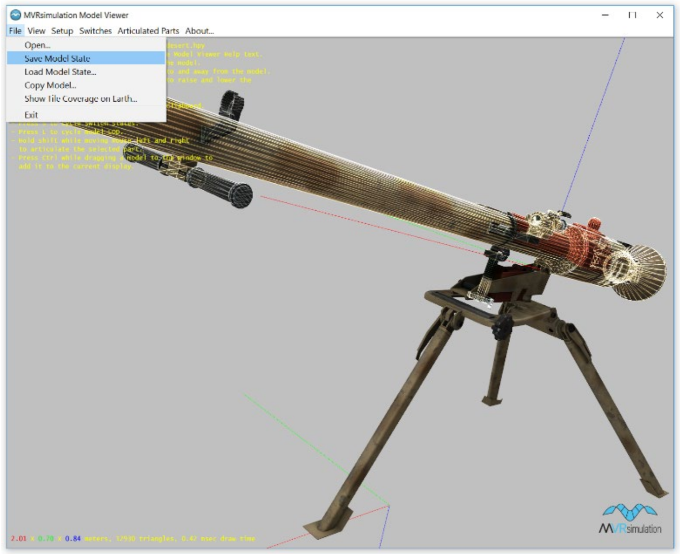
In Scenario Editor, which is delivered with VRSG, you can set these building to change to damage states within a scenario, as described in the *Scenario Editor User’s Guide*.

The culture models that currently contain damage appearance states are:

- Afghan-building-3B-001 to Afghan-building-3B-154.
- Kismayo-building-0008, Kismayo-building-0088, Kismayo-building-0107, Kismayo-building-0115, Kismayo-building-0130, Kismayo-building-0445, Kismayo-building-0466, Kismayo-building-0505, and Kismayo-building-0506.
- All building models prefixed “Hajin-Building-*”
- Defoliated Tree-cassia-001, Tree-cassia-005, Tree-elm-002, Tree-elm-006, Tree-poplar-004; these tree models also have a different appearance for each of the 4 seasons.
- You can preview all MVRsimulation’s culture models that have damage states at: https://www.mvrsimulation.com/3DContent/Damage_State_Culture.shtml.

Specifying a model state

You can view a model in the Model Viewer and manipulate its articulated parts, switch states, light maps, appearance, and thermal hotspots for IR mode. From the Model Viewer's File menu, choose the Save Model State command to save the current state of the model. This action produces a file with the saved state, using the name of the model, but with a ".prt" extension. When VRSG sees a PRT file associated with a given model, it will automatically initialize the model state with the values saved in the PRT file.



You can rename the saved PRT file to a unique name, and assign it to a specific instance of a model in the vrsg.clt file. In this way, you can save multiple states of a model. For example:

```
12051 45722 -1 0 0 0 T-72-M4.CZ.camo.hpy -prt=destroyedT72.prt
```

For more information about the Model Viewer see the chapter "Previewing Models, Effects, and Terrain."

You can specify the state of a model's articulated part in the terrain's vrsg.clt file with the `-setPartValue` command instead of specifying the external PRT file. This command enables you to save the degree-of-freedom (DOF) for an articulated part.

The syntax of the command is:

`-setPartValue(`*`partId`*`, `*`value`*`)`

See the "Specifying the initial state of an articulated part" section earlier in this chapter for an example of the syntax.

You can use multiple `-setPartValue` commands in one entry in the vrsg.clt to set the values for multiple parts of a given model.

Attaching a model to another model

In VRSG you can attach a static or entity model to another entity, by using the either the `-attachModel` command in a ModelMap.ini entry (as described earlier in this chapter) or by using an `-attachEntity` command in an entry in the cultural feature file. This command is useful for cases such as attaching radars or vehicles to ships, or drivers or pilots to vehicle entities. In one case you use `-attachEntity` with assigned x,y,z coordinates for the attachment point; in the other case you assign a .bvh animation.

Adding static characters to vehicle entities

The \MVRsimulation\VRSG\Animations directory contains a set of driver, passenger, and weapon-holding animations for adding static characters to most military ground vehicles as well as many aircraft in the military model library. To add a static character to a ground vehicle or aircraft, add the `-attachEntity` command to the model's entry in the terrain's vrsg.clt file.

The syntax of the command is:

```
-attachEntity=site:host:entity -bvh=<.bvh>
```

You specify in the `-attachEntity` command the entity ID (site:host:entity) of the vehicle to which you intend to attach a character.

This example also shows use of the -bvh command to assign a BVH animation to a static model. For more information about how to add static models directly to the terrain, see the section "Adding static features directly to the rendered scene." For information about working with 3D characters, see the chapter "Using 3D Characters in VRSG."

This next example shows a character holding a Vector 21 binocular/rangefinder:

The entry for the vrsg.clt cultural feature file for this character is:

```
! id=3998
N36 33.6 0 E056 24 0 1.0  -90.00 0 0   human-us-soldier-039.hpy -
human -weapon=weapon-Vector21B.hpy  -bvh=JTAC-standby-Vector21-
firing.bvh -attachEntity= WPB-87329 -attachOffset=7.4,-2.1,-2.24
```

Note in this example the `-weapon=` command, which assigns a weapon to a character also assigns an accessory model like the Vector-21. If the ID of the entity to which you want to attach a character is not known in advance, use the marking of the entity, if that is known. Instead of using the syntax: `-attachEntity=site:host:entity` use the syntax:

```
-attachEntity=<marking>
```

where `<marking>` is the DIS marking text of the entity.

The attachment point is specified with –attachOffset=. The attachment offset is expressed as:

```
-attachOffset= x,y,z
```

where *x, y, z* are the intended coordinates of the attachment point on the attached/parent model. To obtain the coordinates of the attachment point, open the model in the Model Viewer and place the cursor at the intended attachment point. Doing so displays the cursor's location coordinates. (See the chapter "Previewing Models, Effects, and Terrain" for information about the Model Viewer.)

Note: You can also use VRSG Scenario Editor, which is delivered with VRSG, to attach characters to vehicle entities as drivers, passengers, and gunners. For more information about Scenario Editor, see the *MVRsimulation Scenario Editor User's Guide*.

Attaching munitions, radars, or other entities to vehicle entities

VRSG's library of military models contains a set of munitions you can add to vehicles. To attach a munition model to an entity listed in the Modelmap.ini, add the `-attachEntity` command to the model's entry in the terrain's cultural feature (vrsg.clt) file. (Note that the command is *not* to be added to the ModelMap.ini file.)

The syntax of the command is:

```
-attachEntity=site:host:entity -attachOffset= X,Y,Z
```

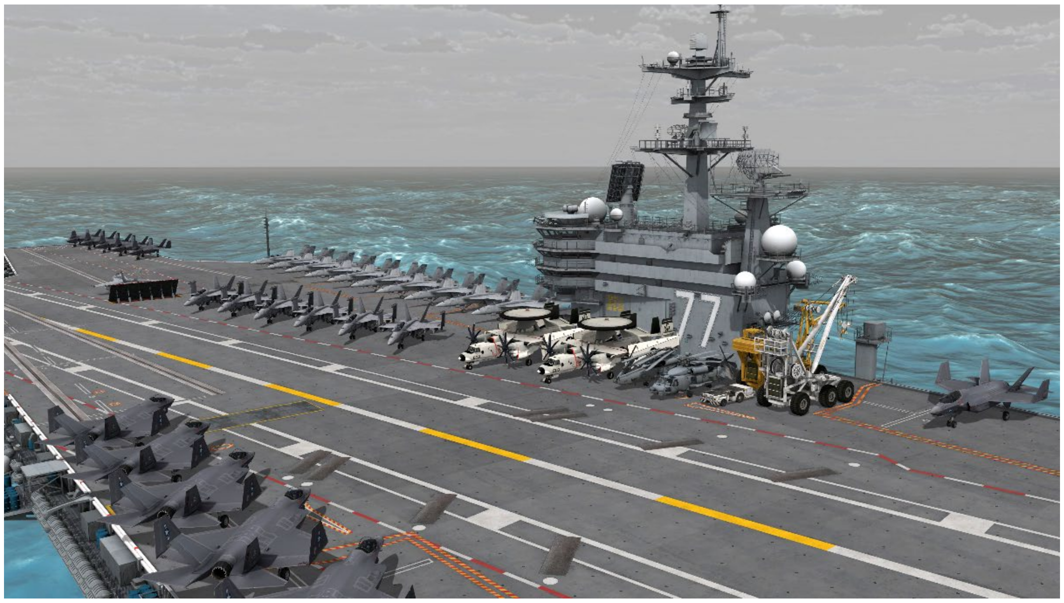
The following example vrsg.clt entry attaches the AGM-88 model to the F-16 entity:

```
! id=1044
N0 0 0 W0 0 0 0 0 0 0 AGM-88.US.grey.hpy -attachEntity=ID -
attachOffset=X,Y,Z
```

In the above entry, the position is irrelevant because the AGM-88 model will be attached to the F-16 entity. The ID should be the marking or the entity ID in site:host:entity syntax.

For the offset, replace *X, Y, Z* with the intended attachment coordinates for the munition model. To determine the attachment point, open the entity in the Model Viewer and place the cursor at the attachment point to display the cursor's location coordinates, press the letter "P" key on the keyboard. This action copies the x, y, z coordinates of the cursor position. (See the chapter "Previewing Models, Effects, and Terrain" for information about the Model Viewer.)

This next example shows several types of aircraft and ground vehicles on the flight deck of the CVN-77.US aircraft carrier model:



Entries for the vrsg.clt cultural feature file for this scene include the following:

```
! id=10100
S00 22 43.644 E042 32 27.102 -0.00  90.00 -0.00 0.00
MH-60R.US.grey.hpy -attachEntity=CVN-77.1 -attachOffset=-59.1,21.0,
-19.91 -appearance=80102000000

! id=10203
S00 22 43.644 E042 32 27.102 -0.00  0.00 -1.90 0.00
E-2C.US.grey.hpy -attachEntity=CVN-77.1 -attachOffset=61.0,-6.9,
-20.45 -appearance=12400000

! id=10206
S00 22 43.644 E042 32 27.102 -0.00  90.50 -0.00 0.00   A-S-32A-
36.US.white.hpy -attachEntity=CVN-77.1 -attachOffset=-74.8,24.8,
-17.45

! id=10210
S00 22 43.644 E042 32 27.102 -0.00  -90.00 -0.00 0.00
F-35C.US.grey.hpy -attachEntity=CVN-77.1 -attachOffset=-88.0,32,
-19.51 -appearance=80102000000
```

Note the entries for the stationary aircraft in this scene also use the models' wheels-down appearance bit with the `-appearance` command.

Adding threat domes as culture

In addition to creating threat domes in a ModelMap.ini file as described earlier, you can also create them as static models in a cultural feature (vrsg.clt) file. Instead of using a model name in an entry in the CLT file, use one of the following syntaxes to display a semi-ellipsoid or cylindrical threat dome:

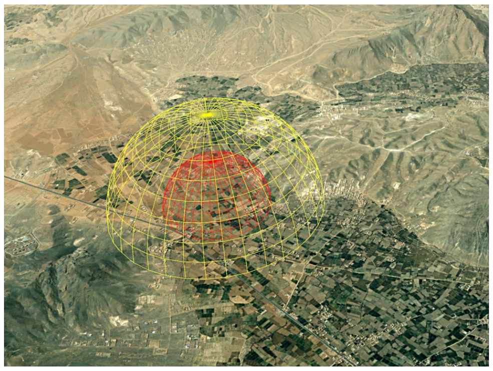
```
bubble(horzRadius,vertRadius,R,G,B)

cylinder(radius,height,red,green,blue)
```

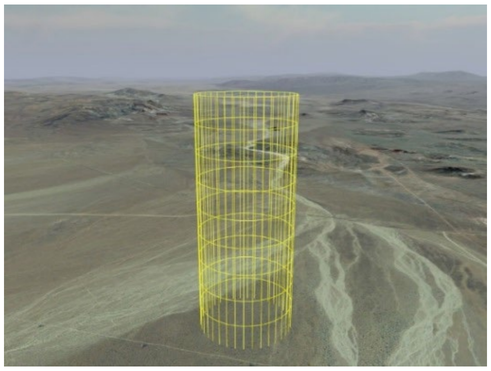
The following example describes a yellow threat dome with a horizontal radius of 4000 meters and a vertical radius of 1000 meters:

```
N35 0 0 W117 0 0 1234 0 0 0 bubble(4000, 1000, 255,255,0)
```

The next example was used to create the dome in the scene below on the left:

```
! id=1004
513042.65 3804766.49 1845.46   0.00 -0.00 0.00
bubble(1000,255,255,0)
! id=1005
513042.65 3804766.49 1845.46   0.00 -0.00 0.00  bubble(500, 1000,
255,0,0)
```





Placing light points on the terrain

A light point is a light source that emits rays of light in all directions from a specific location, but does not cast illumination onto other surfaces. Runway lights are typically light points.

You can add light points to the terrain in two ways:

- In MVRsimulation's Terrain Tools you can use the Cultural Lights terrain feature type to create a shapefile to add a large number of light points to the terrain, like a region's road network, with attributes for color, size, and spacing. This shapefile will compile the light points into the terrain. (See the *MVRsimulation Terrain Tools for Esri ArcGIS Pro User's Guide* for more information.) You can turn off the display of a terrain's compiled light points for ground-based simulations in the More Graphics Options dialog box in the VRSG Dashboard.
- In the terrain's vrsg.clt cultural feature file, you can add an entry for a light point model, as described below.

To create the effect of rows of lights such as runway lights, you first create a light point model in Presagis Creator, or by hand in Notepad as described in the chapter "MVRsimulation 3D Model Format." Once you have created the model, you edit the appropriate vrsg.clt file to specify the location of the light, as shown in the following entries:

```
! LL coordinates
! clamp_lp=on
! agl=on
```

```
N35 16 25.59 W116 37 21.09 0.0 320.5 0.0 0.0 lt_ap_bikelake.hpx 0 1
N35 16 25.59 W116 37 21.09 0.0 320.5 0.0 0.0 lt_rw_bikelake.hpx 0 1
N35 16 34.28 W116 37 24.96 0.0 320.5 0.0 0.0 lt_rt_vasibar.hpx 0 1
! clamp_lp=off
```

Individual vertices of a light point model can be automatically adjusted by VRSG at runtime to conform to the terrain profile. To enable this feature, add the following entry to the vrsg.clt file:

```
! clamp_lp=on
```

When VRSG sees this directive, each light point vertex is modified so that the light point elevation value (Z coordinate) is interpreted as above ground level (AGL). To disable light point ground clamping, add this entry to the vrsg.clt file:

```
! clamp_lp=off
```

Summary of available CLT commands

The following table describes the available commands for entries in a cultural feature (vrsg.clt) file:

Command	**Description**
-appearance=*x*	Specifies appearance bits to be applied to the model.
-attachEntity=*S:H:E or* -attachEntity=*marking*	Specifies the site:host:entity or marking of the entity to which the model should be attached.
-attachOffset=*x,y,z*	Specifies the coordinates of the attachment point, given in the coordinate system of the entity being attached to. Values are in meters, and use the DIS/CIGI axis convention of x= forward, y= right, z= down.
-bvh=<bvh-file>	Assigns a BVH animation to a human character.
clamp_lp= on/off	Toggles whether light point elevation values will be clamped relative to the terrain surface or interpreted as absolute elevation values. Use with agl=on and names of light point models.
-human	Interprets the model as a human character.
-irHot=*E,T,G*	Sets the degree to which specific IR hotspots (engine, tracks/wheels, gun, are displayed for a model. Use 1.0 for full intensity of hot spots, 0.5 for half intensity, and so on.
-ir_Scale=*N*	Scales a model's overall IR intensity.
-material=*N*	Provides an explicit material code assignment to all polygons in a model for physics-based IR simulation.

Command	**Description (continued)**
-minLODRange=N	Controls the level of detail displayed for a model, by placing a lower-bound on the model's computed LOD range.
-normalMap	Enables the use of a model's normal map textures. Because normal maps use a great deal of video memory, you must explicitly enable them on a per-model basis.
-noSpecular	Disables specular highlights for the model.
-oceanDynamicsScale=	Scales the responsiveness of an oceanClamped model's dynamics. Values larger than 1.0 will make the model's dynamics more responsive; values smaller than 1.0 will dampen the entity's dynamic response. Small vessels have larger values than large vessels.
-prt=<prt-file>	Specifies a PRT file saved from the Model Viewer to articulate a model's parts, sets its switch states and appearance, and initializes its IR hot spots.
-radarRangeScale=*N*	Modulates the RADAR return intensity of a model.
-scale=*N*	Scales a model by the factor indicated by N.
-setPartValue(*partId*, *value*)	Sets the DOF rate for an articulated part of a model.
-shadowOffset=*N*	Specifies a vertical offset for planar projected shadows. By default, planar shadows are projected to the model's Z=0 plane.
-terrainModel	Considers this model when performing elevation lookups.
-utmModel	Converts a model, such as a runway model, that is in UTM projection to UTM WGS1984 projection. (VRSG natively supports geocentric WGS1984.)
-uvw=*U,V,W*	Assigns material properties for thermal IR computation.
-vehicle	Denotes a static model as a vehicle for IR material assignment; if a model is qualified as a vehicle, default material assigned is "100111 - steel 2.5 cm, 30 deg."
-weapon=<*hpy-file*>	Assigns a weapon model (and accessories like a cell phone, Vector21B, or shovel) to a human character.

These commands must be on the same line as the rest of the model entry, following the model name. For example:

```
N35 0 0 W117 0 0 0 0 0 0 truck-030.hpy -lod_scale=2
```

```
486953.07 4294452.87 0.00 0.00 0.00 0.00 lppt-lisboa-lights.hpx
-utmModel
```

Examples of using commands for handling static models on VRSG's 3D oceans can be found in the Kismayo Amphibious scenario that is installed with VRSG.

Identifying static models on the terrain

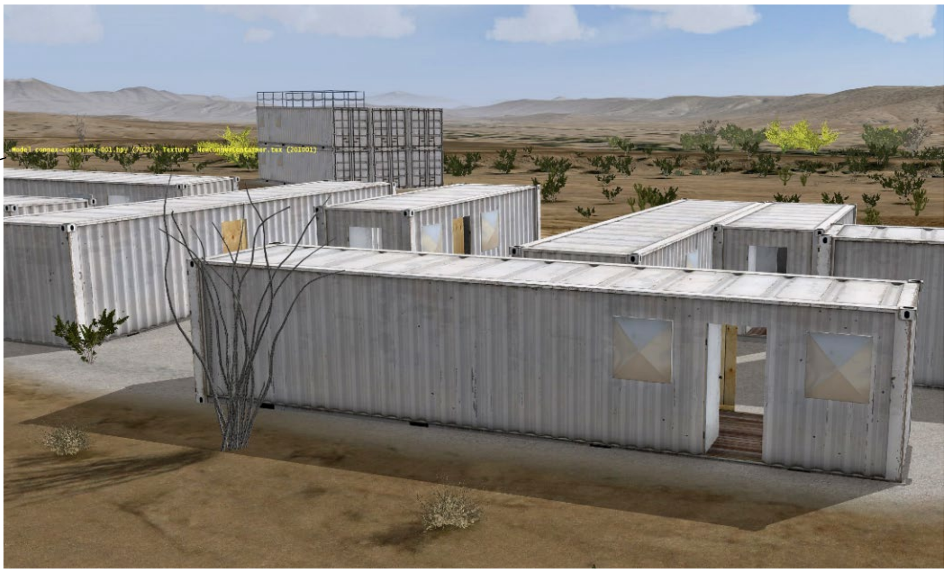
From within VRSG, you can identify a static (or entity) model used in a database by attaching to it. Doing so is useful for determining a model you want to replace, modify, or use again elsewhere.

To identify a model on a database being rendered in VRSG:

1. Run VRSG in Windowed mode or Desktop Cover mode so that you can use your mouse.
2. Position the cursor over the center of the model of interest. Press the middle mouse button/scroll wheel.

VRSG displays the name of the model and the texture that is applied to the face you have clicked, as shown in the following example:

The name of a model in a scene, and the texture of the face where you clicked.



```
Model connex-container-001.hpy (7022), Texture: NewConnexContainer.tex (201001)
```

In full-screen mode, you can identify a model by moving the eyepoint to position the model directly in front of you and then pressing Shift –L. Doing so displays the same information as shown in the example above. However, you have less control in specifying exactly the model you want to identify. (Press Ctrl F12 to display the names of the all the models in the scene.)

You can find out which cultural feature files VRSG is using in the last or current visualization session by inspecting the VrsgInfo_*.txt file, located in the \MVRsimulation\VRSG directory. This can be useful to verify whether VRSG is loading a particular cultural feature file.

Adding static features directly to the rendered scene

As mentioned earlier, using the VRSG Scenario Editor application is the best way to build up culture and refine model placement on terrain tiles, and to produce a cultural feature file as described in this chapter.

Two other ways of adding static culture to the terrain are:

- Editing a terrain's cultural feature file (vrsg.clt) to add or edit an entry for one or more model files. This includes large models comprised of buildings generated by CityEngine and converted to MVRsimulation's model format.
- Dragging a model file from Windows File Explorer directly to the VRSG visualization window and dropping it on the terrain. Press the J key on the keyboard to save the addition to the terrain's cultural feature file.

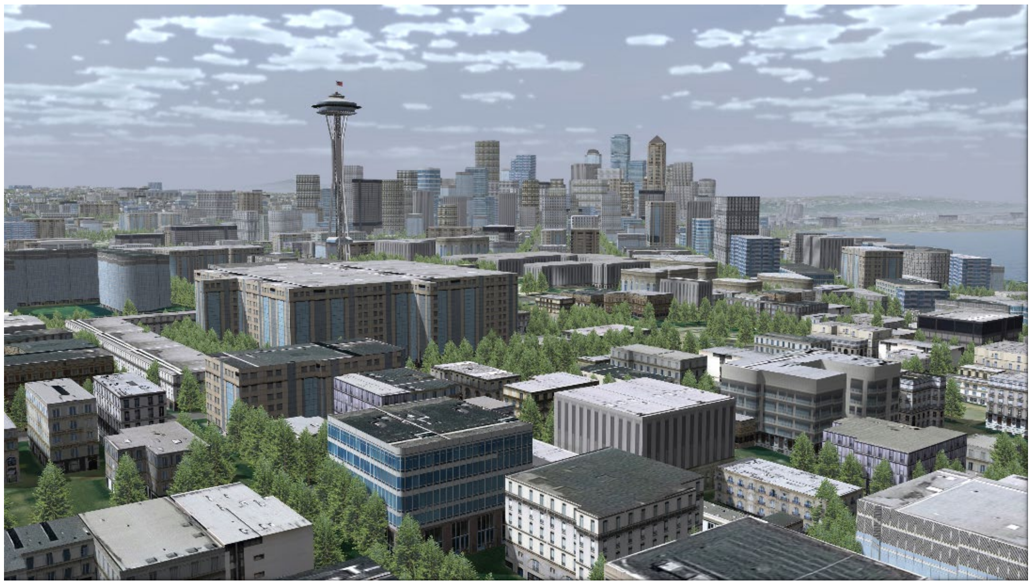
Although placing culture directly in the rendered scene in VRSG (described in the steps below) is supported, Scenario Editor offers much more control and more options for placing and orienting static models precisely on the terrain.

Editing a cultural feature file to add models

You can edit a terrain's cultural feature file (vrsg.clt) to place, replace or change the location of static models on the terrain, using the options and examples described in this chapter. If you make edits to a vrsg.clt file while VRSG is running, there is no need to restart VRSG to have your changes take effect. Simply save the changes, and then drag the modified CLT file onto the VRSG scene, where the updated version will replace the old version.

The following simple example illustrates the vrsg.clt file for MVRsimulation's Seattle, WA, terrain, which includes nine 20 km x 20 km models comprised of hundreds of buildings exported from CityEngine and converted to MVRsimulation's model format.

```
! UTM Meters in zone 10T
547508.17 5276833.67 71.8  0.00 0.00 0.00   Seattle_K6.hpy
550886.64 5276852.53 71.7  0.00 0.00 0.00   Seattle_K7.hpy
554072.41 5276872.94 45.8  0.00 0.00 0.00   Seattle_K8.hpy
547542.54 5271825.89 58.6  0.00 0.00 0.00   Seattle_L6.hpy
550921.14 5271860.68 65.7  0.00 0.00 0.00   Seattle_L7.hpy
553368.77 5271872.87 56.7  0.00 0.00 0.00   Seattle_L8.hpy
547588.99 5266815.05 78.3  0.00 0.00 0.00   Seattle_M6.hpy
550973.86 5266857.35 54.1  0.00 0.00 0.00   Seattle_M7.hpy
554341.26 5266876.58 56.2  0.00 0.00 0.00   Seattle_M8.hpy
! clt_version=2
! agl=on
548896.049 5274332.557 0 44.0033 -0 0 Space-Needle.hpy
548533.016 5275616.308 0 339.8598 -0 -0 kismayo-radio-tower-002.hpy -
scale=1.8
```



MVRsimulation's virtual Seattle, WA, terrain, with building structures extruded and textured in CityEngine, exported as nine 5 km x 5 km FBX models, and converted to MVRsimulation's model format.

Note: CityEngine is a standalone software product for rapidly creating procedural 3D models from building footprints and road vectors using OpenStreetMap (OSM) data. The product can export models in FBX model format or Collada DAE format for outputting as .FLT in Presagis Creator) which you can then convert to MVRsimulation's model format with MVRsimulation's conversion utilities. See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."

Dragging a model directly to the VRSG scene

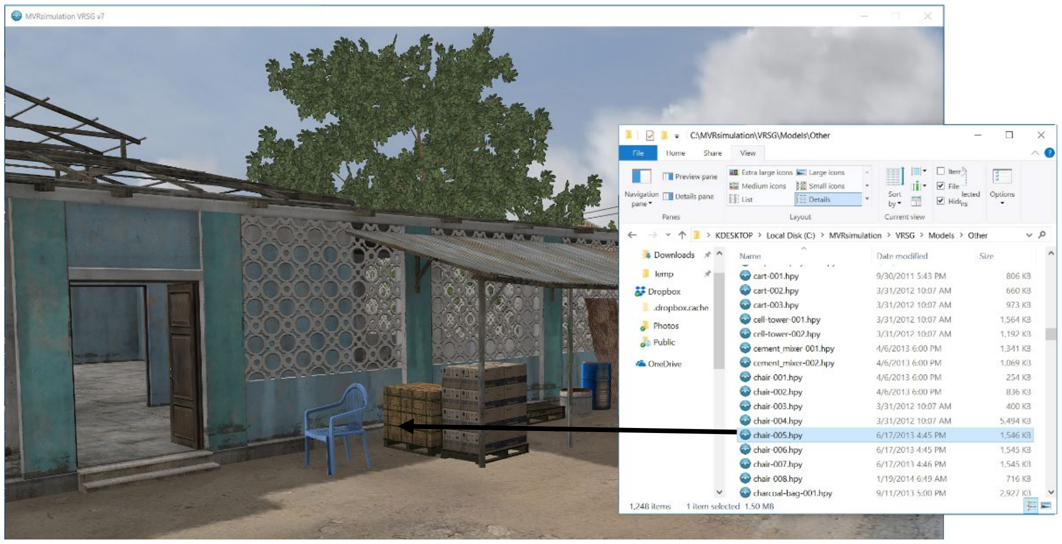
You can add models of cultural features, and static characters or static vehicles directly to the terrain in a "drag and drop" manner by dragging them from Windows Explorer to the terrain loaded in the VRSG visualization window, and then saving their position and orientation in the database's cultural feature file. This section describes how to add and reposition a model this way. Models must be in MVRsimulation's HPY or HPX model format.

Note: MVRsimulation *strongly* advises against using the steps described below to make edits to culture on a terrain that you have already edited in Scenario Editor. It is best to use the method described below to make minor adjustments to one or two culture models on 3D terrain not associated with scenarios created in Scenario Editor. For more information about VRSG Scenario Editor, see the *MVRsimulation Scenario Editor User's Guide*.

To add a static feature to the terrain directly in VRSG:

1. Start VRSG.
2. Using the 6DOF game controller, navigate to the area on the terrain where you want to place the model.
3. From the Windows File Explorer, select (with the mouse cursor) the model you want to place on the terrain, drag the model to the VRSG visualization window, and then place it (drop it) on the terrain by releasing the mouse button exactly where you want the model to be located.

If you drag-and-drop a model that has not already been loaded into VRSG, the visualization window will turn dark for a moment and then refresh, as loading of the new texture requires video memory resources to be allocated.



Name	Date modified	Size
cart-001.hpy	9/30/2011 5:43 PM	806 KB
cart-002.hpy	3/31/2012 10:07 AM	660 KB
cart-003.hpy	3/31/2012 10:07 AM	973 KB
cell-tower-001.hpy	3/31/2012 10:07 AM	1,564 KB
cell-tower-002.hpy	3/31/2012 10:07 AM	1,192 KB
cement_mixer-001.hpy	4/6/2013 6:00 PM	1,341 KB
cement_mixer-002.hpy	4/6/2013 6:00 PM	1,069 KB
chair-001.hpy	4/6/2013 6:00 PM	254 KB
chair-002.hpy	4/6/2013 6:00 PM	836 KB
chair-003.hpy	3/31/2012 10:07 AM	400 KB
chair-004.hpy	3/31/2012 10:07 AM	5,494 KB
chair-005.hpy	6/17/2013 4:45 PM	1,546 KB
chair-006.hpy	6/17/2013 4:45 PM	1,545 KB
chair-007.hpy	6/17/2013 4:46 PM	1,545 KB
chair-008.hpy	1/19/2014 6:49 AM	716 KB
charcoal-bag-001.hpy	9/11/2013 5:00 PM	2,927 KB

1,248 items 1 item selected 1.50 MB

When you release the mouse button, your model (the blue chair model shown in this example) will appear in the VRSG visualization window, placed in the scene on the ground under the cursor, facing north.

Optionally, set a viewpoint at the location of the new model placement so you can easily return to its location later.

By default, the model you add to the scene is placed on the ground below where you dropped it on the terrain, (regardless of whether Ground Clamp is selected in the Preferences tab). If you want to place a model precisely in space, that is, exactly where you drag and not on the ground, press the Ctrl key when you drag the model from Windows File Explorer to the visualization window. If you cannot see the model after you have placed it, zoom out from the scene a little bit—it might be that the eyepoint is located at the center of the model.

Refining model placement and orientation

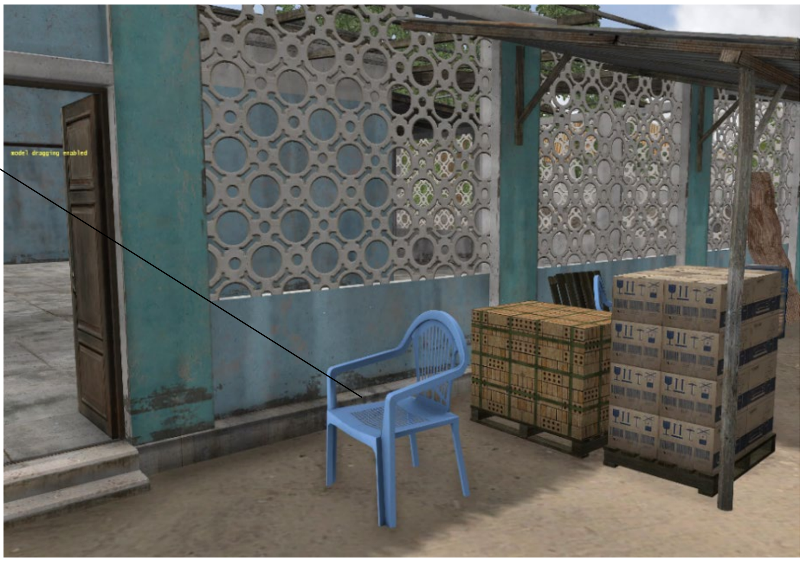
You can refine a model's orientation and location by attaching to it, enabling model dragging mode, and then moving it on the terrain.

To refine a model's placement and orientation:

1. Use the 6DOF game controller to position the eye point to the intended location relative to the model.
2. On the Dashboard Preferences tab, select the Ground Clamp Elevation and Ground Clamp Orientation options so that the model will remain seated on the terrain when you move it.
3. On the Attach Options tab, optionally set the attachment mode to Tether Free or Compass. (You can use other attachment modes, but these two are recommended for moving a model.)

4. Zoom in close to the model of interest and attach to the model by positioning the cursor over the center of the model and clicking the middle mouse button on it:

After attaching to the model, press Shift-D to turn on model dragging mode.



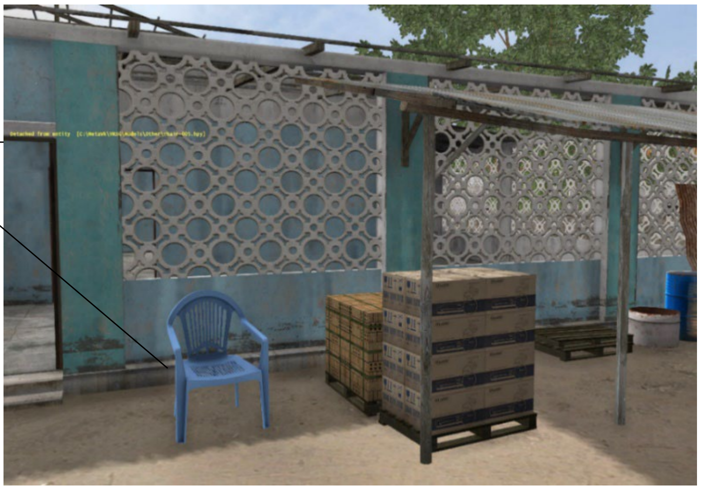
```
model dragging enabled
```

5. Press Shift-D. This keyboard command puts the model in "model-dragging" mode, that is, it enables you to move the model to which you are attached.

6. Move the model by moving the 6DOF game controller. Move the 6DOF game controller forward/backward or left/right to position the model along the terrain surface. Rotate the 6DOF to change the orientation (heading) of the model.

   The model not only moves and rotates in response to your game controller inputs; you can also pitch the model by turning the middle mouse button/wheel, or roll the model by pressing the Shift key while turning the mouse wheel. To change the pitch and roll of a model, be sure to unselect the Ground Clamp Orientation option on the Preference tab, as that option overrides pitch and roll to conform to the terrain. (The sensitivity of these rotations is controlled by the Rotation Gain setting on the Dashboard's Preference tab.)

Press Shift D again to turn off model dragging mode. And then press D to detach from the model.



```
Detached from entity  [C:\MetaVR\VRSG\Models\Other\Chair-005.bgy]
```

7. Once you have the model positioned as you want it, press the D key on the keyboard (or the 4 button on the SpaceMouse Pro) to detach from the model, and optionally press Shift-D on the keyboard to disable the model dragging function.

The model is now positioned in its new location on the terrain. This feature placement is temporary until you save it in the terrain's cultural feature file (vrsg.clt).

Save your changes to the terrain's CLT file by pressing the J key on the keyboard as described in the next section.

You can place several models on the terrain at once, by using one of these methods:

- Drag a model for each feature to the location you want it on the terrain, and then refine their positions.
- Lase multiple locations on the database, by left-clicking those locations to save their coordinates to VRSG's waypoints.dat file for later use in model placement. Edit the CLT file to add entries for all the new models, pasting in the coordinate locations for each entry from the waypoint.dat file and assign them unique IDs. In VRSG, attach to each model and refine the model placements.

If a model has an ID number, you can attach to a model not visible in the VRSG window through the Dashboard's Attach tab, by clicking Select as described in the chapter "Exploring the VRSG System." To specify the selection, specify the site and host as 0, and then the model's ID number.

Note: With a model's ID number, you can send a SetDataPDU to change its appearance, such as to turn buildings or bridges into a destroyed state.

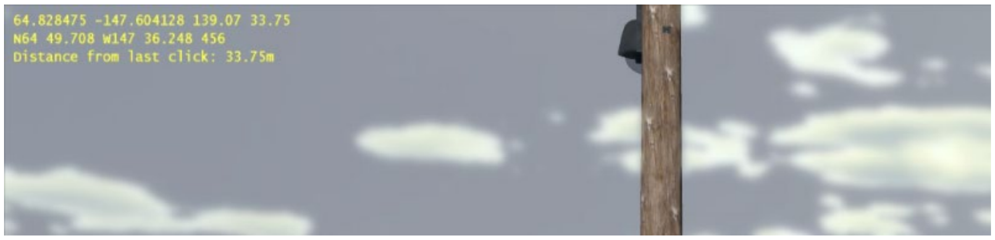
Press Shift-X to delete a static model that you are attached to.

Measuring the distance between two locations

To obtain the distance between two locations (for example, two models) in the VRSG scene, click each location.

The distance between the two points is displayed two ways:

- The distance is displayed briefly with the current coordinates in the upper-left corner of the screen as shown in the example below:



```
64.828475 -147.604128 139.07 33.75
N64 49.708 W147 36.248 456
Distance from last click: 33.75m
```

- The distance is written to the end of the last entry of the waypoints.dat file.

Each time you click in the scene, the location at which you clicked is written as an entry to the file \MVRsimulation\VRSG\waypoints.dat, in the form of lat/long/alt/distance from the last click. Most recent entries are appended to the end of the file. The rightmost field of an entry displays the distance (in meters) between last click and the prior executed click.

For example:

```
64.828416 -147.603430 139.26 83.71
64.828475 -147.604128 139.07 33.75
```

This example shows the distance between the two clicked locations is 33.75 meters.

Saving model placement edits

To save to save model placement edits to a terrain's vrsg.clt file, press the J key on the keyboard. This action updates the CLT file with the model's location and orientation state.

Each time you press the J key to save model placement edits to the vrsg.clt file, a backup of the previous version of the CLT file is created in the same directory. The CLT backup filenames are appended with sequential numbers, starting with "_000" for the first backup. This means the files would be named vrsg.clt, vrsg_000.clt, vrsg_001.clt, and so on. In the case where you might be using multiple vrsg.clt cultural feature files, if you modify a given model and then press the J key, the modified model will be written back to the same CLT file that originally contained the model. If you add new models directly to the visualization window and then press the J key, VRSG will add all new models to the default cultural feature file \VRSG\Models\vrsg.clt. To avoid potential confusion, consider not loading multiple CLT files if you plan to edit models directly in the VRSG visualization window.

VRSG preserves existing CLT files unchanged, but it will save new and modified CLT content in the coordinate system that is set in the Dashboard, with one exception: if you have the display set to MGRS coordinates, VRSG will write out the cultural feature file in UTM coordinates (MGRS coordinates are less precise).

As mentioned earlier, if you make edits to a vrsg.clt file while VRSG is running, there is no need to restart VRSG to have your changes to the edited file take effect. Simply save the changes, and then drag the modified CLT file onto the VRSG scene, where the updated version will replace the old version.

Note: If you have User Access Control (UAC) activated on your Windows system, the CLT file will not be saved or updated in the VRSG installation directory if the \MVRsimulation\VRSG\Models directory is located in C:\Program Files. Instead the updated CLT file will be stored in the directory C:\Users\<username>\AppData\Local\VirtualStore\Program Files\MVRsimulation\VRSG\Models.

To have the updated vrsg.clt file take effect the next time you start VRSG, copy the CLT file from the \VirtualStore\Program Files\MVRsimulation\VRSG\Models directory to C:\Program Files\MVRsimulation\VRSG\Models. To avoid having VRSG files written to the \VirtualStore directory, you could turn off UAC from the User Accounts control panel, if your site policy allows it, or install VRSG in a folder other than C:\Program Files.

Organizing models specified in the cultural feature file

The cultural feature file (vrsg.clt) updates of any models you have placed directly in the scene might contain absolute paths to the models when you are done. This means you can restart VRSG without having to set up the search paths for these models in the Advanced Parameters dialog box. However, you might need to organize the models and edit the paths in the CLT file if you distribute the database and associated CLT file to other users.

If a CLT file contains full path names for models, VRSG will prompt you to browse for the directory before attempting to load the database. You can either cancel and take no action, or select a directory into which to copy the models.

A way to organize them automatically is to add to the CLT file the following comment:

```
! copyToDir=<directory>\
```

This comment instructs VRSG to copy all subsequent models that you drag to the scene (and their referenced textures) to the specified directory. For example:

```
! copyToDir=e:\kabul\models
10257 11421 -1 0 0 0 carpet-display-001.hpy
```

The `copyToDir` instruction is turned off implicitly at the end of the vrsg.clt file, or can be turned off with just an empty specification:

```
! copyToDir=
```

To specify a path that contains spaces in the vrsg.clt file, enclose the full directory path in quotation marks as in the following example:

```
515470.836 3806455.683 0 341.6379 1.4124 1.3274
"h:\extra models\other\WFP-food-bag-stack-005.hpy"
```

If you edit a path by hand in the vrsg.clt file and need to specify a path that contains spaces, you must enclose the full directory path in quotation marks as shown in the example. To remove paths, copy all specified models to a local directory and retain only the model name, add the following command:

```
! stripPaths=on
```

To replace a model with another one, for example to replace one kind of car, building, or tree with another, you can simply open the vrsg.clt file in a text editor, and replace the name of the model that appears at the end of a feature entry. Be sure that the directory in which the model is located is part of the search path specified on the Startup Parameters dialog box.

If you need to verify which cultural feature file(s) VRSG is using in the last or current visualization session, you can do so by inspecting the VrsgInfo_*.txt file, located in the \MVRsimulation\VRSG directory.

Organizing culture for use with BSI's MACE

If you use Battlespace Simulation's (BSI's) Modern Air Combat Environment (MACE) with VRSG, you can also add cultural features directly in MACE, but doing so can become cumbersome for adding extensive areas of culture. A good rule of thumb is to add in MACE directly (via a vrsg.clt file) any culture that you will want to be destructible. Import the vrsg.clt into MACE (choose File > Import CLT). For all other culture, create your culture in Scenario Editor, export the scenario, and then add the exported vrsg.clt to VRSG's search path as you normally would. This culture will automatically load in VRSG for any MACE mission you load in that area. If you have culture that you want to use only in certain missions, name the vrsg.clt file something unique, for example, GraylingForest.clt, and add it to its own uniquely-named folder, and then add the folder to VRSG's search path. In this case, VRSG will not automatically load the specially named CLT file. For a given mission in MACE, choose Mission Settings > Database, and select GraylingForest.clt to load the culture for that specific mission. Bear in mind this action causes VRSG to remove any other CLT files it has loaded when you load that specific mission. For more information about using

VRSG with MACE, see the *BSI MACE and MVRsimulation VRSG Integration Guide* installed with your MACE license.

Resizing cultural feature models

You can obtain the height, length, and width dimensions for each model from the Model Viewer as described in the chapter "Previewing Models, Effects, and Terrain." (Also mentioned in that chapter is a way to measure the distance between two points on a model displayed in the Model Viewer.)

To increase or decrease the scale of a static model while visualizing it in VRSG, attach to the model, and then press Ctrl up-arrow to increase the size of the model, or Ctrl down-arrow to decrease its size. The model expands or shrinks by 1% each time you press the key combination.

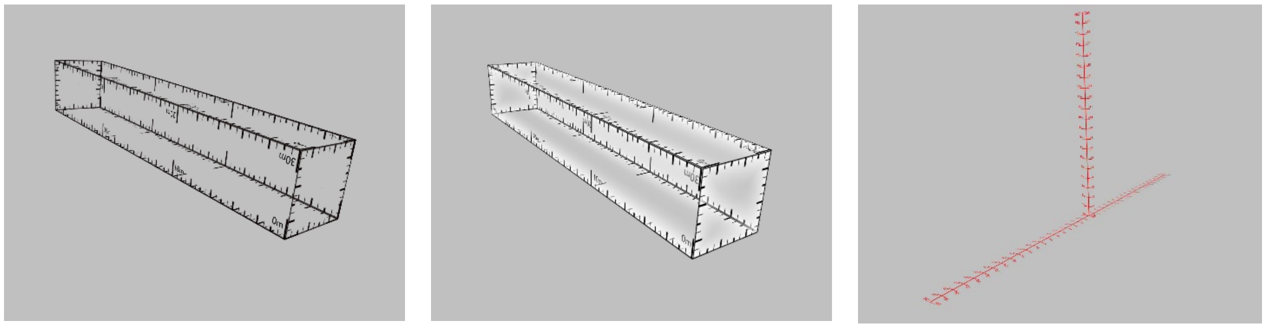
When you press the J key on the keyboard to save an updated version of a vrsg.clt file, VRSG appends the modifier `-scale` to the entry in the CLT file of the scaled model.

For example:

```
227424.663 9959532.97 1.507 45 0 0 kismayo-tree-acacia-001.hpy
-scale=1.2
227420.444 9959533.271 1.483 25 0 0 kismayo-tree-acacia-001.hpy
-scale=0.90
```

You can also specify a scale value directly in the CLT file; following the model name, you add the modifier `-scale=n` as shown in the example above.

To examine the scale of a model or distance between models in the context of a VRSG scene, you can use one of MVRsimulation's 3D ruler models, located in \MVRsimulation\VRSG\Models\Other. Use ruler-001.hpy, ruler-002.hpy, or and ruler-003.hpy to measure the length of a model or the distance between models. (See the section "Measuring the distance between two locations" for an alternate way of measuring distances between two points.)



Models ruler-001, ruler-002, and ruler-003 respectively, shown in the Model Viewer.

As with adding any other static model to the VRSG scene, in Windows Explorer select the file of the ruler model you want to place on the database and drag it to the location of interest in the VRSG visualization window. Attach to the model, enable model-dragging mode, and drag it to refine its location. Attach to it and press the X key to delete the model.

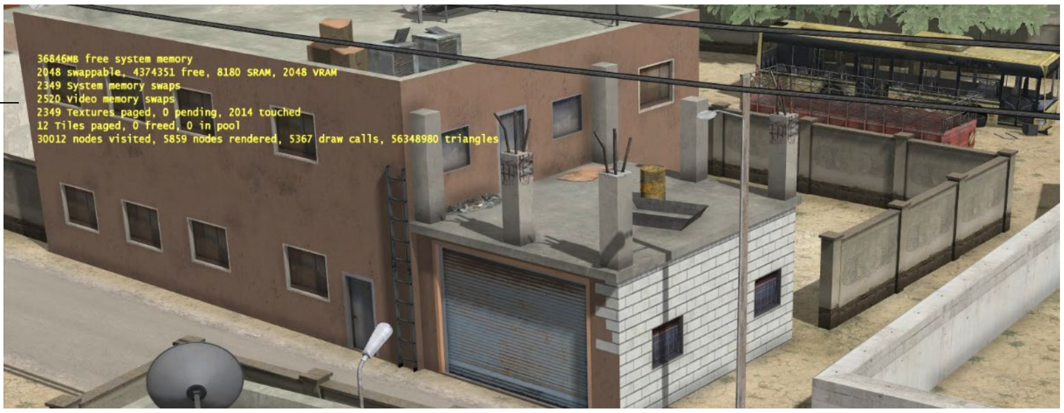
Models ruler-001, ruler-002, and ruler-003 respectively, shown in VRSG rendered scenes.

Accommodating use of large textures

If you add a large set of models to a large database at runtime via the vrsg.clt file, at some point you might exhaust the video memory and might need to allocate more video memory to VRSG. A message about a “thrashing texture” indicates that more textures are being requested from the scene than can fit into video memory.

You can reduce the far clip plane as a quick fix to try and reduce video memory demand. But if that is not appropriate or effective, check the number of video memory textures being allocated by pressing the T key, to display the system and texture memory statistics, as shown:

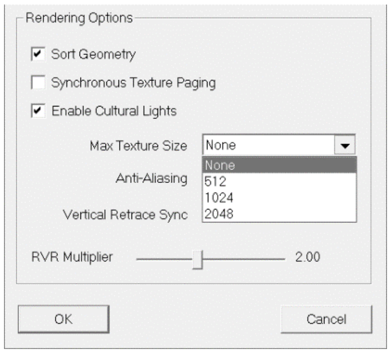
Press the T key to display system and texture memory statistics.



```
36846MB free system memory
2048 swappable, 4374351 free, 8180 SRAM, 2048 VRAM
2349 System memory swaps
2520 video memory swaps
2349 Textures paged, 0 pending, 2014 touched
12 Tiles paged, 0 freed, 0 in pool
30012 nodes visited, 5859 nodes rendered, 5367 draw calls, 56348980 triangles
```

MVRsimulation generally recommends you use a video card with at least 8 GB for visualizing high-resolution models. If you encounter problems visualizing large number of models with high resolution textures and you already have a video card of 8 GB or more, you can force VRSG to truncate texture sizes.

Select a value in the Max Texture Size field of the More Graphics Options dialog box on the Dashboard’s Graphics tab to force VRSG to downsize all loaded textures to 2048 x 2048, 1024 x 1024, or 512 x 512 pixels:


Rendering Options

[✓] Sort Geometry

[ ] Synchronous Texture Paging

[✓] Enable Cultural Lights

Max Texture Size: None (None, 512, 1024, 2048)

Anti-Aliasing

Vertical Retrace Sync

RVR Multiplier 2.00

The None option (the default) means that VRSG will not downsize any textures.

You can override the selected Max Texture Size option on a per-model basis by creating a file \Models\max_texture_sizes.txt, in which you list one model per line, followed by the maximum texture size for that model, as follows:

```
CVN-77.US.grey.hpy 4096
```

You do not need to know the maximum texture size for a given model; the 4096 entry shown in the example above would load the full-resolution model, even if its largest texture was only 2048 x 2048.

The Max Texture Size option, used with a max_texture_sizes.txt are useful tools when VRSG is running on an older machine with an older video card with limited VRAM and you want to limit the resolution of models that are less important in your simulation session.

Troubleshooting z-fighting

When two co-planar surfaces are rendered, sometimes flickering or flashing results are visible in the VRSG rendered scene. This z-fighting, which is an error in depth resolution (z-buffer), occurs when the graphics card tries to decide which item is on top. As this decision happens for every frame, flickering or flashing occurs when one item is rendered on top of the other, followed by rendering them in the reverse order in the next frame. Typical culprits are decals, windows, and runways.

To remedy z-fighting in VRSG, try adjusting the field-of-view (FOV), increasing the Near Clip, or decreasing the Far Clip settings on the Dashboard's Graphics tab to enlarge the depth bias. In 3D ocean simulation, turn off the Advanced Depth Bias setting to enlarge the depth bias.

If the z-fighting occurs in culture placed on the terrain in Scenario Editor, adjust the model(s) slightly in Scenario Editor to put a small bit of distance between the two affected planes.

If the z-fighting occurs on a converted runway model (which is not uncommon due to its multiple layers) or a City Engine urban model, see the remedies described in the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."

CHAPTER 6

Developing VRSG Plugins

VRSG provides a plugin interface, which enables a developer-user to develop software that VRSG executes at runtime. This plugin is a dynamically loaded library (DLL), which VRSG loads when it is initialized, and can potentially control VRSG's behavior.

VRSG supports a set of functions, which you implement inside the DLL. During runtime, VRSG calls these functions, passing the thread of execution to the developer-user's code. From within the DLL function, VRSG state data can be modified.

You use can use any development environment or compiler to create a plugin. This chapter assumes you are familiar with the C++ programming language, and software development in the Windows 64-bit environment.

An F-18 2D overlay on the VRSG scene rendered from the Battlespace Simulations' Modern Air Combat Environment (MACE) plugin.

Note: VRSG is a native 64-bit binary and is compiled with Microsoft Visual Studio 2019 (VS2019). If you are upgrading from VRSG 6.x, any user developed plugin at your site that extends VRSG's Dashboard interface with a property sheet (that is, a tab on the VRSG Dashboard), must be built, or rebuilt, with Visual Studio 2019. If your plugin does not extend VRSG's Dashboard, there is no restriction on the development environment you can use to build your plugin.

Because VRSG supports DirectX 11, any user-developed custom plugin which previously used DirectX 9 in VRSG version 5.x for drawing graphics or any action with the graphics API will need to be modified to use DirectX 11 with VRSG 7.

Plugin signatures and data structure definitions used as arguments are declared in MVRsimulation's Plugins.h header file. You can download the plugin.h header file from the MVRsimulation Download Server, in the /Software/Interfaces directory.

Customers on active maintenance who need an account on MVRsimulation's Download Server can request an account by sending a request to downloads@mvrsimulation.com.

Types of applications for VRSG plugins

VRSG exports network-based interfaces with CIGI and through the Distributed Interactive Simulation (DIS) protocol. Sometimes, however, it is more convenient or efficient to combine all the functionality of the collective system into the VRSG runtime. The plugin interface is designed to meet the needs of these types of applications. You can use plugins to:

- Control dynamics – your plugin could set the position and orientation of the eyepoint frame-by-frame, allowing you to integrate your own custom dynamics code. Within your plugin you could read joystick inputs, propagate the dynamics state, and have this reflected in the imagery generated by VRSG.
- Synthesize PDUs – your plugin could generate DIS messages and send them to VRSG for the purpose of creating and updating dynamic models or special effects.
- Add content to the VRSG scene – your plugin could render 3D and 2D primitives using DirectX 11 calls, supplementing the imagery generated by VRSG. This type of application is especially useful for creating custom 2D overlay displays, or augmenting the 2D HUD overlays that are built into VRSG with additional text or graphics.
- Access VRSG's frame buffer – your plugin can grab VRSG's frame buffer contents frame-by-frame. Potential applications include a digital contrast tracker or an MPEG codec.
- Capture mouse or keyboard events – your plugin can receive notification of mouse clicks on the 3D scene, and even identify which entity the cursor was over when the mouse button was clicked.
- Collect information about entities – VRSG can pass your plugin information about the entities it is managing in the scene.
- Query VRSG – your plugin can query certain VRSG properties, and/or request intervisibility or elevation lookups.

Overview of functions for VRSG plugins

VRSG supports a set of functions that it looks for in the set of DLLs that it loads. If these functions are present in the DLLs, VRSG will invoke them at the appropriate times during runtime. VRSG can support multiple plugins that implement the same entry point. VRSG will call each plugin's implementation in turn. If multiple plugins implement the same function, VRSG will call these functions in the alphabetical order of the plugin filenames. For example, if a plugin named HUD.dll was used to draw 2D overlays, and the H264.dll was used to encode video, H264.dll would be called after HUD.dll. Because of the calling order, the encoded MPEG output would include the graphics rendered by HUD.dll. If you want the encoded H.264 output to not include the graphics rendered by HUD.dll, you could simply rename HUD.dll to ZHUD.dll.

When VRSG looks for these entry points in your DLL, it first tries the symbol name generated by Microsoft Visual C++. If that symbol is not found in the DLL, VRSG will next try the ANSI-C version of the function name. If you are using a compiler other than Microsoft Visual C++, you will need to declare your functions as extern "C" in order for VRSG to load the symbols from the DLL.

Your plugin only needs to implement the functions you need for your application. VRSG will not attempt to invoke functions that are not present in the DLL.

The functions in Plugins.h are supported by VRSG at the time of this writing, and represent the level of control that VRSG customers have required to date. If you need additional control beyond these functions, contact support@mvrsimulation.com.

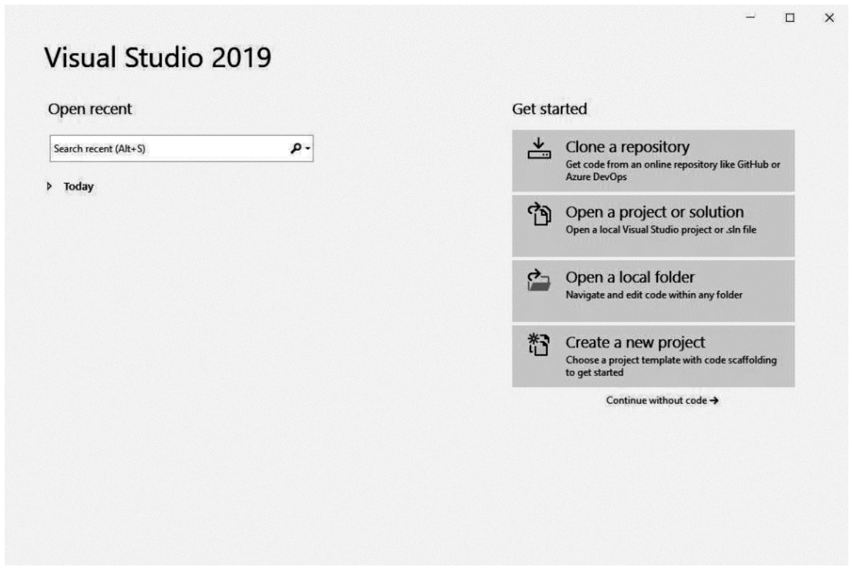
An OpenGL interoperability plugin is available which enables you to code 2D overlay graphics such as HUDs, in machine-native OpenGL for rendering in VRSG. (This plugin supersedes VRSG having to emulate it with a Direct3D layer.) Contact support@mvrsimulation.com for a sample Visual Studio 2019 project that illustrates how you can draw graphics using OpenGL and have them display over a VRSG scene.

Creating a plugin

This section describes how to set up a plugin project for Microsoft Visual Studio 2019. If you use a different development environment, see the documentation for that product for creating a dynamically-loadable DLL. The following steps illustrate how to create a plugin for VRSG.

To create a plugin:

1. From the File menu in Visual Studio 2019, choose Create a New Project.



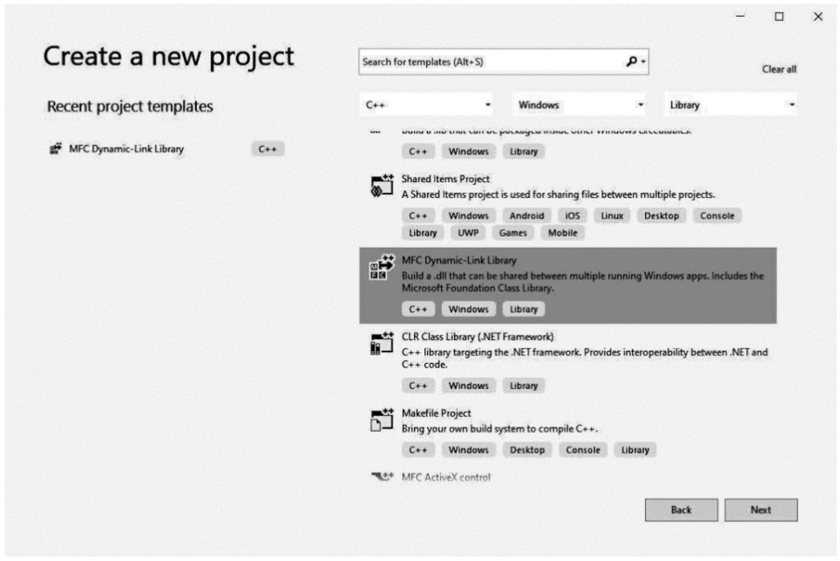
Get code from an online repository like GitHub or Azure DevOps

Open a local Visual Studio project or .sln file

Navigate and edit code within any folder

Choose a project template with code scaffolding to get started

2. Select MFC DLL from the project templates list.



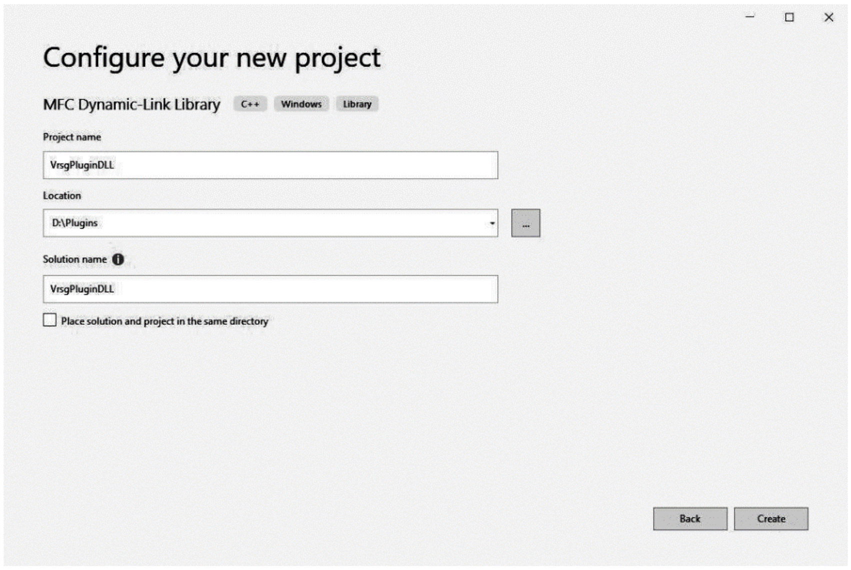
A Shared Items project is used for sharing files between multiple projects.

Build a .dll that can be shared between multiple running Windows apps. Includes the Microsoft Foundation Class Library.

C++ library targeting the .NET framework. Provides interoperability between .NET and C++ code.

Bring your own build system to compile C++.

3. Enter a name and a folder for your project.



Configure your new project

MFC Dynamic-Link Library C++ Windows Library

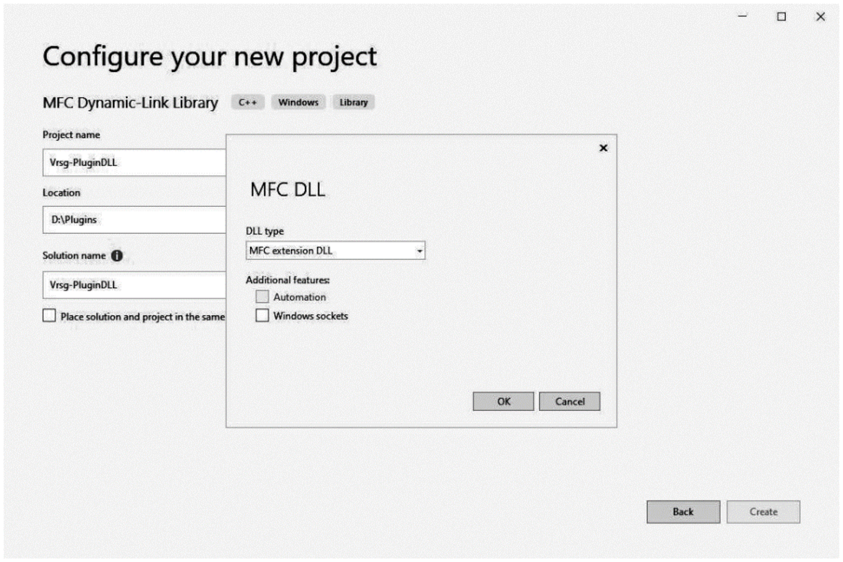
Project name: VrsgPluginDLL

Location: D:\Plugins

Solution name: VrsgPluginDLL

☐ Place solution and project in the same directory

4. Click Create.



Configure your new project

MFC Dynamic-Link Library C++ Windows Library

Project name: Vrsg-PluginDLL

Location: D:\Plugins

Solution name: Vrsg-PluginDLL

☐ Place solution and project in the same

MFC DLL

DLL type: MFC extension DLL

Additional features:
☐ Automation
☐ Windows sockets

5. Select the MFC Extension DLL option as shown above, then click OK.

6. Visual Studio will create initial source files named dllmain.cpp and MyVrsgPlugin.cpp. You can begin implementing your plugin entry points into these files, or add other source files to the project.

   VRSG is a 64-bit application. Therefore, for a VRSG plugin you should always build the 64-bit configuration of your DLL.

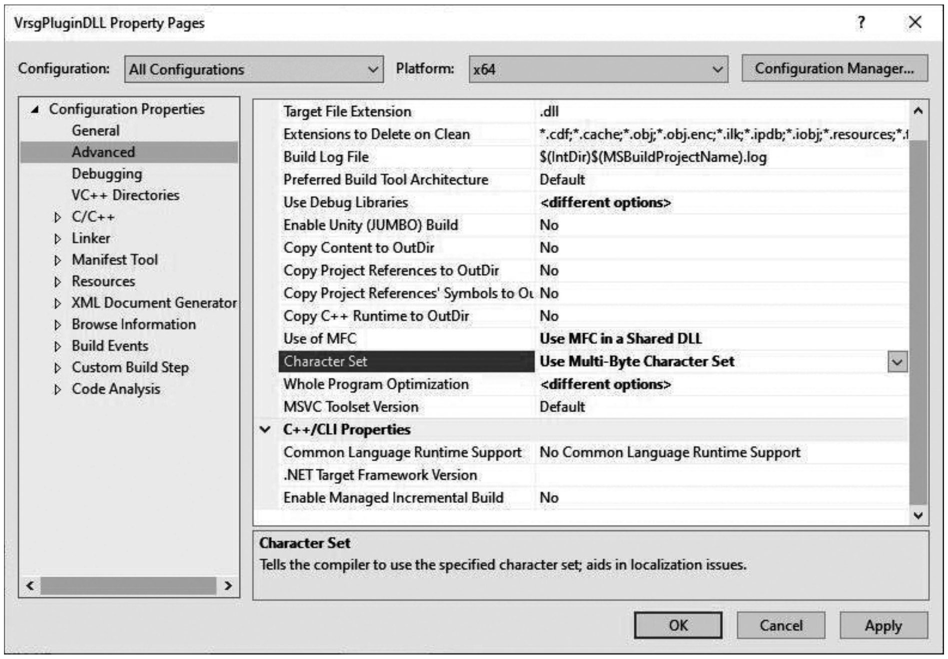
Now you are ready to build your project for VRSG. The result of a successful build will be a DLL in the directory \MyVrsgPlugin\x64\Release\MyVrsgPlugin.dll. You can manually copy this file to the VRSG\Plugins folder, or add a post-build event to your project to automatically copy the file after each build.

Adding a property page to VRSG's Dashboard

To add an Property Page to VRSG's Property Sheet, use the following form of extinitialize():

```
__declspec(dllexport) BOOL __cdecl extInitialize( CWinApp *theApp,
CPropertySheet *pPropSheet )
```

Change the character set to "Multi-Byte Character Set" using the projections settings as follows:



Target File Extension	.dll
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.
Build Log File	$(IntDir)$(MSBuildProjectName).log
Preferred Build Tool Architecture	Default
Use Debug Libraries	**<different options>**
Enable Unity (JUMBO) Build	No
Copy Content to OutDir	No
Copy Project References to OutDir	No
Copy Project References' Symbols to Ou	No
Copy C++ Runtime to OutDir	No
Use of MFC	**Use MFC in a Shared DLL**
Character Set	**Use Multi-Byte Character Set**
Whole Program Optimization	**<different options>**
MSVC Toolset Version	Default
C++/CLI Properties	
Common Language Runtime Support	No Common Language Runtime Support
.NET Target Framework Version	
Enable Managed Incremental Build	No

Character Set
Tells the compiler to use the specified character set; aids in localization issues.

Note: If your plugin extends VRSG's Dashboard, you can only use a Release version of your plugin with VRSG. You will not be able to run your plugin inside the debugger. To run your plugin in the debugger, the Debug build of your plugin must use this version of extinitialize():

```
__declspec(dllexport) BOOL __cdecl extInitialize( CWinApp *theApp )
```

Your debug version of the plugin will not have access to the Dashboard, so inputs must be obtained from other means (such as, registry defaults, read from a file, and so on).

Verifying your DLL is loaded by VRSG

To verify that VRSG loads your DLL properly, start VRSG with your DLL in the \MVRsimulation\VRSG\Plugins directory. If any dependencies are not satisfied, VRSG will issue a warning and will not load the DLL. VRSG outputs to its information log all functions it finds in a plugin. This information log is stored in \MVRsimulation\VRSG\VrsgInfo_<hostname>.txt. Examine this log to confirm that all of your entry points are recognized by VRSG. For every plugin loaded, VRSG outputs all found entry points; for example:

```
Loading plugin c:\MVRsimulation\vrsg\plugins\myvrsgplugin.dll
found extInitialize()
found extTick()
```

Distributing plugins in a shared environment

If your site has a VRSG directory structure shared across multiple computers, but not all computers need a particular plugin (plugins distributed with VRSG like the ones described in this chapter or plugins your site-specific plugins), you can make unique \Plugins directories by renaming the \MVRsimulation\VRSG\Plugins\ directory, to \Plugins*_hostname* where *_hostname* is the target computer for the plugin. VRSG will check for any \Plugins*_hostname* directories before it checks the \Plugins directory. This way, in the shared directory environment, different VRSG channels can load different plugins, or no plugins at all.

CHAPTER 7

Working With Sensor-View Modes and Physics-Based IR

A sensor view is an electronically enhanced out-the-window view that enables a user to discern elements of the world that are not readily apparent in the visual spectrum.

VRSG provides thermal sensor imagery that portrays the environment of a simulated device with true perspective and real-time responses to the movement of a vehicle and its sensor. Thermal sensors portray relative temperature differences in a given scene and are therefore able to provide information that is not available in the visible spectrum (that is, not available in the "out-the-window" view). The contrast in temperatures aids users in identifying and classifying the features and entities depicted in the image.

Working jointly with Technology Service Corporation (TSC), MVRsimulation has developed an advanced physics-based IR sensor modeling capability, which is available in the domestic release of VRSG.

This chapter describes VRSG's notional sensor-view features and MVRsimulation's physics-based IR simulation in VRSG, which includes a physics model.

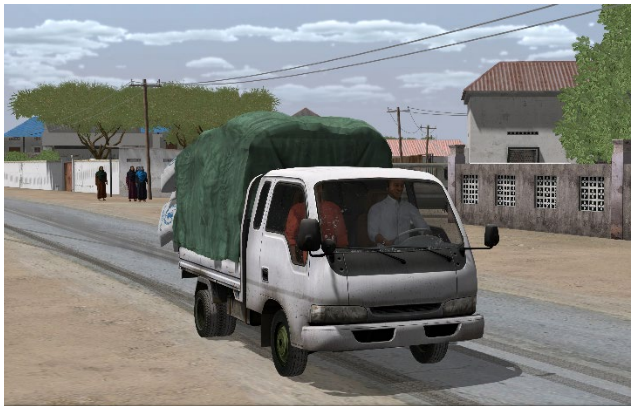
The VRSG sensor view is a user controllable image that is designed to model two levels of thermal views of fidelity:

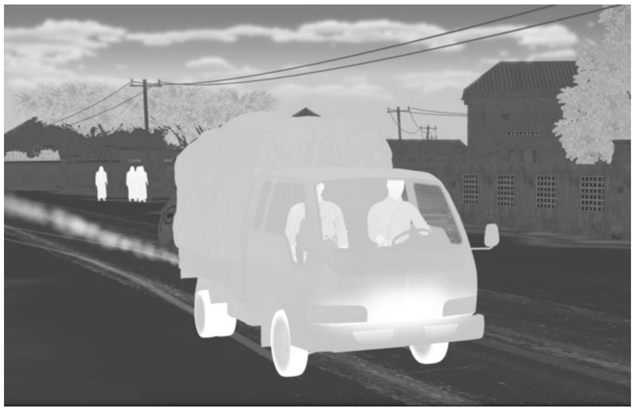
- Physics-based – VRSG's physics-based IR uses a physics-based model licensed from TSC, in conjunction with IR rendering technology developed by MVRsimulation, featuring real-time computation of the IR sensor image directly from the visual database, without the need to store a sensor-specific database. This real-time model combines automatic material classification of visual RGB imagery, and a physics-based IR radiance and sensor model.
- Material-based – On a per-material basis, you can provide thermal radiance profile data as a function of time-of-day. The fidelity of the radiance profile data is under user control. Notional data may be supplied for installations requiring ITAR export compliance. You can also provide radiance profile data that was derived from a third-party physics-based model.

VRSG's notional sensor-view features

Operation of the basic VRSG sensor view does not impact the capabilities of normal out-the-window visual channels. VRSG is capable of switching between out-the-window and sensor modes with single-frame latency.

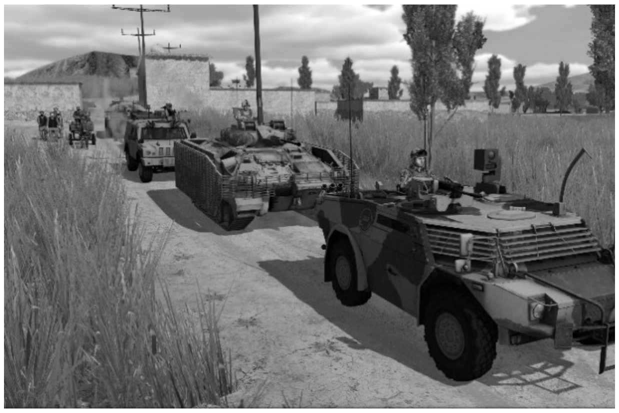
The images below depict a VRSG out-the-window view on the left and the corresponding notional thermal IR in white-hot mode view on the right. The sensor scene shows the heated engine and wheels of the truck entity; these parts are hot from the movement of the vehicles relative to the surrounding environment. The scene also shows the characters in the vehicle and standing on the street.


VRSG view with daylight conditions without sensor effects.


VRSG's notional white-hot IR view of the same scene.

A VRSG visual channel can dynamically switch between the following types of views, as shown in the following images:

- Normal out-the-window (OTW).
- Electro-Optic (EO) or daylight television mode (Day TV) sensor.
- FLIR (Forward-looking infrared) white hot, where thermally hot areas are gradations of white.
- FLIR black hot, where thermally hot areas are gradations of black.
- Fusion options of EO with White Hot or Black Hot.



You can also manually set a sensor view in VRSG by typing the keyboard letter "O", which gives you engineering-level access to the sensor mode.

Daylight TV (DTV) electro-optic (EO) sensor

The EO sensor simulated by VRSG captures the intensity component of the human's visible waveband. Thus a monochrome image is rendered, which is essentially a grey-scale version of the OTW visual scene. Since the imagery captured by the sensor is in the same spectrum as the human's visual sensor, the same atmospheric attenuation and illumination modeling apply.

EO sensor imagery can be degraded using the sensor post-processing effects presented in the previous section. An EO sensor can be switched to IR or OTW with single-frame latency.

Thermal infrared (IR) sensor

Both the physics-based IR and the material-based IR require material attribution information. In the case of 3D models, material attribution can be associated with a model's texture maps. For geospecific imagery, material attribution is assigned via classification of RGB color data. These processes are described in subsequent sections.

Both the physics-based and material-based models can benefit from these additional features:

- Alternate thermal textures – VRSG switches to a different set of texture maps when operating in IR mode. These texture maps encode the desired signature of the model to include localized dynamic hot spots. VRSG's military model library and human character library supply pre-built thermal textures for IR mode. To edit textures with hot spots for your own models, see the chapter "Manipulating Textures."
- Dynamic hot spots – Thermal textures can encode hot spots that are dynamically blended in as a function of vehicle activity. For example, as a vehicle starts moving, the hot spots for the wheels or tracks are faded in. Likewise, when a vehicle stops moving, the hot spot is faded out. Hot spots are also available for the engine compartment and the gun.
- Post-processing effects – Any sensor scene can be degraded with a variety of post processing effects, described in subsequent sections.

Sensor post-processing effects

Built-in sensor post-processing effects are available in all sensor modes, including the visual out-the-window mode. Post-processing effects can be changed on a frame-by-frame basis with zero latency. The post-processing effects, many of which are available on the Dashboard's Sensor tab, are:

- Noise – The scene can be degraded by a random noise pattern. Use the slider to adjust the amount of artificial noise of the scene in all sensor modes.
- Focus – The scene can be convolved with a Gaussian kernel to simulate the effects of optical focus. You can control the intensity (degree of blur) using the "sigma" parameter representing the shape of the Gaussian kernel. For IR modes, a small degree of blur is recommended, as it makes the scene appear less crisp and more characteristic of an IR sensor.
- Level – Adjusts the brightness of the scene in all sensor modes.
- Gain – Adjusts the contrast of the scene in all sensor modes.
- Digital zoom – Many sensors offer a digital zoom capability that zooms in on an image by multiplying the size of pixels. Digital zoom provides a blocky, aliased image as the pixel boundaries become visible at high levels of zoom.
- Motion blur effect – Simulates a blurred or smeared appearance of objects along the direction of relative motion.
- Heat refraction – Simulates the heat haze shimmer appearance of a scene when it is viewed through a layer of heated air (produced from conditions such as jet fuel exhaust).
- A/C banding – The simulation of the scrolling horizontal stripe that is produced by power supplies that have frequencies dissimilar to the vertical retrace period of the monitor.
- Automatic gain control – Maps the physics-based IR radiance range into a display dynamic range; uses a histogram analysis of the radiance image to determine an appropriate display dynamic range for the scene.
- Polarity inversion – Typically used by IR sensors but available to all sensors, VRSG can reverse the polarity of an image. This enables you to view the image using the polarity that yields the greatest perceived contrast.
- Depth-of-field – Simulates a range at which objects are in full focus, and a range at which objects are at their maximum blur range. The existing blurSigma parameter indicates the degree of blur for objects at the max blur range. The resulting image will render objects in focus at the given focus range, with increasing blur out to the maximum blur range.
- Dynamic range washout effect – Used to simulate sensor blooming effects. A value of zero results in no washout effect; a value between 0 and 1.0 increases image intensity while reducing contrast; a value of 1.0 results in a fully white image; a value less than zero increasingly reduces contrast and darkens the image; a value of -1 results in a completely black image.

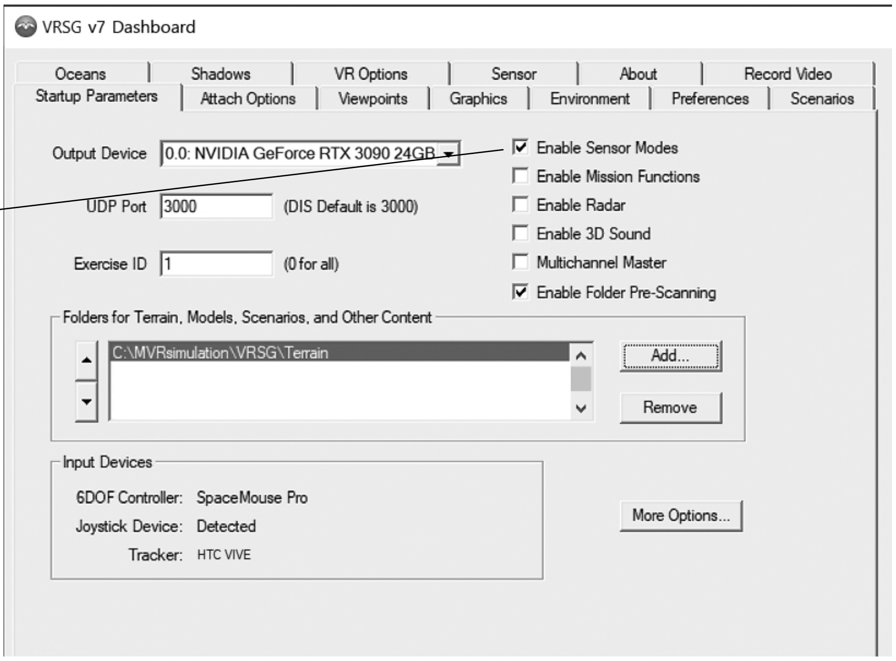
Setting basic sensor-view mode properties

In an integrated simulator, sensor mode properties would likely be controlled by a simulation host, using one of the network-based interfaces (e.g. CIGI, DIS). When using VRSG in standalone mode however, these properties may be controlled and previewed with the VRSG Dashboard.

In the VRSG Dashboard, you set the basic sensor-view properties on the Sensor tab. First, turn on sensor view mode by selecting the Enable Sensor Modes checkbox on the Startup Parameter tab.

The Enable Sensor Modes option turns on EO (DayTV), infrared (IR) White Hot and Black Hot, and night goggle vision (NGV) modes and two sensor fusion modes. It also activates the Sensor tab on the Dashboard. If the Enable Sensor Modes option is not selected on the Startup Parameters tab, you will not be able to use keyboard command “O” to cycle through the visual spectrum, and the Sensor Tab will not be displayed in the Dashboard.

Select this option only if you need sensor modes. By default this option is not selected, so that VRSG can load faster and consume fewer system resources.


VRSG v7 Dashboard

Output Device: 0.0: NVIDIA GeForce RTX 3090 24GB
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)

[✓] Enable Sensor Modes
[ ] Enable Mission Functions
[ ] Enable Radar
[ ] Enable 3D Sound
[ ] Multichannel Master
[✓] Enable Folder Pre-Scanning

Folders for Terrain, Models, Scenarios, and Other Content: C:\MVRsimulation\VRSG\Terrain

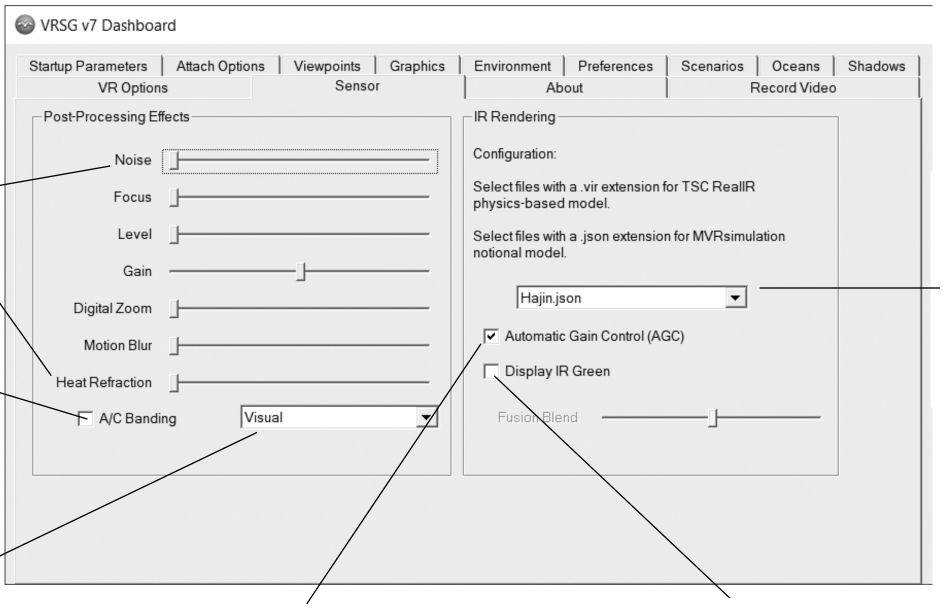
Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: HTC VIVE

If you do not intend to use sensor modes, leave the Enable Sensor Modes checkbox unselected; this way thermal textures will not be loaded unnecessarily.

Next, adjust various sensor effects on the Dashboard’s Sensor tab.

In VRSG, you can also choose a geospecific IR configuration file and optionally turn on automatic gain control, as shown above.

The Post-Processing Effects settings control scene characteristics such as simulated noise, contrast, and blur. See the previous section “Sensor post-processing effects” for a description of each option.



Configuration:

Select files with a .vir extension for TSC RealIR physics-based model.

Select files with a .json extension for MVRsimulation notional model.

Hajin.json

[✓] Automatic Gain Control (AGC)

[ ] Display IR Green

[ ] A/C Banding

Visual

Adjusts the sliders to modify the amount of the effect on the sensor display.

Displays a scrolling horizontal stripe to simulate banding produced by power supplies that have frequencies dissimilar to the vertical retrace period of the monitor.

Sensor mode options.

IR configuration file for physics-based or notional sensor simulation.

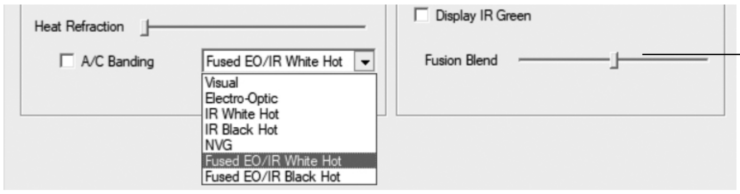
Automatically selects the radiance minimum and maximum range to which the display dynamic range is mapped.

Displays the scene in simulated NVG green if the display mode NVG is also selected.

The IR Rendering settings control the intensity (brightness) values for sensor scenes. For each option you specify a value in the range of 0 to 255. Note that the luminance values in this section are *only* for the non-physics-based mode of IR, and *will not* have any effect if you are using an IR configuration file (described later in this chapter) with the physics-based model.

- IR Configuration is the file that contains a physics-based or material-based sensor-rendering profile for the terrain being rendered, as described later in this chapter, in the section "Using VRSG's physics-based IR simulation."
- The Display IR Green option displays the White Hot and Black Hot and fusion modes in green, when NVG sensor mode is also selected.

- Use the Fusion Blend slider to adjust the ratio at which to blend either of the selected fused sensor mode options.



The fusion blend ratio of the selected Fused EO/IR White Hot or Black Hot sensor mode.

Sensor characteristics can also be controlled programmatically using CIGI. For more information, see the appendix “CIGI Version 4.0 Support.”

Controlling dynamic hot spots

Entities with thermal textures can encode dynamic hot spots for the wheels or tracks, the engine compartment, the main gun, or other custom components (such as rocket launcher tubes). These hot spots are controlled by events within the simulation. This section describes the details of these events.

Wheels / Tracks – These hot spots fade in as the vehicle begins motion, and fade out as the vehicle ceases motion. All that is required to activate this feature is to move the entity, either with the DIS protocol or with a CIGI component control as described in the appendix “CIGI Version 4.0 Support.”

Engine compartment – The engine compartment hot spot fades in when the “power plant” bit of the appearance mask is set. The appearance mask of an entity is conveyed in either the DIS Entity State PDU, or with a CIGI component control. If a simulation does not stimulate this bit, VRSG will turn it on automatically if the vehicle begins movement.

Main gun – The main gun hot spot will fade in when the entity issues a DIS FirePDU. The heated gun barrel will retain heat for a period of time following the fire, and will slowly cool down over time.

Custom components – the simulation may control the hot spot intensity of custom components identified by an integer ID. A model can have up to 16 controllable components. The wheel/track, engine compartment, and main gun components are controlled automatically by VRSG in response to simulation events. The custom components must be explicitly controlled by the controlling simulation (host), using CIGI or DIS interfaces.

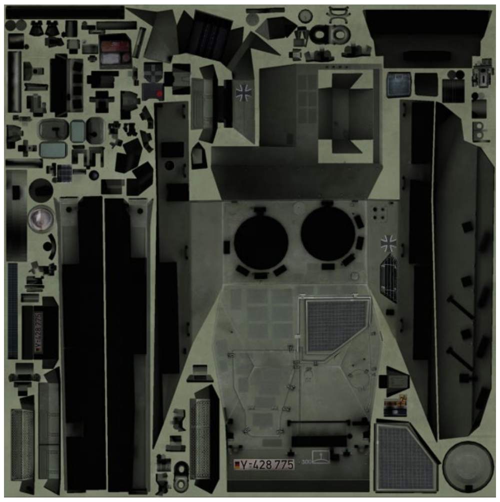
Note: You can preview a model’s hot spots in MVRsimulation’s Model Viewer, which will show all hot spots at full intensity. See the chapter “Previewing Models, Effects, and Terrain” for more information.

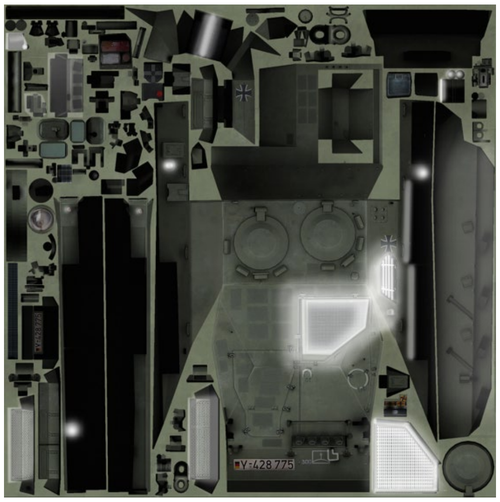
Editing textures for sensor-view mode

For sensor-view mode VRSG loads texture files that have the extension *_IR.*. The texture maps of the military vehicle models that are delivered in MVRsimulation’s 3D content libraries have IR alternate textures. The models include heat signatures or hot spots modeled into their IR textures. These IR heat signatures fade in and out in response to simulation events.

You can create alternate IR textures for any textures used in a VRSG simulation. For example, you could modify road surfaces or building fronts to make alternative sensor view textures. The chapter “Manipulating Textures” describes how you can create these textures.

The following example illustrates an original texture map for a model and how one can make an alternate IR texture by adding hot spots to the engine compartment area of a copy of the texture. To do so, first copy the original texture .bmp file to a new file *texture-name*_IR.bmp. Next, open the *texture-name*_IR.bmp file in an image editing application such as Adobe PhotoShop or in a modeling program such as Presagis Creator.





As shown in the image above, using an airbrush tool you could paint white hot spots on the engine compartment where real-world thermal bleed-through might occur. When you are ready to use IR texture alternates in a scene, place each IR texture file in the same directory as its originating texture.

To test your IR hot spot textures in VRSG in a standalone situation, press the O key on the keyboard. Doing so cycles through thermal and Day TV sensor modes and then returns to the normal mode.

You can also test an individual model's IR textures in the Model Viewer, as described in the chapter "Previewing Models, Effects, and Terrain."

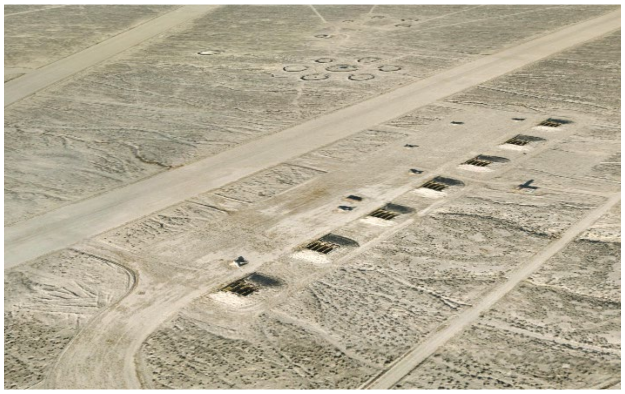
Using VRSG's IR simulation

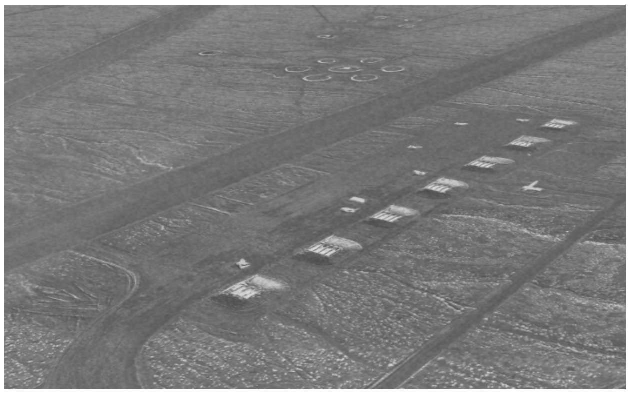
VRSG's IR simulation uses per-material radiance profile data which provides radiance information for each material over a 24 hour diurnal cycle. These files are stored in JSON files in the \MVRsimulation\VRSG\IR subdirectory. When VRSG starts, all json files in \VRSG\IR are discovered and an entry is placed for each into the IR configuration menu on VRSG's Sensor tab.

These radiance profiles describe how a material heats up and cools down over a 24 hour cycle. An entry is included for every material of interest. For geospecific imagery classification of RGB data, RGB values are included in addition to material codes.

These radiance profiles can be created manually, or by using our IRSetup utility. When using IRSetup, the physics-based Model from TSC is used to produce the radiance profiles. Manually created radiance profiles can be used also, in export-controlled situations where use of IRSetup is not allowed. With the file format understood, it would also be possible to use a third-party physics-based model to create the radiance profiles.

For geospecific imagery applied to terrain, VRSG allows for the real-time material classification of the visual database using the visual spectrum RGB colors. In other words, materials are inferred from the colors of the visual spectrum database. The construction and storage of a sensor-specific database is not required. The required sensor database information is generated on-the-fly from the visual spectrum database. As shown in the following example, which uses 2 cm per-pixel imagery captured by MVRsimulation's imagery data collection UAV, the higher the resolution of the imagery, the more detailed the resulting sensor view.

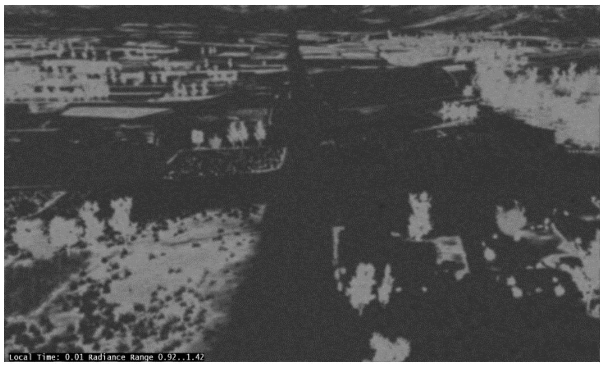

VRSG real-time rendering of 2 cm terrain built with imagery collected by MVRsimulation's SUAS of a target range at the Naval Air Station (NAS) Fallon Range Training Complex.


VRSG screen capture of the 2 cm 3D terrain on the left in sensor view. The higher the resolution of the visual database, the more accurate the IR profile for sensor simulation.

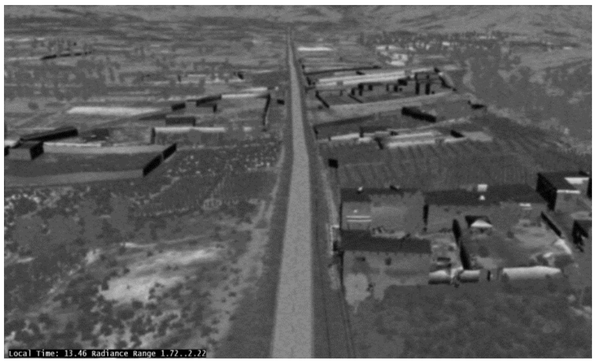
For scene elements that are not geospecific terrain, VRSG allows for the explicit assignment of material codes to these elements. Material codes can be assigned at the per-model level, the per-texture level, or at the per-polygon level.

The VRSG image below to the left shows MVRsimulation's Afghanistan village 3D terrain at midnight. Notice the warmer vegetation and the cooler asphalt road and concrete structures. The cooler 3D structures are difficult to discern as they have cooled to an even temperature and blend into the background.

The VRSG image below to the right shows the same area in the afternoon, after the sun has heated the asphalt road and the stucco sides of the buildings. The vegetation is cooler than the concrete and asphalt structures heated by the sun. This example illustrates thermal inversion.



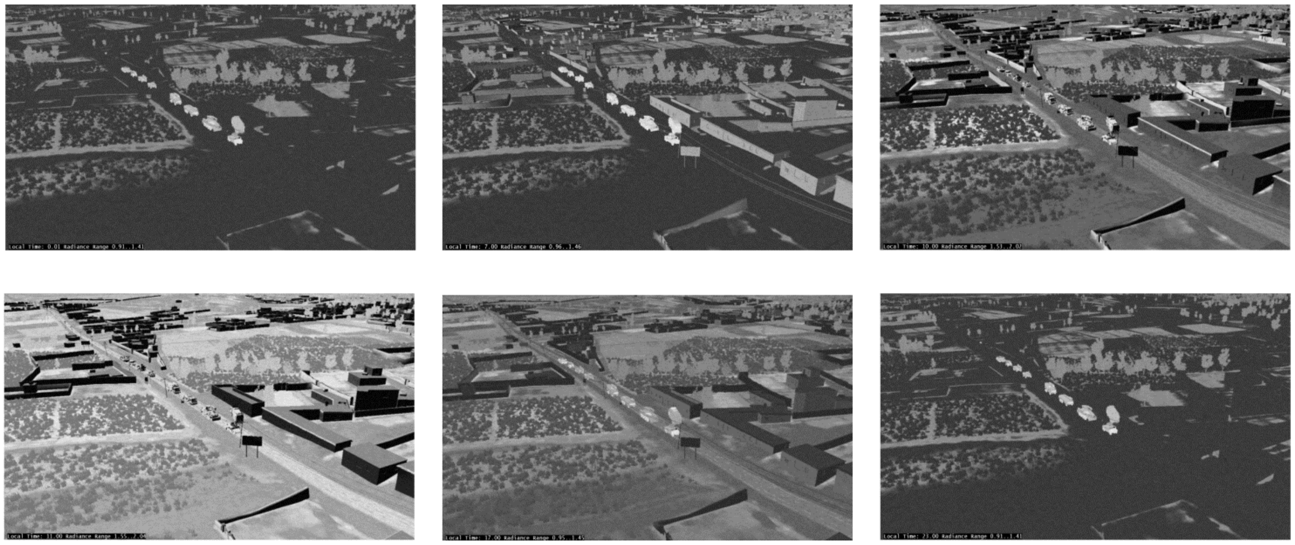
VRSG physics-based sensor image at midnight.



VRSG phyics-based sensor image of the same area in the early afternoon.

Using a physics-based sensor model or the material-based model involves providing VRSG with material code attribution of the scene elements. For geospecific imagery, this means providing a material palette that maps visual-spectrum RGB color values to material codes from TSC's material library. This takes place in the IR Setup utility described below. For geotypical content such as moving models, buildings, 3D vegetation models, and so on, you can choose among three mechanisms for providing material attribution:

- By using the "-material" command in the cultural feature file (vrsg.clt) or ModelMap.ini file. This command assigns the given material to all polygons in the model.
- By providing a table to map textures to materials. You use the file vrsg.json to map texture file names to material codes from the TSC library.
- By building material codes into an OpenFlight model before converting it to MVRsimulation's HPX model format.



A 24-hour cycle IR view from VRSG, showing the thermal progression starting at 12:00 midnight.

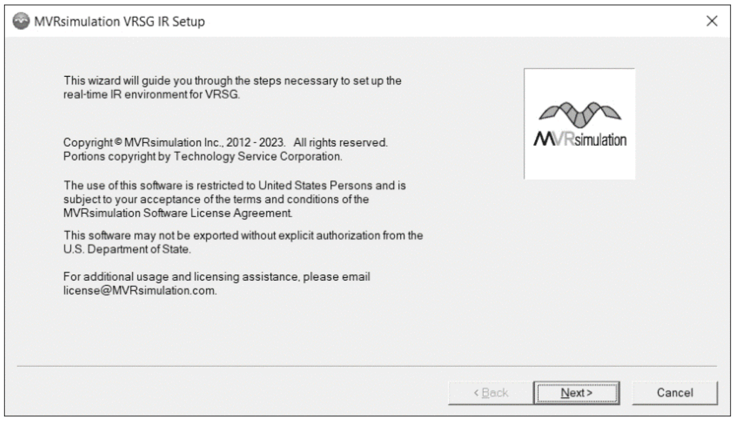
Setting up physics-based IR configuration

For US domestic users who can use physics-based IR, the IRSetup utility is the easiest way to produce a json file containing a radiance profile. This section describes how to use the IRSetup utility. The process of creating a radiance profile manually is described in a later section.

You use the IR Setup utility to describe the sensor's spectral response within its waveband of interest, train the material classifier, and specify the environmental characteristics that influence the appearance of an IR scene.

To start the IR Setup utility:

1. From the Windows Start menu, choose All Programs > MVRsimulation > VRSG Setup for Physics-Based IR. The welcome screen will be displayed:


MVRsimulation VRSG IR Setup

This wizard will guide you through the steps necessary to set up the real-time IR environment for VRSG.

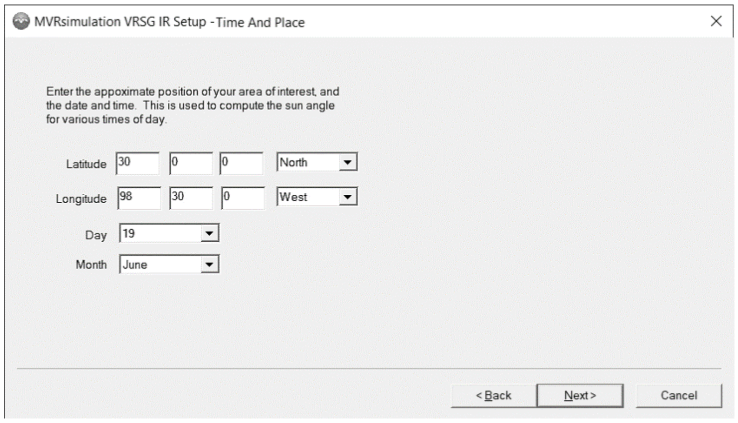
Copyright © MVRsimulation Inc., 2012 - 2023. All rights reserved. Portions copyright by Technology Service Corporation.

The use of this software is restricted to United States Persons and is subject to your acceptance of the terms and conditions of the MVRsimulation Software License Agreement.

This software may not be exported without explicit authorization from the U.S. Department of State.

For additional usage and licensing assistance, please email license@MVRsimulation.com.

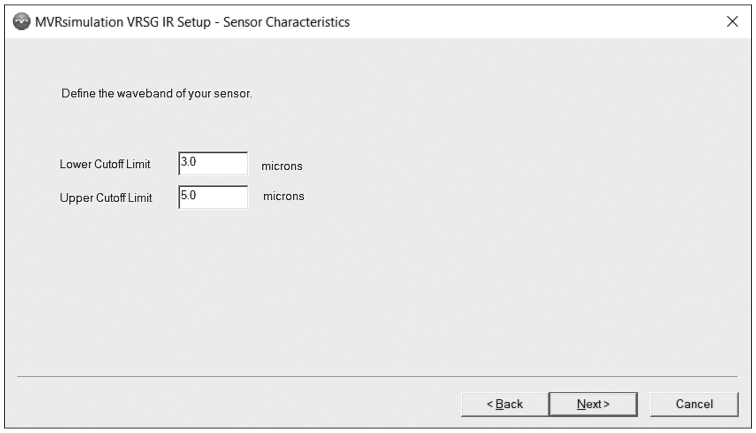
2. Click Next to move to the Time and Place screen:


MVRsimulation VRSG IR Setup -Time And Place

Enter the appoximate position of your area of interest, and the date and time. This is used to compute the sun angle for various times of day.

Latitude	30	0	0	North
Longitude	98	30	0	West
Day	19			
Month	June			

3. Click Next to move to the Sensor Characteristics screen:

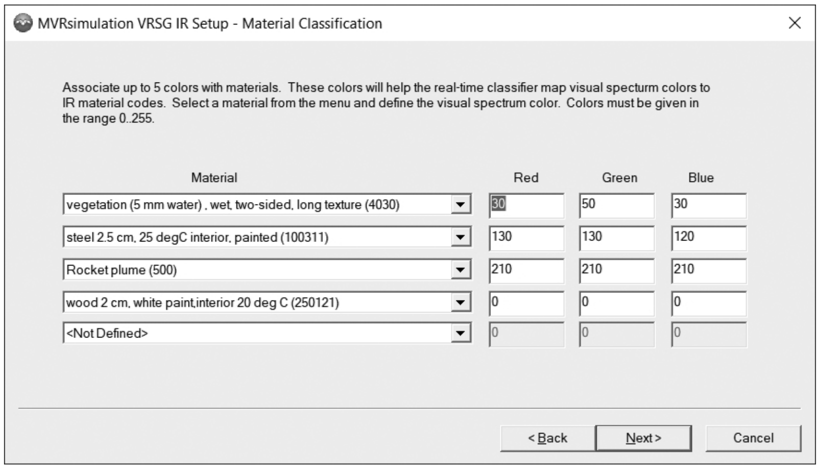

MVRsimulation VRSG IR Setup - Sensor Characteristics

Define the waveband of your sensor.

Lower Cutoff Limit	3.0	microns
Upper Cutoff Limit	5.0	microns

Enter the lower and upper cutoff wavelength limits of the sensor you are simulating. The sensor is defined by its wavelength limits, given in microns. For example, a mid-wavelength IR sensor might use a lower cutoff limit of 3.0 microns and an upper cutoff limit of 5.0 microns. A long-wavelength sensor might use a lower cutoff of 8.0 micros and an upper cutoff of 12 microns.

4. After entering the sensor waveband, click Next to advance to the Material Classification screen.



Associate up to 5 colors with materials. These colors will help the real-time classifier map visual specturm colors to IR material codes. Select a material from the menu and define the visual spectrum color. Colors must be given in the range 0..255.

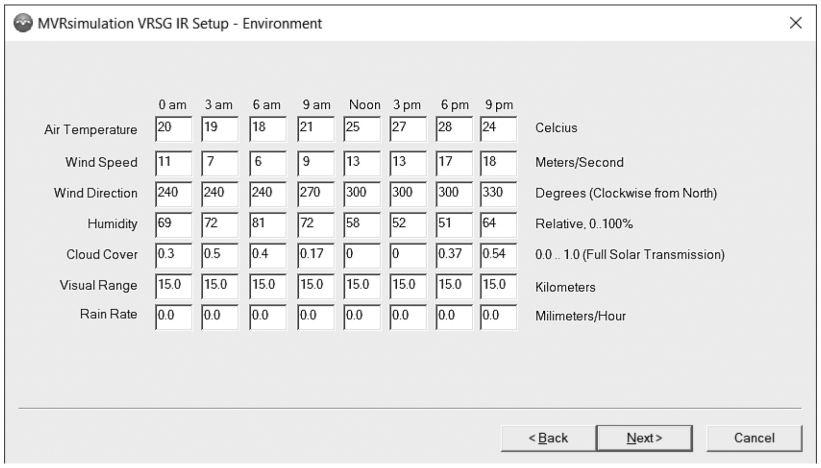
Material	Red	Green	Blue
vegetation (5 mm water) , wet, two-sided, long texture (4030)	30	50	30
steel 2.5 cm, 25 degC interior, painted (100311)	130	130	120
Rocket plume (500)	210	210	210
wood 2 cm, white paint,interior 20 deg C (250121)	0	0	0
<Not Defined>	0	0	0

5. You use this screen to train the real-time material classifier. Enter classification information for up to 5 materials by selecting their material code from a row on the left side of the screen, and then providing the corresponding RGB values. During runtime, colors from the geospecific imagery are used to infer characteristics of materials necessary for the IR physics model.

   Different material classification palettes are likely needed for different geographic areas, different seasons within those areas, and different types of sensors used to collect the imagery. MVRsimulation recommends taking screen captures of representative materials you want to classify from your visual database. Then, open a selection of those screen captures in an image editor such as Corel Paint Shop Pro or Adobe Photoshop and determine the average RGB values of these materials. During runtime, colors from the visual database rarely exactly match colors from your material palette. The resulting material is generally a blend of all materials in your palette, those materials having a closer proximity in color space receiving a greater weighting in the interpolated result.

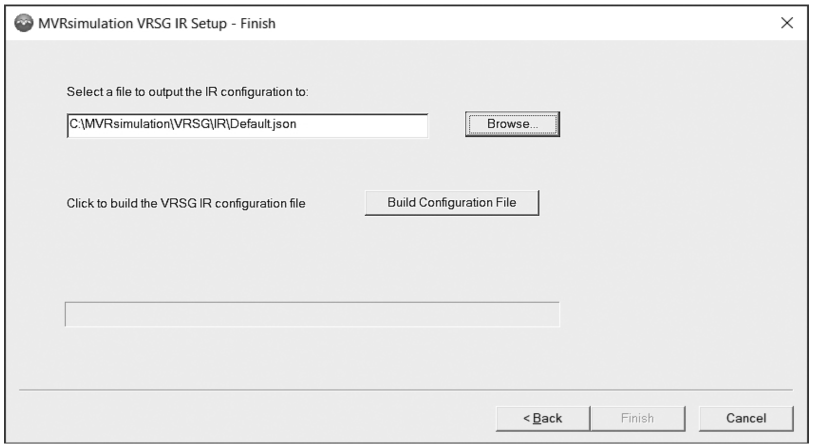
   The state-of-the-art in automatic material classification has its limitations. When differing materials are similar in color, the classification process will have trouble distinguishing between them. Furthermore, misclassification is common using auto-classification. For example, a dark shadow cast by a tall mountain may make the terrain in the shadow appear similar in color to asphalt, which is dark by nature. These limitations are inherent in any auto-classification system, whether it is an off-line system or real-time such as this one. For large geospecific databases, there is rarely corresponding material attribution information collected, so in such cases auto-classification is the only practical option.

6. After completing the material classification palette, click Next to advance to the Environment screen.



	0 am	3 am	6 am	9 am	Noon	3 pm	6 pm	9 pm	
Air Temperature	20	19	18	21	25	27	28	24	Celcius
Wind Speed	11	7	6	9	13	13	17	18	Meters/Second
Wind Direction	240	240	240	270	300	300	300	330	Degrees (Clockwise from North)
Humidity	69	72	81	72	58	52	51	64	Relative, 0..100%
Cloud Cover	0.3	0.5	0.4	0.17	0	0	0.37	0.54	0.0 .. 1.0 (Full Solar Transmission)
Visual Range	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	Kilometers
Rain Rate	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Milimeters/Hour

7. On this screen, enter the factors of the environment that the physical model requires for its computation. When you have finished, click Next to advance to the final screen.



Select a file to output the IR configuration to:

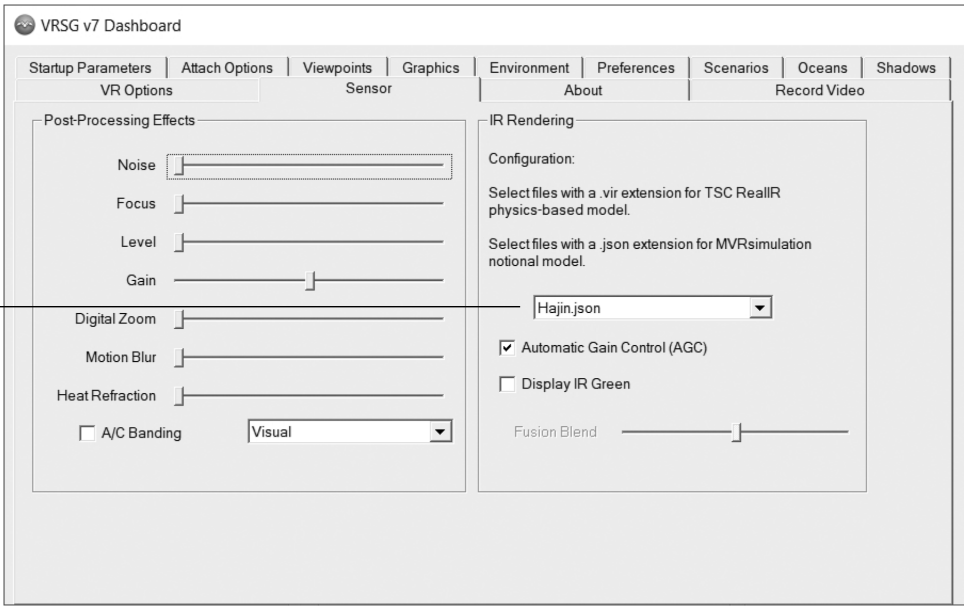
C:\MVRsimulation\VRSG\IR\Default.json

Click to build the VRSG IR configuration file

8. Click the Browse button to create or select a filename for the IR configuration. The produced configuration file will have the extension ".json" and must reside in VRSG's IR subdirectory. VRSG will not find the IR configuration file if it is not created in this directory.
9. Once you have created or selected the directory path and name of the output configuration file, click Build Configuration File to build the IR configuration file. After a few seconds, notification message will appear stating that the process is complete and you can click Finish to exit from the IR configuration setup wizard.

When you next start VRSG and click Enable Sensor Modes on the Dashboard's Startup Parameters tab, you should see your new IR configuration listed in the IR Configuration menu on the Sensor tab:

IR configuration file for nominal or physics-based IR simulation.



Select files with a .vir extension for TSC RealIR physics-based model.

Select files with a .json extension for MVRsimulation notional model.

Hajin.json

[✓] Automatic Gain Control (AGC)

Manually creating a radiance profile

The IRSetup utility is under ITAR control, which prevents its use in non-US domestic environments. The radiance profile can be created manually, which is useful for populating with notional data or data from a third-party physics-based model. This section describes the format of the file to enable you to produce one manually.

The file uses the popular JSON format, which is human-readable ASCII. The JSON format is a highly structured format. There are tools available on the web to assist in syntax checking of JSON files. They can also be created/edited in a simple text editor such as Notepad.

MVRsimulation recommends using a working file as a starting point, such as the Afghan.json sample provided in the \MVRsimulation\VRSG\IR subdirectory. The description below explains the contents of the Afghan.json file, which should enable you to create your own customized radiance profile.

The first datum in the file is a version identifier. At the time of this writing, the current version is 1. The version entry in the file will appear as follows, indicating version 1:

```
"version":1,
```

The second entry is an array of materials, as indicated by this line:

```
"materials": [
```

What follows is a set of lines for each material that describe an example entry for a given material:

```
    {
      "code": 3111,
      "description": "sand (light) 20 cm, 10 degC interior rock
texture",
      "rgb": [209, 183, 126],
      "shade": [1.29,1.25,1.23,1.36,1.46,1.53,1.63,1.43 ],
      "inSun": [1.36,1.31,1.30,1.50,1.55,1.65,1.71,1.55 ]
    },
```

The "code" line indicates the integer material code. Material codes can be assigned to textures, polygons, or to entire models. How material codes get assigned to content is described in the following section.

The "description" line is used for documentation purposes only, it does not affect the simulated sensor scene.

The "rgb" line indicates the color values for a material given in red, green, and blue intensities. The valid range of intensities is 0 .. 255. This color value is used to auto classify geospecific terrain imagery. You should only include the RGB line for materials that you want to use with geospecific imagery. This should be a very bounded set of materials, we recommend using no more than 5 materials with the RGB line provided.

The "shade" line indicates the material radiance values for a 24 hour diurnal cycle, of the material not affected by sunlight. You can supply a single value if desire a constant radiance across the diurnal cycle. In this example, 8 values are provided, corresponding to 3 hour increments (e.g. midnight, 3:00 am, 6:00 am, etc.). If you wanted to provide hourly radiance samples, you would enter 24 values.

The "inSun" line is similar to the "shade" line, but the material radiances correspond to radiance values associated to the material receiving sunlight. Both the shade and inSun lines should contain the same number of samples.

The ability to manually create a radiance profile gives you complete control over how a given material presents in the sensor view as a function of time-of-day. Effects such as thermal crossover between two materials can be expressed in this data. Furthermore, the physics-based output from IRSetup can be modified to address the preferences of subject matter experts.

Material attribution of non-geospecific content

The material classification process described in the previous section automatically generates material attribution from visual spectrum colors. This is used exclusively for the geospecific imagery of the terrain. For other content, such as moving models, vegetation models, buildings, etc., there are other mechanisms of applying material attribution to this content. These methods allow you to retrofit material attribution to existing MVRsimulation-provided content or user-developed content.

Per-model assignment of material codes

An entire model may be assigned a material code using the "-material=N" command. This command may be applied to models in ModelMap.ini or static models in a cultural feature (vrsg.clt) file. This command assigns the given material code to all polygons in the model. This is the coarsest, most global method of material attribution, and is useful for vegetation models, culture, or other models where the entire model can be represented by a single material type for IR rendering purposes. Consider this example from an entry in the ModelMap.ini file:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.camo.hpy –material=100132
```

In the above example, the material "100132 – Steel 10cm, two sided, painted" would be assigned to all polygons in the M1A2.M2.US.camo.hpy model. See the end of this chapter for a complete list of possible material codes.

For models not given an explicit material code assignment with "-material=N", they will be assigned a default material based on how the model is used. Models listed in the ModelMap.ini file will be assigned a material code of "100111 - steel 2.5 cm, 30 deg."

Models listed in the cultural features file (.clt) will be assigned a default material of "201001 - concrete 10 cm, 10 deg", unless they are qualified as a vehicle using the "-vehicle" command. If the model is qualified as a vehicle, the default material assigned will be "100111 - steel 2.5 cm, 30 deg."

Per-texture assignment of material codes

For a more refined assignment of material codes, use the per-texture attribution. If a material is bound to a particular texture, and that model was given a per-model material attribution, the per-texture assignment will override the per-model assignment. Any textures in the model not given an assignment will use the per-model attribution. Thus you can use per-model attribution for quick and coarse attribution, and per-texture assignment as needed to achieve the desired level of attribution fidelity.

To establish a per-texture attribution, create an ASCII file named "vrsg.irm." This file can reside in any directory in VRSG's search path. For example, you might have a unique vrsg.irm file per terrain. VRSG supports multiple vrsg.irm files; you can store them in subdirectories of models to which the vrsg.irm files apply. Bear in mind that search path order is important; VRSG loads the first directory it finds. For this reason, consider putting a default vrsg.irm file in the \MVRsimulation\VRSG\Textures directory, or editing the default vrsg.irm file already in that directory. You can check VRSG's info log to verify the correct location of vrsg.irm was loaded.

The format of vrsg.irm is simple. Each entry contains the mapping of one texture to a material code on one line, for as many lines as required.

Consider the following example vrsg.irm entries:

```
door_large_05               400121
juniper-tree-branch         4019
road_dirt                   201001
roads                       201001
roof-tile-brown03           109020
roof-tile-dark-brown-02     501020
wall_sand_06                201001
```

You can find another example IRM file at:
\MVRsimulation\VRSG\Terrain\Afghanistan\vrsg.irm.

Per-polygon assignment of material codes

Per-polygon binding of material codes is the strongest binding possible. Per-polygon attribution of material codes will override per-texture attribution, which will in turn override per-model attribution. Most models in the VRSG model library do not have per-polygon attribution, so either per-texture or per-model attribution will be required. MVRsimulation's HPY models are not user-editable, so you will not be able to retrofit per-polygon attribution to HPY models. Per-polygon attribution is most useful for user-developed models using the

Presagis Creator model editing tool. See the “Assigning polygon material codes” section of this chapter for details about assigning material codes to models.

Dynamic moving model signatures

VRSG physics-based and material-based IR users benefit from dynamic moving model signatures. These dynamic signatures are not purely physics-based, rather they are physics-correlated and they provide additional stimulus for training benefit. Dynamic moving model signatures are caused by activities of the vehicle that influence its IR signature. Examples are a running engine warming a portion of the vehicle, the wheels or tracks becoming heated due to motion, or the gun barrel becoming hot as the result of firing a round.

The simulation network generally does not provide enough detailed information for these dynamic effects to be run though the physics model. In addition, the simulations themselves most likely do not model the vehicle to the fidelity required to run surface temperatures though a physics model. Together VRSG and its physics-based IR component work around these limitations by allowing additional display intensity to be blended into the underlying surface temperatures provided by the physics model. Thus the physics model is used to compute the base display intensity of a moving model as a function of its material type and environmental conditions, and the dynamic effects are extra display intensity, which is added to the signature.

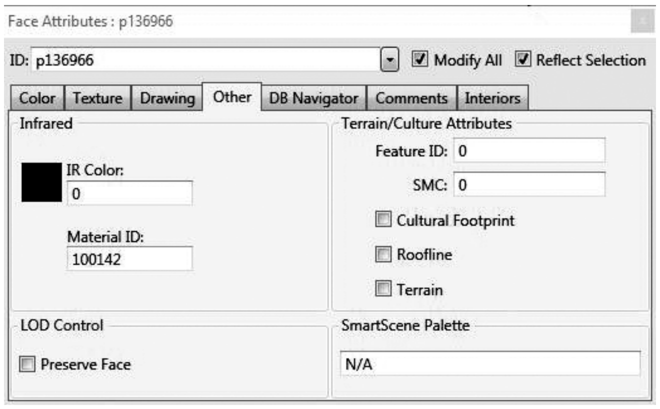
For VRSG to take advantage of the dynamic moving model signature capability, the models converted to MVRsimulation’s format from either FBX or OpenFlight format must be made compliant before conversion. Model compliance consists of polygon material assignment, hot spot and non-hot spot textures, and metadata built into the model’s scene graph. The following examples show how to make an FBX or OpenFlight model compliant with the VRSG/physics-based IR component architecture. You can edit the model in your modeling tool and then convert it to MVRsimulation’s HPX runtime model format using the conversion utilities provided with the VRSG installation. See the chapter, “Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats” for more information about how to prepare FBX and OpenFlight models for use in VRSG.

Assigning polygon material codes

If a material assignment can be made at the per-texture level, then per-polygon assignment is not necessary. In some cases you might need to assign material codes to polygons to override the per-texture assignment.

To assign the IR material code to a face in Presagis Creator:

1. Select the set of faces to apply the material code to.
2. Choose the Appearance tab > Modify Attributes. The Face Attributes dialog box appears.
3. Click the Other tab. In the Infrared section, enter the material code to assign to the set of selected faces. In the example below, the selected face is assigned the IR material ID of 100142, which corresponds to steel.


Face Attributes : p136966

ID: p136966 ☑ Modify All ☑ Reflect Selection

Color | Texture | Drawing | Other | DB Navigator | Comments | Interiors

Infrared

IR Color: 0

Material ID: 100142

Terrain/Culture Attributes

Feature ID: 0

SMC: 0

☐ Cultural Footprint

☐ Roofline

☐ Terrain

LOD Control

☐ Preserve Face

SmartScene Palette

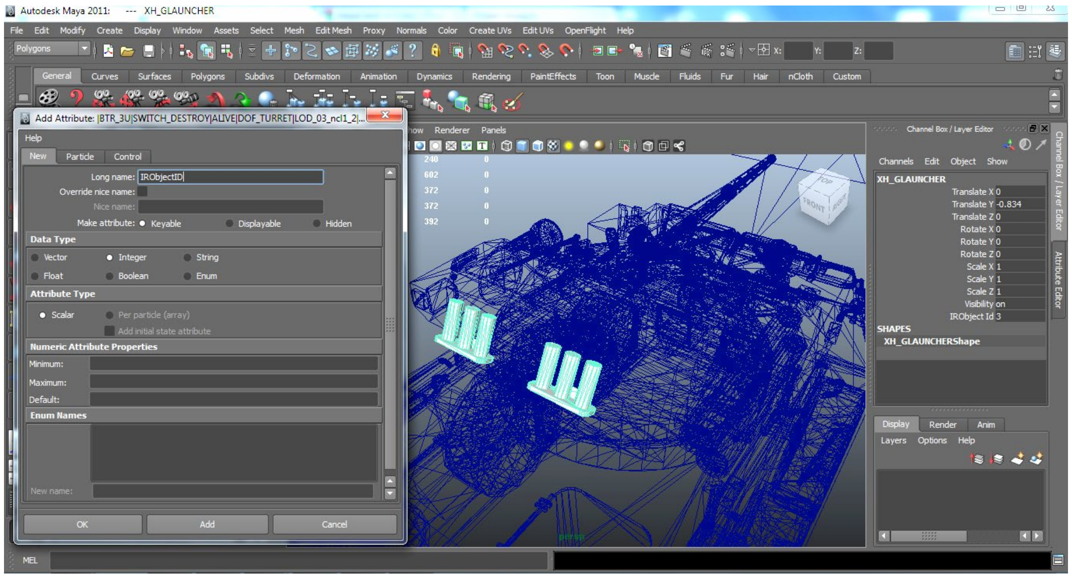
N/A

Creating IR animations

Your model's scene graph should group the faces of the model into sections or nodes that can be addressed at runtime by VRSG as a logical unit. For example, you would place all polygons associated with the engine compartment under one node or object, all polygons associated with the gun barrel under a different node or object, and so on.

VRSG specific nodes for an FBX model are specified by adding prefixes to node names within the model editing software. You can add IR animations using an FBX Null node and giving its name the prefix IR_ANIMATION_. An IR_ANIMATION_ node requires a custom property that defines IRObjectId of the node. Child nodes inherit this IRObjectId property. The IRObjectId property is used as an identifier when manipulating hot spots on the model. Additionally, an IR Object Id custom property can be assigned per FBX Mesh Node. The per-Mesh property overrides any previous IR Object Id settings. The syntax/type of this custom property must be exact: IRObjectId / integer.

The following image shows the IRObjectId setting in the modeling tool Autodesk Maya:


Autodesk Maya 2011: --- XH_GLAUNCHER

Add Attribute: |BTR_3U|SWITCH_DESTROY|ALIVE|DOF_TURRET|LOD_03_ncl1_2|...

Long name: IRObjectID

Override nice name: ☐

Make attribute: ● Keyable

Data Type: ● Integer

Attribute Type: ● Scalar

XH_GLAUNCHER	
Translate X	0
Translate Y	-0.834
Translate Z	0
Rotate X	0
Rotate Y	0
Rotate Z	0
Scale X	1
Scale Y	1
Scale Z	1
Visibility	on
IRObject Id	3

SHAPES

XH_GLAUNCHERShape

In an OpenFlight model, once you have completed the grouping, you can add comments to the object that enable VRSG to treat the set of geometry as an IR animation.

To add comments to the object in Creator for an OpenFlight model:

1. Select the object to assign metadata to. The object should be an engine, tracks, or gun barrel group.
2. Choose Attributes > Modify Attributes. The Object Attributes dialog box appears.
3. Click the Comments tab.
4. In the Comments field, type “IRAnimation engine” if this object is for an engine compartment, “IRAnimation track” if this object represents the tracks or wheels, or “IRAnimation gun” if this object represents the vehicle’s gun.

The example above shows the set of polygons of an M1’s engine compartment being assigned to group with the IRAnimation metatags. You can have multiple objects in the model with the same IRAnimation metatags associated with them if that is more convenient for the modeling process. For example, you could put the left and right tracks of a vehicle under two separate objects, both attributed with the “IRAnimation track” metatags.

VRSG will create implicit IR animations for articulated parts, so if your model employs articulated parts, you may not need to define an explicit IR animation.

The following rules apply to implicit IR animations:

- Geometry under a DOF node that is attributed as a Primary Gun (part 4416) will be considered a gun for IR hot spot blending purposes.
- Geometry under a RotatingAnimation or a TextureAnimation will be considered as wheels/tracks for IR hot spot blending purposes.
- Geometry that is not under any of the above will be considered as the engine compartment for IR hot spot blending purposes.

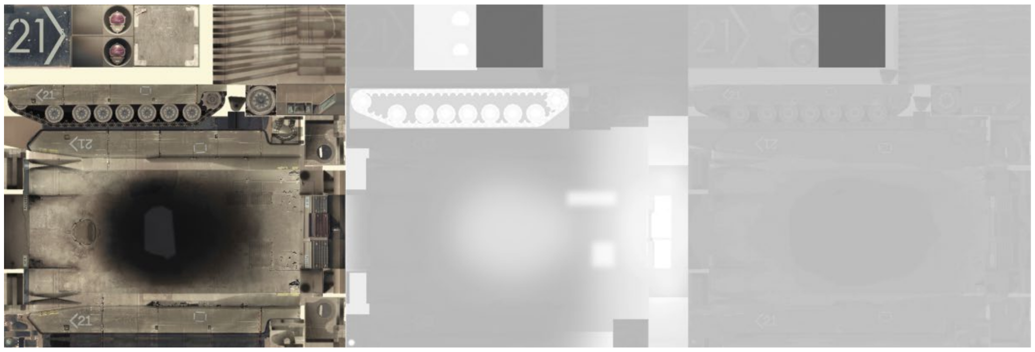
Assigning IR textures

The final stage in model compliance is assigning IR textures to the model. You do not need a modeling tool for this task; you simply provide alternate textures for the model that can be used in IR mode by VRSG. VRSG uses the following convention for using IR textures for a model:

If a visual spectrum texture is named *texture_name.ext*, VRSG will load *texture_name*_IR.*ext* to use in IR mode. This texture should contain the hot spots for the dynamic signature portions of the vehicle. VRSG will also attempt to load *texture_name*_IR_NHS.*ext* for the same texture. This texture should be the IR signature of the vehicle without hot spots (that is, the vehicle in a cold state). If no visual spectrum (non-hot spot) texture is provided, the vehicle will always be displayed with hot spots, regardless of simulation events. If both the hot spot and non-hot spot textures are provided, VRSG will display a blending between these two textures as a function of time and simulation events. For example, the engine compartment will gradually heat up when the engine is started, and slowly cool down when the engine is turned off.

Whenever possible, you should use models with model-specific textures rather than generic textures; doing so it makes it easier to identify which portions of the vehicle the texture map applies to. A model-specific texture is typically a mosaic of photographs or artwork of different sides of the vehicle. With model specific textures, the places to add hotspots can be readily identified.

The following three images show an M1 model texture in the visual spectrum, IR with hot spots, and IR without hot spots:



Material codes

The following table provides the set of available material codes supported by MVRsimulation's physics-based IR. These materials are to be assigned to polygons, vertices, or texels in the visual database and moving models.

Material code	Description
270	270 K calibration
275	275 K calibration
280	280 K calibration
285	285 K calibration
290	290 K calibration
293	293 K calibration
295	295 K calibration
300	300 K calibration
305	305 K calibration
310	310 K calibration
313	313 K calibration
315	315 K calibration
320	320 K calibration
330	330 K calibration
340	340 K calibration

Material code	Description (continued)
350	350 K calibration
500	Rocket plume
502	Turbofan plume
2000	water at 0 C
2010	water at 10 C
2011	snow/ice (-10 C)
2012	snow/ice (-25 C)
2019	water at 10 C with sea texture
3110	sand (light) 20 cm, 10 degC interior
3111	sand (light) 20 cm, 10 degC interior rock texture
3112	sand (light) 20 cm, 10 degC short texture
3115	sand (light) 20 cm, 10 degC interior
4012	leaf (5 mm water) two-sided
4019	tree foliage (5 mm water) two-sided, short texture
4028	vegetation (5 mm water) two-sided, synthetic texture
4029	vegetation (5 mm water) two-sided, long texture
4030	vegetation (5 mm water) , wet, two-sided, long texture
4110	grass 20 cm, 10 degC interior
4119	grass 20 cm, 10 degC interior, synthetic texture
4120	grass 20 cm, 10 degC interior, wet, synthetic texture
4210	grass, dry
4212	grass, wet
4214	grass, dry, shadow
4301	Tree canopy
7110	asphalt 20 cm, 10 degC interior
7111	asphalt 20 cm, 10 degC interior, road texture

Material code	**Description (continued)**
20100	explosion, 1,000 C
100111	steel 2.5 cm, 30 degC interior, painted
100112	steel 2.5 cm, two-sided, painted
100121	steel 5 cm, 30 degC interior, painted
100122	steel 5 cm, two-sided, painted
100131	steel 10 cm, 30 degC interior, painted
100132	steel 10 cm, two-sided, painted
100141	steel 1 cm, 30 degC interior, painted
100142	steel 1 cm, two-sided, painted
100151	steel 1 cm, 60 degC interior, painted
100153	steel 1 cm, 50 degC interior, painted
100155	steel 1 cm, 40 degC interior, painted
100211	steel 2.5 cm, 20 degC interior, painted
100221	steel 5 cm, 20 degC interior, painted
100231	steel 10 cm, 20 degC interior, painted
100241	steel 1 cm, 20 degC interior, painted
100311	steel 2.5 cm, 25 degC interior, painted
100321	steel 5 cm, 25 degC interior, painted
100331	steel 10 cm, 25 degC interior, painted
100341	steel 1 cm, 25 degC interior, painted
100441	steel 1 cm, 10 degC interior, painted
101111	steel 2.5 cm, 40 degC interior, painted
101121	steel 5 cm, 40 degC interior, painted
101131	steel 10 cm, 30 degC interior, painted
101141	steel 1 cm, 40 degC interior, painted
101211	steel 2.5 cm, 60 degC interior, painted

Material code	**Description (continued)**
101221	steel 5 cm, 60 degC interior, painted
101231	steel 10 cm, 60 degC interior, painted
101241	steel 1 cm, 60 degC interior, painted
101242	steel 1 cm, 40 degC interior, painted
102141	steel 1 cm, painted, 200 degC interior flow
102142	steel 1 cm, painted, 250 degC interior flow
102143	steel 1 cm, painted, 100 degC interior flow
102151	steel 1 cm, painted, 200 degC interior flow, engine covers
103111	steel 2.5 cm, 20 degC interior, painted ,engine cover sides
103211	steel 2.5 cm, oil-filled, 20 degC interior, painted
104000	steel door with air space (composite)
105000	steel 1 cm, two-sided, spectral emissivity
108100	steel painted, fixed 100 C, exhaust port
109020	steel painted, fixed 20 C
109035	steel painted, fixed 35 C
109040	steel painted, fixed 40 C
109050	steel painted, fixed 50 C
109060	steel painted, fixed 60 C
109080	steel painted, fixed 80 C
109100	steel painted, fixed 100 C
109200	steel painted, fixed 200 C
109500	steel painted, fixed 500 C
110111	aluminum 1.0 cm, 10 degC interior
110112	aluminum 1.0 cm, two-sided
110121	aluminum 1.0 cm, 20 degC interior
110131	aluminum 1.0cm, 30 degC interior (vesna)

Material code	Description (continued)
110161	aluminum 1.0 cm, 100 degC interior
111111	aluminum,painted, 1.0 cm, 10 degC interior
111121	aluminum,painted, 1.0 cm, 30 degC interior
111122	aluminum, painted, 1.0 cm, two-sided
111151	aluminum,painted, 1.0 cm, 50 degC interior
111201	aluminum,painted, 1.0 cm, 100 degC interior
112201	aluminum,painted, 1.0 cm, 200 degC interior flow
113005	aluminum 1.0 cm, two-sided leading edge
113010	aluminum 1.0 cm, two-sided leading edge
113105	aluminum 1.0 cm, 10 degC interior leading edge
113110	aluminum 1.0 cm, 10 degC interior leading edge
138111	steel 2.5 cm, 20 degC interior, painted, exhaust port sides
201001	concrete 10 cm, 10 degC interior
201002	concrete 10 cm, two-sided
201005	concrete, white paint, 10 cm, 10 degC interior
201011	concrete 30 cm, 10 degC interior
201012	concrete 30 cm, two-sided
201021	concrete 20 cm, 10 degC interior
201101	cinder block
250121	wood 2 cm, white paint,interior 20 deg C
251112	roof tile 2.5 cm, interior 20 degC
300112	brick 10 cm cold chimney (2-sided)
300313	brick 10 cm for hot chimney (50 degC interior, 1 m/sec flow)
301112	roof tile 2.5 cm, interior 20 degC
400121	wood 2 cm, white paint,interior 20 deg C
400152	wood 20 cm, two-sided, bark exterior

Material code	Description (continued)
400321	wood 10 cm, black paint, interior 20 deg C
500101	window glass, 0.5 cm, 20 degC interior
501020	Rubber, 1 cm, 20 C interior
501045	Rubber, 1 cm, 45 C interior
502020	Canvas, 2mm, 20 cm interior
600111	Plexiglas, 1 cm, 20 C interior

CHAPTER 8

Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats

Delivered with VRSG are utilities for converting models in FBX and OpenFlight format to MVRsimulation's model format. These tools enable you to both use your existing FBX or FLT formatted models in VRSG and to create your own FBX or OpenFlight models with your own modeling tools for use in VRSG. VRSG is also delivered with utilities for converting OpenFlight terrain to MVRsimulation's round-earth VRSG terrain format (MDS). These tools provide the means of both reusing legacy OpenFlight terrain in VRSG and the ability to create new terrain in OpenFlight format with tools such as Presagis Creator or Terra Vista.

This chapter describes how to convert FBX and OpenFlight models and OpenFlight terrain databases to MVRsimulation's runtime formats so that you can visualize them in VRSG scenes. This chapter also includes some considerations for CityEngine models exported to FBX or FLT format. Finally, the chapter describes recommendations for storing your site's own models within \VRSG\Models subdirectories.

Converting the FBX model format

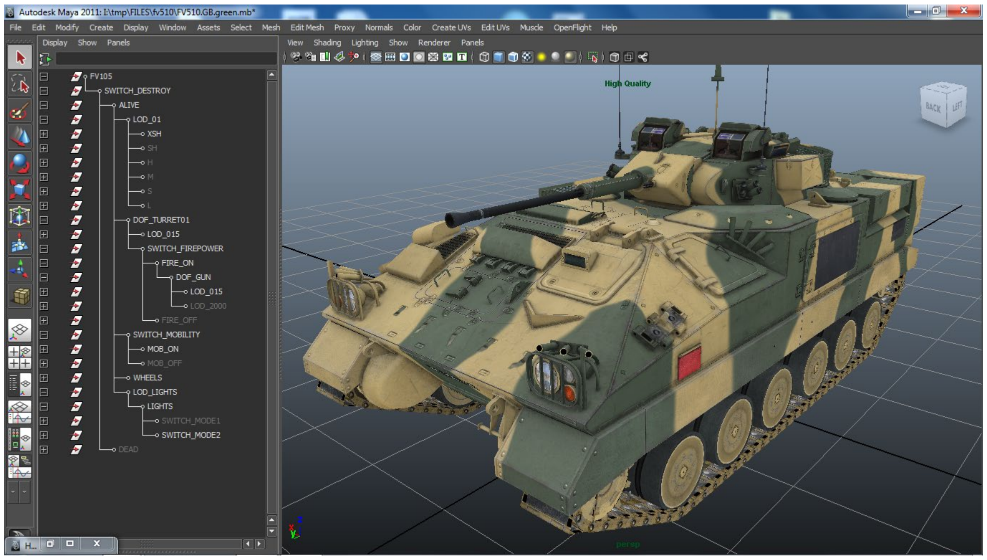
MVRsimulation's Fbx2Hpx utility converts a FBX-formatted model to MVRsimulation's HPX model format for rendering in Virtual Reality Scene Generator (VRSG). MVRsimulation's support of Autodesk's popular FBX model format enables modelers to use a number of third-party modeling tools including Autodesk 3ds Max, Autodesk Maya, and Luxology Modo, to create models for use within VRSG.

Similar to MVRsimulation's OpenFlight to HPX converter described later in this chapter, Fbx2Hpx is a command-line utility that imports a FBX model and exports it as an HPX model. A number of MVRsimulation-specific custom attributes (node names and properties) are supported in the FBX model converter to include node types that are needed for models used in visual simulation. A model of interest can be created in any modeling tool that enables you to create polygonal geometry, use custom attributes, and export the model to FBX format. For processing an FBX model, Fbx2Hpx accepts diffuse, normal, and ambient occlusion (AO) texture maps. The utility is installed with VRSG in the directory \MVRsimulation\Common\Util\Fbx2Hpx. The utility can be moved and used anywhere on your computer hard drive.

The Fbx2Hpx utility is built with Microsoft's Visual Studio 2015. To properly execute the tool the Visual Studio 2015 runtime needs to be installed on your machine. (You can obtain the Visual Studio 2015 runtime installer at www.microsoft.com/en-us/download/details.aspx?id=40784.)

Delivered with the Fbx2Hpx utility are several files:

- The utility itself (Fbx2Hpx.exe).
- The file mvr_control_rig.fbx for converting character models. This control rig defines the transformations for each bone in MVRsimulation's standard rig. (A sample character model in FBX format is available in the \ExampleModels subdirectory.)
- MVRsimulation’s Software License Agreement.
- A directory of example models (vehicle, human character, and building) in FBX and in Autodesk Maya or 3ds Max formats, which you can examine to become familiar with setting up a model in your modeling tool for conversion to HPX format: a vehicle model (KVP-Towed.RU.green), a character model (human-example) and a building model in (EBBL-Kleine-Brogel-AFB-052).
- An image, MVRsimulation-FBX-Model-Conversion-Example.jpg, which contains screenshots of the sample model KVP-Towed.RU.green in Maya and in MVRsimulation’s Model Viewer.
- A readme file.



MVRsimulation's FV510 model, in AutoDesk's Maya modeling tool, prior to exporting it in FBX format for subsequent conversion to MVRsimulation's model format. A hierarchy of MVRsimulation custom attributes is shown on the left.

Using the fbx2hpx utility

To use the Fbx2Hpx.exe utility:

1. Open a command-line window.
2. Type Fbx2Hpx.exe in a command-line window, followed by the name of the model you want to convert, with or without any other parameters, using the syntax of the following form:

```
Fbx2hpx [-f] [-q] [-i] <model-name.fbx> [<target-directory>]
```

When you type Fbx2Hpx.exe without any parameters, a description of the parameters appears on the screen. The parameters are:

-f to overwrite an existing HPX output file. (The default is to not overwrite a previous HPX file of the same name.)

-q to suppress any error messages.

-i to ignore any errors and proceed with converting the model to HPX format.

-b to allow the conversion of binary FBX files.

-n *x* to limit the mesh node maximum vertex count to x. Mesh nodes with more than *x* vertices are split into multiple mesh nodes. The default value is 2000 vertices.

-a to display copyright notices and terms of usage.

The resulting HPX model will be given the same name as the FPX model (with an HPX file extension) and will be saved in the same directory as the FBX model unless you specify a target directory on the command line.

If you have a large number of models to convert at once, you can convert all FBX models in a given directory by providing the input directory and (optionally a target directory).

For example:

```
Fbx2Hpx.exe d:\ models  d:\tempc
```

This example converts all the FBX models in the directory d:\models and outputs the HPX models to the target directory d:\temp.A number of MVRsimulation-specific custom attributes (node names and properties) are supported in the FBX model converter to include node types that are needed for models used in visual simulation. Such attributes include DOF, LOD, rotation animation, collision mesh, and so on. You can add these custom node names and properties to KfbxNull nodes as needed.

To ensure a smooth conversion, place the textures in the same directory as the FBX model file. If you want, you can place the textures in a subdirectory called \Textures.

Note: Use of the FBX utility is tied to an MVRsimulation product license, which means that you can run the utility on any machine that has a valid MVRsimulation license and active maintenance.

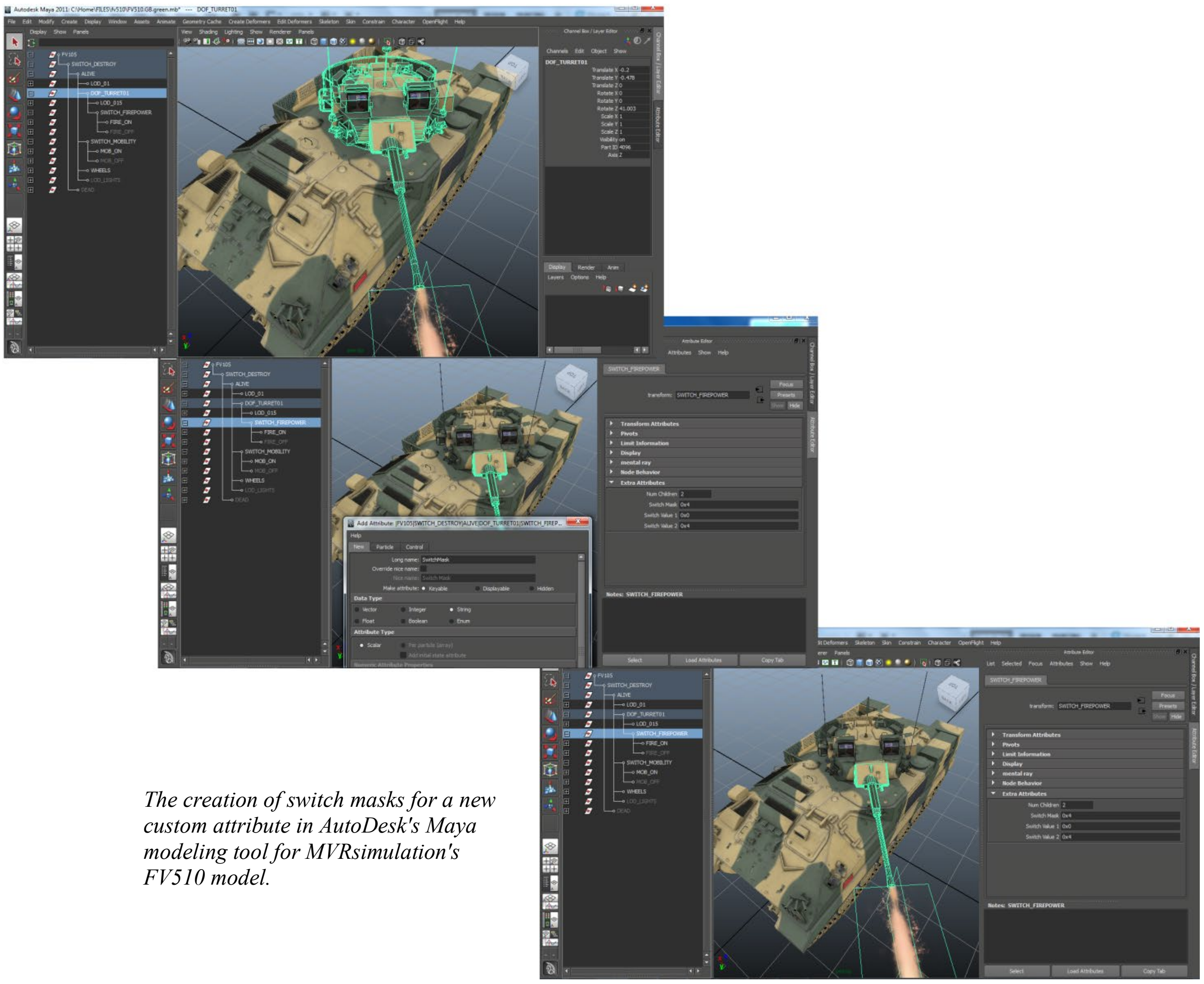
Creating an FBX model for real-time use in VRSG

To create and customize an FBX model for use in VRSG:

1. In your modeling tool, create or import a model's 3D geometry, and apply the materials and textures.
2. Organize a DIS-compatible hierarchy with dummy groups (LOD, DOF, and so on).
3. Specify the node type for each group by its prefix, such as "LOD_<any name>" or "DOF_<any name>."
4. Add the specific custom properties of the chosen node type: "Num Children", "Switch Mask" and so on. See the description below for the custom properties required for each node type.
5. Export the model to FBX file format.

6. Convert the FBX file to MVRsimulation's HPX format by using the Fbx2Hpx utility.

After you convert the model to HPX format, you can inspect the model in MVRsimulation's Model Viewer before using it in VRSG.


Autodesk Maya 2011: C:\Home\FILES\fv510\FV510.GB.green.mb* --- DOF_TURRET01
File Edit Modify Create Display Window Assets Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character OpenFlight Help
Channel Box / Layer Editor
Attribute Editor
Display Render Anim
Select
Load Attributes
Copy Tab

```
FV105
  SWITCH_DESTROY
    ALIVE
      LOD_01
      DOF_TURRET01
        LOD_015
        SWITCH_FIREPOWER
          FIRE_ON
          FIRE_OFF
      SWITCH_MOBILITY
        MOB_ON
        MOB_OFF
      WHEELS
      LOD_LIGHTS
    DEAD
```

```
DOF_TURRET01
Translate X -0.2
Translate Y -0.478
Translate Z 0
Rotate X 0
Rotate Y 0
Rotate Z 41.003
Scale X 1
Scale Y 1
Scale Z 1
Visibility on
Part ID 4096
Axis Z
```

```
Add Attribute: |FV105|SWITCH_DESTROY|ALIVE|DOF_TURRET01|SWITCH_FIREP...
Long name: SwitchMask
Nice name: Switch Mask
Make attribute: ● Keyable
Data Type: ● String
Attribute Type: ● Scalar
```

```
transform: SWITCH_FIREPOWER
Extra Attributes
Num Children 2
Switch Mask 0x4
Switch Value 1 0x0
Switch Value 2 0x4
Notes: SWITCH_FIREPOWER
```

The creation of switch masks for a new custom attribute in AutoDesk's Maya modeling tool for MVRsimulation's FV510 model.

Description of custom node types

A number of VRSG-specific custom attributes (node names and properties) are supported in the FBX model converter, including node types that are needed for models used in visual simulation. You can add these custom node names and properties to KfbxNull nodes as needed.

Switch node

Governs which child node is currently being rendered. A switch node has multiple child nodes; usually only one child node is rendered at a time. All switch nodes require the prefix "SWITCH_" in the node's name.

Custom properties:

- NumChildren - An integer representing the number of child nodes attached to the switch node.
- SwitchMask - A string which represents the hexadecimal mask value used when evaluating the validity of the child nodes' switch values. This is usually the bitwise union of all child-node masks. Examples: "0x18" or "0xe00".
- SwitchValue# - A separate 'SwitchValue#' property is required for each child node of the switch. SwitchValue properties are numbered sequentially starting at 1. ("SwitchValue1", "SwitchValue2", ...) Each SwitchValue property contains a string representing the hexadecimal switch value. Example: "0x1000".

Degree Of Freedom (DOF) node

Allows its child nodes to be rotated around a specified axis. All DOF nodes require the prefix "DOF_" in the node's name.

Custom properties:

- PartID - An integer node that assigns an integer ID value to this DOF. To be compatible with the DIS standard, these values should be multiples of 32. PartIDs can also influence the IR Object Id of any child meshes. PartID values of 4416 and 4096 are treated as gun IR objects. PartID values of 4064 are treated as wheel IR objects. By default a child mesh of a DOF node is treated as a hull IR object. See the description of IRObjectId for further details.
- Axis - An enum node that indicates which primary axis the DOF rotates around. Values of 0, 1, and 2 are used to represent the X, Y, and Z axes, respectively. If a rotational or translational transformation modifies this node, this transformation is applied to the nodes child meshes.

Level of Detail (LOD) node

Determines which child mesh will be displayed, based on the distance between the eyepoint and the centroid of the LOD node's bounding sphere. The bounding sphere encompasses the LOD node and all of its child nodes. All LOD nodes require the prefix "LOD_" in the node's name.

Custom properties:

- MaxDistance - An integer representing the maximum distance (in meters) between the bounding sphere centroid and the eyepoint. This LOD node will not be visible if the eyepoint moves beyond this distance.
- MinDistance - An integer representing the minimum distance (in meters) between the bounding sphere centroid and the eyepoint. This LOD node will not be visible if the eyepoint moves closer than this distance.
- NumThresholds - An integer representing the number of child meshes of the LOD node.
- ThresholdsLevel# - An integer representing the switch-in distance (in meters) for each child node; a separate "ThresholdsLevel#" property is required for each child node. ThresholdsLevel properties are numbered sequentially starting at 0 ("ThresholdsLevel0", "ThresholdsLevel1", ...) and should be sorted in increasing order from minimum to maximum. Thus, child 1 will be visible when the eyepoint distance is between

ThresholdsLevel0 and ThresholdsLevel1. Beyond that distance, child 2 will become visible, and then child 3 will become visible, and so on. The model will disappear beyond MaxDistance.

Animation node

Cycles through its child nodes one at a time, over a specified duration. This node can be thought of as a flip-book animation. All Animation nodes require the prefix "ANIMATION_" added to the node's name.

Custom properties:

- Duration - A floating-point value that indicates the duration of the animation cycle, in seconds.
- Looping - A Boolean that indicates whether the animation is one-time animation or a looping animation.
- NumChildren - An integer indicating the number of child nodes of the Animation node.

Fire Animation node

Also defines a flip-book animation; one that displays all its child nodes over a specified duration. Fire Animation nodes are always one-time animations. All Fire Animation nodes require the prefix "FIRE_ANIMATION_" in the node's name.

Custom property:

- Duration - A floating-point value that indicates the animation duration in seconds.

Rotating Animation node

Governs rotation rates on objects that are either velocity - dependent or duration-dependent. For example, wheel meshes on vehicles would use this node. All Rotating Animation nodes require the prefix "ROTATING_ANIMATION_" in the node's name.

Custom properties:

- VelocityDependent - A Boolean that indicates if the rate of animation of this node should be velocity-dependent. If the node is velocity dependent, the Radius property is used to govern the rate of rotation. If the node isn't velocity dependent, the AnimationDuration property is used to govern rate of rotation.
- Radius - A floating-point value that indicates the radius of the object, in meters. This value is used to determine rotation rate based on velocity.
- AnimationDuration - A floating-point value used to indicate duration (in seconds) of non-velocity-dependent animations.
- Axis - An enum that indicates the principle axis of rotation. Values of 1, 2, and 3 are used to represent the X, Y, and Z axes, respectively.
- DirectionX/Y/Z - Three distinct properties that define rotation around an arbitrary axis. Used to have the mesh rotate around an axis other than a principle/primary axis.

IR Animation node

Assigns an IR object Id to all child nodes. All IR Animation nodes require the prefix "IR_ANIMATION_" in the node's name.

Custom property:

- IRObjectId - An integer value that indicates the IR object Id to be assigned to all of the node's children.

For more information about the creating IR animations for an FBX model, see the chapter, "Sensor Views and Physics-Based IR."

Texture Animation node

Defines a UV animation for all child nodes. Such nodes can be used to simulate the motion of the track texture on a tracked vehicle such as a tank. All Texture Animation nodes require the prefix "TEXTURE_ANIMATION_" in the node's name.

Custom properties:

- uvRadius - A floating-point number indicating the radius (in meters) of the object being animated. This value is used to determine the rotation rate based on the vehicle's velocity.
- uRate - A floating-point number controlling how fast the texture will animate in the u- direction (if at all).
- vRate - A floating-point number controlling how fast the texture will animate in the v-direction (if at all).

For both uRate and vRate, a value of 0 prevents the texture from animating in that direction. A nonzero value implies motion in that dimension, and it should be interpreted as having the units as "textures/meter."

For example, if the model uses a tiling tread texture that represents two meters of tread-length along the u parameter, then the uRate and vRate values should be 0.5 and 0, respectively. With these values, the texture would cycle once (along u) for every two meters of motion.

Light Point node

Defines one or multiple light points. Light points emit a colored light from a specific location. To define light points in an FBX model requires a minimum of two nodes. The first node is a LightGroup node. Child nodes of the LightGroup node represent the individual light points. The LightGroup node specifies the properties that are inherited by all child LightPoints. LightGroup nodes require the prefix "LIGHTGROUP_" added to the node's name. Light points require the prefix "LIGHTPOINT_" added to the node's name.

LightGroup custom properties:

- Period - A floating-point number indicating the length of the animation cycle in seconds.
- TimeOn - A floating-point number indicating the duty cycle for the light points in seconds. Blinking lights will have a TimeOn less than the Period.
- Diameter - A floating-point number indicating the real-world diameter of the light points in meters.
- MinSize - A floating-point number indicating the minimum size in pixels of the light points diameter.
- MaxDist - A floating-point number indicating the maximum distance in meters that the light points are visible.

- Az - A floating-point number indicating the number of degrees relative to North that light points are visible.
- El - A floating-point number indicating the number of degrees relative the ground plane that the light points are visible.
- VFov - A floating-point number defining the full-angle vertical field-of-view in degrees. For omni-directional lights a value of 180 degrees should be used.
- HFov - A floating-point number defining the full-angle horizontal field-of-view in degrees. For omni-directional lights a value of 180 degrees should be used.
- RotRate - A floating-point number indicating the rotation rate in degrees per seconds. For omni-directional light points a value of 0 should be used.
- Texture - A string containing the name of the custom texture to be used when rendering the child light points.
- SkipAttenFlags - Flags define a bit mask indicating special treatment of a light string. The value of flags is a logical OR of all desired features. The current set of features includes:
  0x1 - do not perform angular attenuation on the top edge of a light's fov
  0x2 - do not perform angular attenuation on the bottom edge of a light's fov
  0x4 - do not perform angular attenuation on the right edge of a light's fov
  0x8 - do not perform angular attenuation on the left edge of a light's fov

  Each SkipAttenFlags property contains a string representing the hexadecimal value; for example: "0x3".

LightPoint nodes are Mesh nodes that define a single light point vertex. A light point vertex consists of position (X, Y, Z), phase (P) and intensity (A) and color (R, G, B). X, Y, Z are defined by the Mesh's control points. Phase (P) indicates the phase delay in seconds of the current light point. Both phase (P) and intensity (A) are stored in the Mesh's UV layer. R, G, B are stored in the Diffuse property of the Mesh's material layer. There are no custom properties for a LightPoint node. For more information about creating FBX light points, see the chapter "MVRsimulation Model Format."

External Reference node

Defines an external model file that will be loaded into the FBX model's scene graph. The node's native attributes scale, rotation, and translation are used to transform the external model. External reference nodes require the prefix "XREF_" added to the node's name.

Custom properties:

- File - A string containing the external reference model's filename.
- SaveLink - A Boolean which determines if the model is an external reference (true) or if the model's geometry is integrated into this model (false).
- Appearance - A string that represents the hexadecimal appearance value that will be used to render the external reference model.

Collision Mesh node

Defines a collision geometry node and is primarily used with rigged (character) models. A Collision Mesh node is not rendered but is used during intersection testing. All KFbxMesh nodes that contain collision geometry require the prefix "COLLISION_" in the node's name.

Description of custom properties

The following custom properties are added to native FBX node types to add VRSG-specific functionality.

Mesh node

Custom properties:

- Shader Override – An integer value that declares what shader will be applied to the node's geometry. This property can be applied to textured mesh nodes. For example, the value '4' is currently used to define when alpha test should be applied in the pixel shader when rendering the mesh. Valid values are:
  0: No shader override is applied to the geometry.
  4: Use shader-based alpha testing on the texture applied to this geometry.
  8: Use shader-based alpha testing and texture animation on this geometry.
- IRObjectId - An integer value that indicates the IR object Id to be assigned to all of the node's children.
- Material - An integer value that indicates the IR material classification of this node's geometry.
- NoDeferRender - A Boolean value that forces the node's geometry into the opaque (non-distance sorted) pass.
- PassThroughMesh - A Boolean value that marks the geometry for use in mixed-reality applications. Results in the mesh node being used to identify a location in the scene where real-world surface pixels will be rendered, replacing the virtual-world pixels.

Null node

Custom properties:

- groundPlaneZ - Defines the ground clamping offset from the model's origin that will be used when this model is clamped to the terrain. This property should be assigned to the first node of a model.

Multi-textured mesh nodes

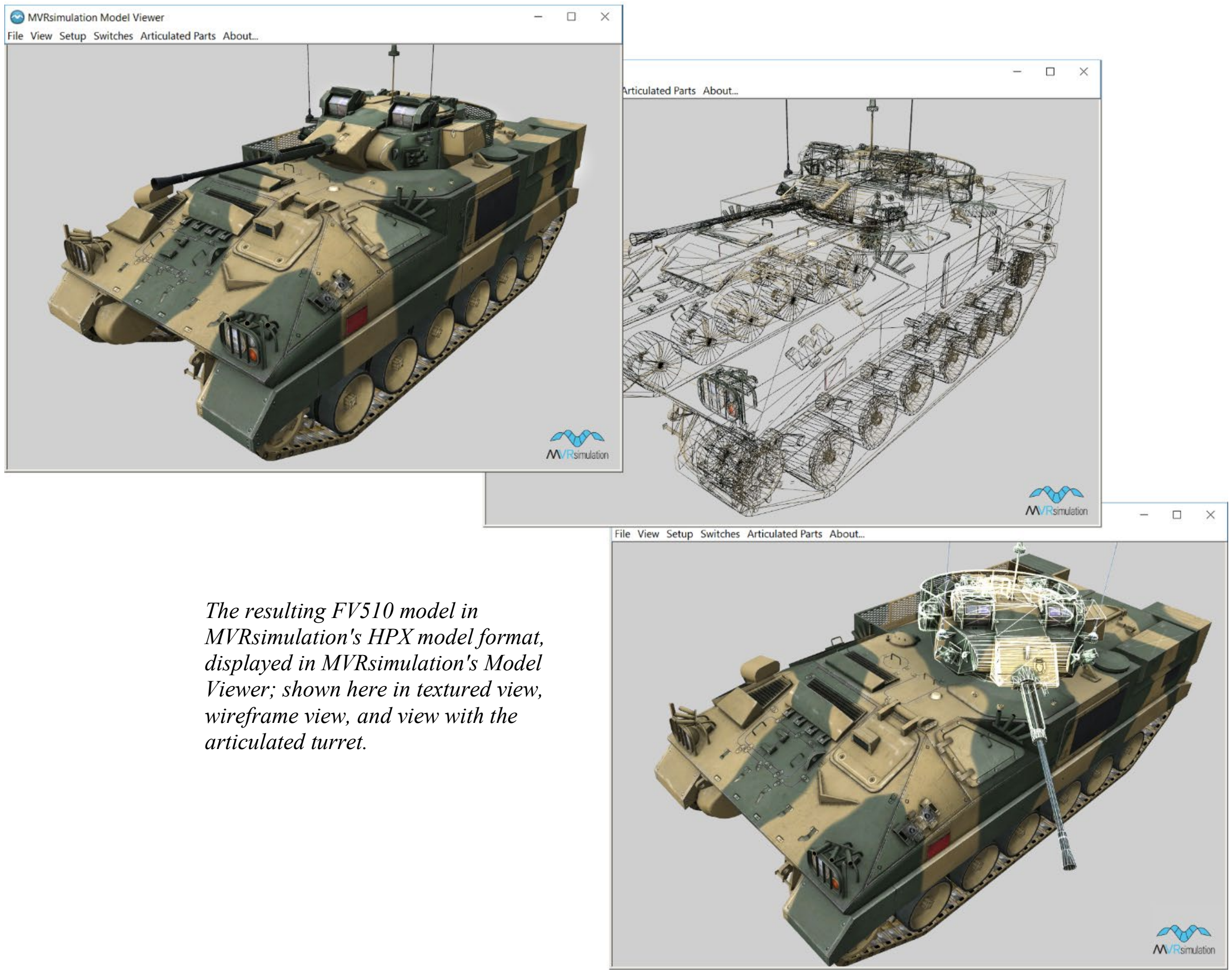
The tool supports Light Map and Ambient Occlusion textured geometry. These textures are identified via "_LM" and "_AO" appended to the end of their filename. The textures should be of the same size/resolution and share the same filename (<filename>_LM.rgb & <filename>_AO.rgb). These textures share a second set of texture (UV) coordinates applied to the geometry node.

The Light Map and Ambient Occlusion textures can be referenced to the mesh node by two methods:

- The Light Map and Ambient Occlusion map share the same filename as the diffuse texture applied to the mesh node. For example, if the diffuse texture is named "door.rgb" then the Light Map and Ambient Occlusion texture are named accordingly "door_LM.rgb" and "door_AO.rgb." The converter will look for _LM and _AO textures appended to the diffuse texture filename. If these textures are found to exist, the geometry will use multi-texturing.

- The Ambient Occlusion texture can be referenced via the "Ambient Color" layer attached to the mesh node. An associated Light Map texture will share the same name as the Ambient Occlusion texture but with _LM (instead of _AO) appended to the filename.

The tool also supports decal textured geometry. Decal texture support is currently implemented using FBX Layered Texturing. Both the diffuse and the decal texture associated with the mesh are expected to be added to the Layered Texture. The two textures are identified by the blending mode assigned to them. The diffuse texture uses the opaque blending mode (FbxTexture::eOver). The decal texture uses the additive blend mode (FbxTexture::eAdditive). A second set of UV coordinates is applied to the geometry node and used by the decal texture.

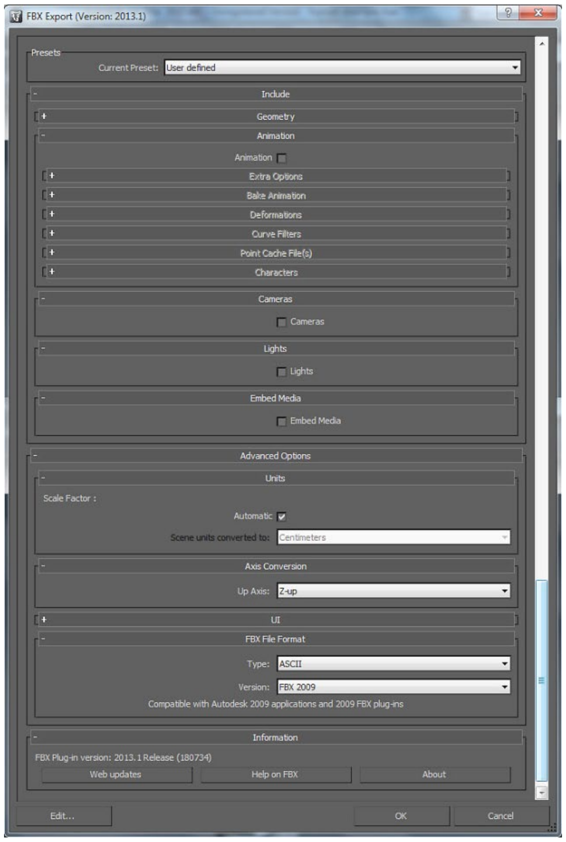

MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
Articulated Parts About...
File View Setup Switches Articulated Parts About...
MVRsimulation

The resulting FV510 model in MVRsimulation's HPX model format, displayed in MVRsimulation's Model Viewer; shown here in textured view, wireframe view, and view with the articulated turret.

Considerations for converting FBX character models

Upon request, MVRsimulation will provide customers under active maintenance a sample character model in .MAX, .FBX and .HPX formats so that you can examine the details, and use the example character in .MAX format as a template. Contact support@mvrsimulation.com to request this set of sample models. You can delete the geometry supplied in the template, (except for the collision geometry) and then link your geometry to the existing nodes, which are located under the LOD node. The character includes a LOD node (which can have any number of child nodes), a collision node (which is the same for all characters), and the skeleton (contains the bones). For information about the nodes, see the earlier part of this chapter. After you link your geometry to the existing nodes, you need to skin the meshes. Your characters can have any number of separate meshes. All meshes must be skinned so that the character can perform the animations successfully.

When you export your character in 3ds Max to FBX format, only select the Scale Factor – Automatic checkbox, and select the FBX file format options ASCII type and FBX 2009 version. Unselect the options Animation, Cameras, Lights, and Embed Media (if they are already selected). These options are shown in the following example of the FBX export dialog box:



FBX Export (Version: 2013.1)

Presets
Current Preset: User defined

Include
Geometry
Animation
Animation ☐
Extra Options
Bake Animation
Deformations
Curve Filters
Point Cache File(s)
Characters
Cameras
☐ Cameras
Lights
☐ Lights
Embed Media
☐ Embed Media

Advanced Options
Units
Scale Factor :
Automatic ✓
Scene units converted to: Centimeters
Axis Conversion
Up Axis: Z-up
UI
FBX File Format
Type: ASCII
Version: FBX 2009
Compatible with Autodesk 2009 applications and 2009 FBX plug-ins

Information
FBX Plug-in version: 2013.1 Release (180734)

The options to select and unselect in 3dsMax for exporting a character model to FBX format.

The T-pose is the bind pose (initial pose) for all MVRsimulation-format character models. All models should have similar proportions; they can vary with some small differences, but do not scale the bones. Scaling the bones will prevent a character from being able to use MVRsimulation’s standard character animations. Similarly, the origin of your character should remain the same origin as in the template (at the hips). If you are not using the MAX template file from MVRsimulation, be sure to use the file mvr_control_rig.fbx, which defines the transformations for each bone in MVRsimulation's standard rig and their offsets that all our animation files use. (The .MAX template file is an example of how to create a model using this default control rig.)

Considerations for FBX models output from CityEngine

If you intend to export building models from CityEngine in FBX format:

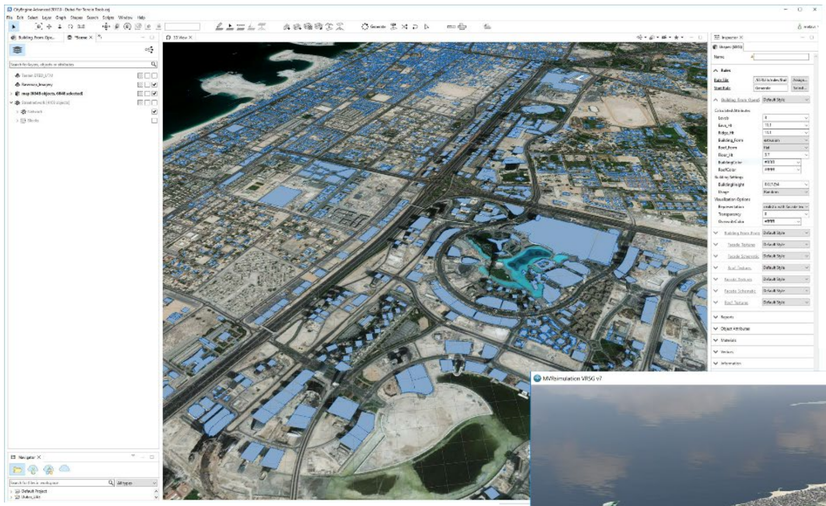
- You must have CityEngine Advanced, which supports exporting urban models in FBX format. (CityEngine Basic does not export models in FBX format.)
- Make sure that when you import source data into the CityEngine scene file, you specify the source data's coordinate system as projected UTM WGS1984, and then choose the corresponding UTM zone. Again, this elevation data must be in the UTM WGS84 projection. (VRSG natively supports geocentric WGS1984, but will support UTM WGS 1984 when the –utmModel flag accompanies the model's entry in the vrsg.clt cultural feature file.)
- Break up large city footprints into sections approximately 5 km x 5 km to ensure optimal performance in VRSG for rendering large models. (You can use Esri ArcGIS Pro to split up a building footprint shapefile into smaller models.) You would list all these models as separate entries in the terrain's vrsg.clt cultural feature file.
- If you encounter z-fighting in your exported CityEngine model, the cause is likely a building polygon layer that has overlapping building polygons and a building was extruded inside another building. The remedy is to clean up the footprint layer or edit the model.

When you export your urban model from CityEngine, be sure to set the following:

- Choose Advanced Settings > File Type, and then select Text from the drop-down menu to export the model in ASCII format.
- Choose Geometry Settings > Global Offset, and then select Center to center the origin of the model before exporting the model.
- Choose General Settings > Terrain Layers, and then select 'Do not export any terrain layers.'
- Choose AutoDesk FBX as the output format.

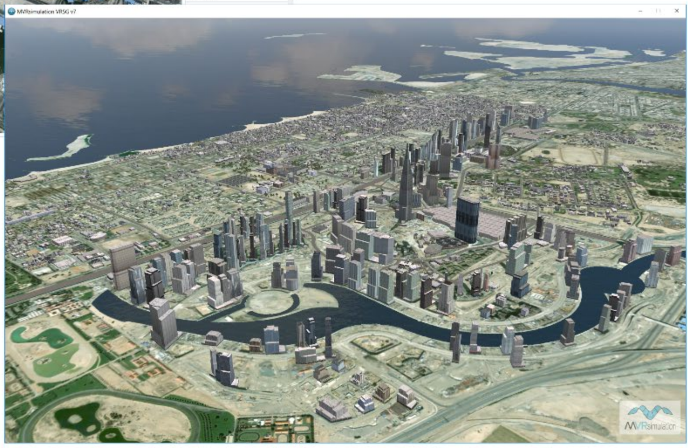
CityEngine does not export levels of detail (LODs) with a model. If you want your resulting model to have LODs when it is rendered in VRSG, you will need to add them in the appropriate model editing application (such as Autodesk Maya or 3ds Max) prior to converting the model to MVRsimulation's model format.

The following example illustrates the building footprints of Dubai in CityEngine prior to export as a large FBX model (left), and the resulting rendering in VRSG after the conversion and creation of a cultural feature file for the model.



Left: The city of Dubai's building footprints in CityEngine prior to exporting an urban model of 6,000 realistic 3D buildings, in FBX format for conversion to MVRsimulation's model format.

Below: The resulting city rendered in VRSG. The terrain was built with 60 cm imagery in MVRsimulation Terrain Tools.



Above: The city of Dubai's building footprints in CityEngine prior to exporting an urban model of 6,000 realistic 3D buildings, in FBX format for conversion to MVRsimulation's model format with the FBX conversion utility.

Right: The resulting city model rendered in VRSG, placed on 60 cm resolution terrain built with MVRsimulation Terrain Tools.

Below is the cultural feature file for placing the converted model of 600 buildings on the terrain:

```
! LL coordinates
N25 12 35.962 E055 16 39.895 6.00  -0.00 -0.00 0.00
Dubai_Buildings.hpy  -utmModel
```

Inspecting converted models in the Model Viewer

Once you have converted a 3D model from FBX format to MVRsimulation's model format, you can use MVRsimulation's Model Viewer to inspect the converted model to make sure that the geometry and textures are correctly assembled. See the chapter "Previewing Models, Effects, and Terrain" for more information about using Model Viewer.

Note: If your model appears dark in Model Viewer and no error message is displayed stating that textures are missing, check the vertex normals on your model. Incorrect vertex normals will cause a converted model to appear dark. You can resolve this issue by recalculating the vertex normals in your 3D modeling software. This issue can occur with models converted from a CityEngine export. It is a known issue that CityEngine sometimes exports models with vertex normals pointing downward instead of upward. CityEngine also applies a default material to exported models. This material can make the models appear darker in VRSG as well. In such a case, removing this default material from the model is recommended.

Converting OpenFlight formats

The OpenFlight format is a popular standard for entity models and terrain databases in the visual simulation industry. Although OpenFlight is a powerful on-disk representation of virtual worlds, it does not natively allow VRSG to exploit many of its performance enhancing features such as terrain paging, texture paging, and specialized hybrid culling algorithms. Therefore, you must convert OpenFlight models and terrain databases into the optimized MVRsimulation runtime formats before using them with VRSG. Delivered with VRSG are tools to perform these conversions.

MVRsimulation provides two utilities for converting OpenFlight models and databases to MVRsimulation runtime formats:

- *oflt2Hpx.exe* – converts an OpenFlight format dynamic model to the MVRsimulation HPX model format.
- *terrex-oflt2Mds.exe* – converts an OpenFlight format terrain database produced with Terra Vista, Synthetic Environment Core (SE Core), or TerraSim to MVRsimulation's VRSG round-earth terrain tiles format (MDS).

You run each utility from within a command-line window, as described in this chapter. These tools support version 13 of the OpenFlight API, and are 64-bit applications.

In addition to these conversion utilities is an MVRsimulation plugin for Presagis Creator. The plugin hpxPlugin.dll converts an OpenFlight format dynamic model to MVRsimulation's HPX model format. The plugin and conversion utilities are located in the directory: \MVRsimulation\Common\Util\OpenFlight.

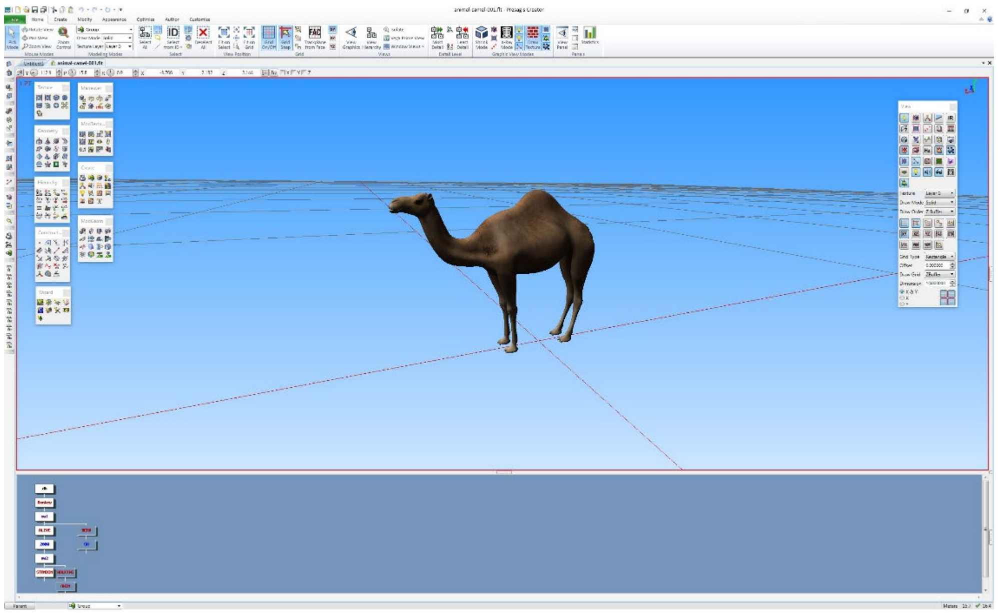
Converting OpenFlight models to HPX format

If you create your own 3D content, your models must be in OpenFlight format in order to use MVRsimulation's utility to convert them to HPX format so that you can use the models in VRSG. You can not only use Presagis Creator to create OpenFlight models, but Autodesk Maya and 3DS Max are two other modeling tools that have OpenFlight export capabilities. If you are using a modeling tool that does not have an OpenFlight export option, you can use tools such as Okino Polytrans or Right Hemisphere Deep Exploration to convert your models to OpenFlight format.

To convert an OpenFlight format model to an MVRsimulation HPX-formatted model, you can use one of the following methods:

- If you are using Presagis Creator, you can use MVRsimulation's hpxPlugin.dll converter from within Creator.
- If you have OpenFlight models from some other source, you can use MVRsimulation's oflt2Hpx.exe converter from the command line.

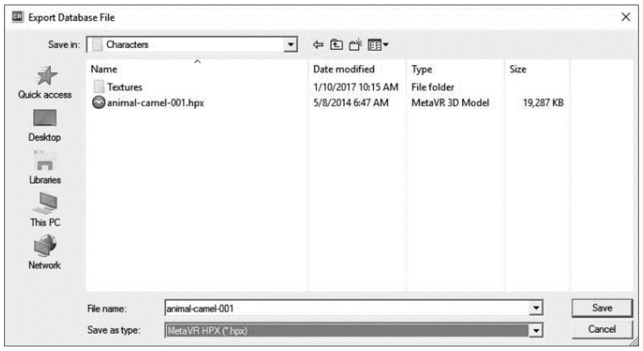
If you have a large number of models to convert at once, you should consider using the oflt2Hpx.exe command-line converter as a faster way of completing the conversion. By creating a batch file (.bat), you can list each individual invocation of oflt2hpx and convert several models at once.



Using the hpx plugin from within Creator

The plug-in hpxPlugin.dll converts an OpenFlight-formatted dynamic model to an MVRsimulation HPX model format from within Creator. Once you install this plugin in the \Plugins directory of Creator and then restart Creator, the MVRsimulation HPX runtime format will be available to you as an export format.

To export a model in HPX format from within Creator, choose File > Export from within Creator, and select MVRsimulation HPX as the export format, as shown below:



Export Database File

Save in: Characters

Name	Date modified	Type	Size
Textures	1/10/2017 10:15 AM	File folder	
animal-camel-001.hpx	5/8/2014 6:47 AM	MetaVR 3D Model	19,287 KB

File name: animal-camel-001

Save as type: MetaVR HPX (*.hpx)

Note: A current limitation of the Creator plugin is its inability to traverse any external references in the OpenFlight hierarchy. Therefore, if your dynamic model contains external references, you must use the standalone command-line converter, oflt2hpx.exe, to have the external references included in the output.

Using the oflt2Hpx utility

The oflt2Hpx utility outputs the resulting model as an HPX file. You run the oflt2Hpx utility in a command-line window, from the directory that contains the OpenFlight model file you want to convert.

To run oflt2Hpx:

1. Open a command-line window.
2. Change to the directory that contains the OpenFlight file you want to convert.
3. At the command line, type a command using the following form:

```
oflt2Hpx mymodel.flt model2.hpx
```

In this example, the model mymodel.flt is converted to a file called model2.hpx, in the current directory. You can give the HPX file any name you want, with the .hpx extension.

If the OpenFlight model consists of any external references, the converter first attempts to locate the externally referenced file at the path called out by the external reference record that references the external file. If that fails, the oflt2Hpx utility attempts to locate the file in the current directory. If that fails, a warning message is displayed, and the utility ignores the externally referenced file.

After the conversion is complete, move the model2.hpx file to the \VRSG\Models\User\<*subdirectory*> where <*subdirectory*> is the category of model and move any texture files that are used by the model to the \VRSG\Models\User\<*subdirectory*>\Textures directory. You can find out which texture files are used by the model by opening the HPX file in a text editor. All referenced textures are listed at the top of the HPX file.

In addition to converting files individually, you can convert a whole directory of OpenFlight models to HPX format at once. To do so, run the oflt2Hpx utility without specifying any file names in the command line. The utility will proceed to convert every FLT model in the current directory to an HPX file.

By adding an entry for a converted model to the ModelMap.ini file you can use the model as a dynamic entity in VRSG. See the chapter "Configuring Models and Events" for information about mapping entity models to DIS enumerations.

To enhance the dynamic display of a moving model in VRSG, you can add metadata to an OpenFlight model before exporting it to HPX format. See the chapter "MVRsimulation 3D Model Format" for more information.

If a converted culture model is in UTM projection, it can be converted to UTM WGS1984 projection, by adding the command –utmModel to the model's entry in the terrain's vrsg.clt cultural feature file. (VRSG natively supports geocentric WGS1984, but supports UTM WGS 1984 when the –utmModel flag accompanies the model's entry in the .clt file.)

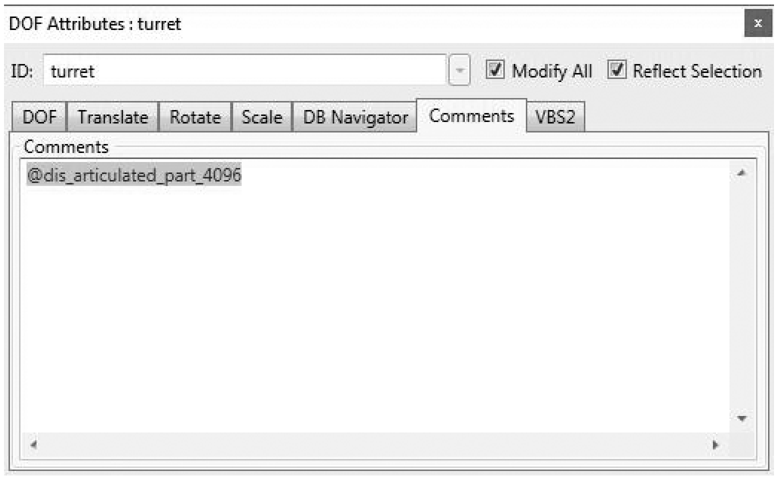
Shadow map and light map support

For models created in Presagis Creator, VRSG includes enhanced multi-texture support, including shadow maps and light maps. For faces with two textures applied, the second texture will be treated as a light map or a shadow map. In daytime viewing, the alpha channel of the second texture will be interpreted as a shadow map. In nighttime viewing, the RGB channels of the second texture will be interpreted as a light map.

World Reference Model support

The oflt2Hpx utility and the Creator plugin support a subset of the World Reference Model (WRM) Flight Entity Specification of the Distributed Interactive Simulation (DIS) standard.

WRM support is limited to assigning articulated part codes to Degree Of Freedom (DOF) nodes, and mapping Switch nodes to DIS appearance masks. To assign an articulated part code to a DOF, use the syntax: @dis articulated_part N in the comment field of the DOF node. N in this case designates the desired articulated part code. As an example, this comment would assign 4096 (main turret) to the selected DOF node.



ID: turret

[✓] Modify All [✓] Reflect Selection

@dis_articulated_part_4096

The full set of supported DIS enumerations for articulated parts can be found in IEEE 1278. This document is included in the VRSG installation in \MVRsimulation\VRSG\Models\SISO-REF-010-2023 Enumerations v31.pdf.

If a DOF node does not have a comment, it is assumed to be a turret (4096) if it rotates about the z-axis, or a gun (4416) if it rotates about the y-axis.

You can use WRM comments to map switch states to DIS appearance bits. To do so, add a comment to the switch node of the form:

```
@dis switch switch-type
```

Switch-type can be a symbolic name, or a hexadecimal number, that defines which bits of the appearance mask are relevant for the switch node. The DIS protocol allocates two bits in the appearance mask, allowing four damage states. As an example, the DIS protocol allocates bits 3 and 4 of the appearance mask for damage state, which corresponds to a mask of 0x18. Add a comment to the switch node using one of the following forms:

```
@dis switch 0x18
```

```
@dis switch damage
```

In general, the hexadecimal notation is recommended over symbolic names to comment switch nodes. If a switch node has no comment at all, it is assumed to be a damage switch. The following symbolic names are supported by the conversion tools:

WRM symbolic name	**Equivalent hexadecimal notation**
damage	0x18
mobility	0x2
fire_power	0x4
launcher	0x10000
hatch	0xe00
lights	0x7000
engine	0x400000
life_form_state	0xf0000

The children of the switch node correspond to all the possible states. In the case of a damage switch, 2 bits imply 4 possible states. Associated with each switch child is a value. VRSG performs a logical AND between the DIS appearance mask and the switch mask, and the switch child with the matching value is rendered. You need not supply switch children for all possible states. If VRSG does not find a matching state, the first child is used. If a switch node has 4 children, their left-to-right order implies their value. For example, with a damage switch of mask 0x18, the 4 switch children would have the values 0x00, 0x08, 0x10, 0x18.

You can specify the value of a switch child explicitly by entering:

```
@dis state N
```

in the comment field of the switch child. The value of N is typically a small integer ranging between zero and the number of switch children minus one. In the case of the damage switch with 4 possible children, the possible states could be 0, 1, 2, or 3.

The WRM standard allows for a switch child to support a range of values with syntax such as:

```
@dis state 0-3
```

MVRsimulation's OpenFlight conversion tools do not support this and require one distinct switch value per child.

Data to supply as per-texture comments

- @mvr:treeShader - tells VRSG to threshold the alpha channel, converting pixels below 50% opaque to fully transparent, and anything over 50% opaque to fully opaque. Useful for removing alpha-blending induced haloes around trees, and allows for more efficient rendering of such objects.
- @mvr:grassShader - similar to @mvr:treeShader, with an animated offset applied to the horizontal texture coordinates. This comment can be used to create the appearance of movement of grass textures, or tree foliage.

Data to supply as comments for light point nodes

- @mvr:DisplayInVisual - the light point is visible to the naked unaided eye.
- @mvr:DisplayInIR - the light point is visible in thermal infrared wavebands.
- @mvr:DisplayInNVG - the light point is visible in near-IR (NVG) wavebands.

Data to supply as comments for object nodes

@mvr:lightmap_id=N - indicate the geometry below the Object node corresponds to light map controls addressed to light map N, where N is an integer in the range of 0 to 15.

Data to supply as comments for group nodes

@mvr:runway - indicates that the geometry below should be rendered as a subface, with depth writes disabled and depth bias enabled. This comment is used to prevent z-fighting (flashing or flickering) between the coplanar or nearly coplanar terrain geometry beneath.

Z-fighting is common on runway models because they have many layered faces. The order of the hierarchy of the model is important for how an airfield is drawn in VRSG. The bottom layer should go on the very right (usually asphalt). The topmost layer, which is usually some level of striping placed on top of the model, should be at the very end of the hierarchy.

If you plan to compile a runway model in MVRsimulation's Terrain Tools, put the @mvr:runway comment in each geometry group layer on top of the bottom layer (asphalt group). If you are using MVRsimulation's Curved Runway tool, put this comment in the bottom layer (asphalt group).

See the *MVRsimulation Terrain Tools User's Guide* for more information about compiling a runway model into the terrain with Terrain Tools.

Inspecting converted models in the Model Viewer

Once you have converted a 3D model from OpenFlight format to MVRsimulation's model format, you can use MVRsimulation's Model Viewer to inspect the converted model to make sure that the geometry and textures are correctly assembled. See the chapter, "Previewing Models, Effects, and Terrain" for information about using Model Viewer to inspect your converted models.

Considerations for using models output from CityEngine

Both CityEngine Basic and Advanced have the option to export building models in Collada DAE format, which you can import into Presagis Creator for exporting building models in OpenFlight format.

If you intend to export building models from CityEngine:

- Make sure that when you import source data into the CityEngine scene file, you specify the source data's coordinate system as projected UTM WGS1984, and then choose the corresponding UTM zone. Again, this elevation data must be in the UTM WGS84 projection. (VRSG natively supports geocentric WGS1984, but will support UTM WGS 1984 when the –utmModel flag accompanies the model's entry in the vrsg.clt cultural feature file.)
- Break up large city footprints into sections approximately 5 km x 5 km to ensure optimal performance in VRSG for rendering large models. (You can use Esri ArcGIS Pro to split up a building footprint shapefile into smaller models.) You would list all these models as separate entries in the terrain's vrsg.clt cultural feature file.
- If you encounter z-fighting in your exported CityEngine model, the cause is likely a building polygon layer that has overlapping building polygons and a building was extruded inside another building. The remedy is to clean up the footprint layer or edit the model.

When you export your urban model from CityEngine, be sure to set the following:

- Choose Advanced Settings > File Type, and then select Text from the drop-down menu to export the model in ASCII format.
- Choose Geometry Settings > Global Offset, and then select Center to center the origin of the model before exporting the model.
- Choose General Settings > Terrain Layers, and then select 'Do not export any terrain layers.'
- Choose Collada DAE as the output format, if your modeling tool is Presagis Creator (you can choose a Collada version as well). You would import the DAE model into Presagis Creator to convert it to FLT format, and use MVRsimulation's HPX plugin for Creator.

CityEngine does not export levels of detail (LODs) with a model. If you want your resulting model to have LODs when it is rendered in VRSG, you will need to add them in Creator prior to converting the model to MVRsimulation's model format.

MVRsimulation has developed three command-line options for the oflt2Hpx.exe converter specifically for converting models that were made with CityEngine:

`-flatten` ignores group and object nodes, flattening the model hierarchy and clustering all geometry under LOD nodes.

`-randomLightMapIDs` applies a light map texture and randomly applies intensity to ensure all buildings are not lit with the same intensity.

`-utm=` unprojects the vertices of specified (large) previously UTM-projected model to ensure it sits without distortion on the underlying imagery.

The following example shows a subset of oflt2Hpx.exe commands using these options in a batch file for converting OpenFlight buildings originally created in CityEngine to MVRsimulation HPX model format:

```
.\oflt2hpx.exe Seoul_A4_0.flt -flatten -randomLightMapIDs
-utm=52,4169638.65771484,320474.161621093,187.331665039062
```

```
.\oflt2hpx.exe Seoul_A5_0.flt -flatten -randomLightMapIDs
-utm=52,4171813.80175781,325403.143554687,297.615661621093

.\oflt2hpx.exe Seoul_A6_0.flt -flatten -randomLightMapIDs
-utm=52,4171337.23046875,329892.025390625,231.33349609375

.\oflt2hpx.exe Seoul_B3_0.flt -flatten -randomLightMapIDs
-utm=52,4166672.34204101,316289.086914062,103.945922851562
```

VRSG real-time scene featuring a detail from the high-resolution geospecific virtual replica of Seoul, South Korea. The terrain features over 800,000 building structures extruded and textured in CityEngine and exported for conversion to MVRsimulation's model format with the FLT conversion utility. The terrain itself was built with MVRsimulation Terrain Tools.

Below is an excerpt from the cultural feature file for virtual Seoul, with the entry for placing the converted models of over 800,000 buildings on the terrain:

```
! UTM Meters in zone 52

! clt_version=2

320474.16162109375 4169638.65771484375 187.3316650390625 0.0 0.0 0.0
Seoul_A4_0.hpx

325403.1435546875 4171813.8017578125 297.61566162109375 0.0 0.0 0.0
Seoul_A5_0.hpx

329892.025390625 4171337.23046875 231.33349609375 0.0 0.0 0.0
Seoul_A6_0.hpx

316289.0869140625 4166672.342041015625 103.9459228515625 0.0 0.0 0.0
Seoul_B3_0.hpx
```

Inspecting converted models in the Model Viewer

Once you have converted a 3D model from FLT format to MVRsimulation’s model format, you can use MVRsimulation’s Model Viewer to inspect the converted model to make sure that the geometry and textures are correctly assembled. See the chapter "Previewing Models, Effects, and Terrain" for more information about using Model Viewer.

Note: If your model appears dark in Model Viewer and no error message appears stating that textures are missing, check the vertex normals on your model. Incorrect vertex normals will cause a converted model to appear dark. You can resolve this issue by recalculating the vertex normals in your 3D modeling software. This issue can occur with models converted from a CityEngine export. It is a known issue that CityEngine sometimes exports models with vertex normals pointing downward instead of upward. CityEngine also applies a default material to exported models. This material can make the models appear darker in VRSG as well. In such a case, removing this default material from the model is recommended.

Converting OpenFlight terrain databases to VRSG terrain tiles format

MVRsimulation's terrex-oflt2mds utility converts OpenFlight terrain databases, produced with Terra Vista or Synthetic Environment Core (SE CORE), to terrain tiles in VRSG round-earth terrain format (MDS). This conversion utility is located in the directory \MVRsimulation\Common\Util\OpenFlight.

As OpenFlight databases support geocentric coordinate systems, the resulting converted dataset of VRSG terrain tiles is geocentric (WGS84), and has all advantages associated with it with that format. The conversion workflow supports running multiple instances of the terrex-oflt2mds utility conversion utility concurrently, as described in full below. Each process can be restarted in the event of a power failure or any other disruption to the processing. Upon restart, the conversion process will pick up where it left off, and referenced by the tile by filename.

Using the terrex-oflt2mds utility

To convert an OpenFlight database to MVRsimulation's round-earth terrain tiles, you run the terrex-oflt2mds utility from the directory that contains the OpenFlight (FLT) database files you want to convert. By default, the utility looks for the presence of a master.flt file, which is the catalog of all the FLT tiles in the OpenFlight database; it references all the flightNxM.flt files. If this master.flt exists with a different name, you can specify it as a parameter. You could consolidate all FLT and texture files into a single directory and place the utility in this directory. Note that the converter requires all the DLL files located in C:\MVRsimulation\Common\Util\OpenFlight.

VRSG supports point feature instancing, which improves both terrain conversion time and VRSG runtime performance. This feature is especially useful for converting and rendering millions of tree models. The converter automatically decides whether to instance point features, based on usage patterns.

You run the terrex-oflt2mds.exe utility in a command-line window, from the directory that contains the OpenFlight database file(s) you want to convert.

To run terrex-oflt2mds:

1. Change to the directory that contains both the terrex-oflt2mds.exe utility and the OpenFlight file you want to convert.
2. Do one of the following:
   - Double-click the terrex-oflt2mds.exe utility.
   - Open a command-line window, and at the command line, type:

```
terrex-oflt2mds [-master <filename>]
                [-folder <output-directory >]
                [-flipdds]
                [-water <texture filename1> -water <texture
filename2> …]
```

   where:

   `-master <filename>` is the name of the master.flt file, if it is named something other than master.flt. (Without this parameter, the name master.flt is assumed.)

   `-folder <foldername>` is the target directory of the converted terrain tiles. (Without this parameter, the current directory is assumed to be the output directory.)

   `-flipdds` corrects databases that use flipped vertical orientation for DDS textures.

   `-water <texture filename>` cites one or more water textures that you want to be replaced with VRSG's water system.

The utility produces terrain tiles in the parent directory of the directory that contains the OpenFlight data. It also creates a \Textures subdirectory. VRSG will use the \Textures subdirectory at runtime for any terrain tiles that make references to external textures. Note the \Textures directory will contain only a small subset of the source textures, as the bulk of the textures will be compiled into the MDS terrain tiles.

If you move the resulting terrain tiles (.MDS) after the conversion, be sure to move the \Textures subdirectory with them; it must remain a subdirectory for VRSG to render the tiles correctly.

To speed up the terrain conversion, you can run multiple instances of the terrex-oflt2mds utility concurrently, by double-clicking terrex-oflt2mds.exe again (or multiple times) or running another instance of the utility in another command-line window. The number of instances is limited to the number of cores of your computer. (This means that if you have an 8 core processor, you can run 8 instances of the utility for an 8-fold speedup in the terrain conversion.) When a given terrex-oflt2mds instance encounters a terrain tile being built by another process, it skips that tile and moves on to build the next tile in the conversion queue.

Water

The OpenFlight converter that is delivered with VRSG 7 is the ability to specify one or more OpenFlight water textures that you want replaced during conversion with VRSG's water system. Use the command-line option -water followed by the texture name. For example:

```
-water water_green_otw_01_512.rgb
```

Provide the names of multiple textures to be replaced as shown:

```
-water water1.rgb -water water2.rgb
```

Rebuilding terrain tiles

Like the VRSG terrain tiles built in MVRsimulation Terrain Tools, a terrain tile is read-only by default, which means the terrex-oflt2mds utility will not rebuild an existing tile in a subsequent conversion of the OpenFlight database. To force the rebuilding of one or more terrain tiles, for example in the case of converting an updated source OpenFlight database, you can either delete the tile(s) or make them writeable. In this way, you can rebuild subsets of terrain tiles if the source OpenFlight database changes without having to rebuild the whole database. If you suspect any tiles were not fully built, you can rerun the conversion to build any tiles that were not completed earlier.

Logging the conversion process to an output file

In order to determine which terrain tiles you would need to rebuild, the utility enables you to capture the output of a terrex-oflt2mds process so that you can later examine which FLT files were used to build each MDS tile. Having this information will help determine which terrain tiles to delete and rebuild if there is a change to a source FLT file. For example, at the command-line you could enter a command of the following form:

```
terrex-oflt2mds > mylog.txt
```

Output of the process will look similar to the following example:

```
building tile 4bd42600.mds
  flight_0_48.flt
  flight_1_48.flt
  flight_0_47.flt
  flight_0_49.flt
  flight_2_49.flt
  flight_1_49.flt

building tile 4bd42700.mds
  flight_0_50.flt
  flight_2_50.flt
  flight_0_52.flt
  flight_2_52.flt
  flight_4_52.flt
  flight_1_48.flt
  flight_1_50.flt
  flight_1_52.flt
  flight_3_50.flt
  flight_3_52.flt
```

Using FBX and OpenFlight models in 3D terrain creation

You can use FBX or OpenFlight models converted to HPX format as described in this chapter to add additional content to 3D terrain in MVRsimulation's round-earth (MDS) format.

To use FBX or OpenFlight models in the 3D terrain creation process:

1. Do one of the following:
   - Create a set of VRSG terrain tiles using MVRsimulation Terrain Tools.
   - Convert the OpenFlight database to a set of VRSG terrain tiles.

2. Convert the FBX or FLT models to HPX format as described earlier in this chapter. Inspect the models in the Model Viewer to make sure that they are satisfactory. Place the models in the \MVRsimulation\VRSG\Models\User\ directory and their textures in a \Textures subdirectory under the \User directory.
3. Create a cultural feature file (vrsg.clt) by doing one of the following:
   - Create the file in Notepad or another ASCII editor and add to the file the name of the model and its position coordinates and orientation in the database. Obtain the coordinates from the VRSG scene display. Place the vrsg.clt file in the directory where the terrain is located.
   - Run VRSG to visualize the 3D terrain and drag one or more of your models from the Windows Explorer to the terrain. Attach to each model to fine-tune its placement and orientation. Press J to save a new or updated vrsg.clt file with the information about the newly placed model.
4. If any model will be used as a dynamic entity, add an entry for the model to the ModelMap.ini file. Otherwise, proceed to step 5.
5. Run VRSG to visualize the database and inspect the model's placement and orientation.
6. If necessary, fine-tune the feature placement by either opening the vrsg.clt file and editing it or attaching to the model in real time to move and orient it directly.

Storing user-built or custom models for use in VRSG

MVRsimulation recommends you store your site's own content separately from the content installed with VRSG. Create a subdirectory for storing the models, in the installed \MVRsimulation\VRSG\Models path called \User, to keep them separate from VRSG's native models: \MVRsimulation\VRSG\Models\User.

VRSG's policy for handling the ground clamping of models depends on where the model is stored, particularly with respect to how culture models are ground-clamped. (Clamping is based on the origin of a building model, but based on the bottom of models that are fences, bushes, signage, and so on. Thus MVRsimulation strongly recommends storing you site's models in subdirectories as shown:

```
\User\Military
\User\Commercial
\User\Characters
\User\Buildings
\User\Trees
\User\Signs
\User\Other
```

Create a \Textures subdirectory under each\User directory to store the textures of models stored in that subdirectory, as shown:

```
\MVRsimulation\VRSG\Models\User\Buildings\Textures
```

In a VRSG session, VRSG will search the \User subdirectories in the same manner it searches its own installed model directories, which means you do not need to explicitly specify this directory in the Folders for Terrain, Models, Scenarios, and Other Content list.

CHAPTER 9

MVRsimulation 3D Model Formats

MVRsimulation’s Hierarchical Polygon Set (HPX)™ and HPY formats are used to encode 3D dynamic and static models for the VRSG 3D visualization system. HPX and HPY are optimized runtime formats for VRSG; they are not currently supported by any 3D modeling package or CAD system. However, you can create models in the FBX or OpenFlight format and then convert them to the HPX model format with MVRsimulation’s converters as described in the chapter “Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats.”

This chapter provides an overview of MVRsimulation’s model formats.

About MVRsimulation’s HPY format

MVRsimulation HPY files are compressed binary files that store all data associated with a given model. In particular, an HPY model file contains all of the textures referenced by that model, which eliminates the burden of maintaining a separate set of associated texture files along with each model. All models in MVRsimulation's 3D content library are provided in HPY format only. HPY models can be used in exactly the same manner as MVRsimulation’s older HPX format models. VRSG and MVRsimulation’s terrain building applications continue to support the HPX model format, and MVRsimulation’s model conversion utilities continue to convert FBX and OpenFlight models to the HPX model format.

Contents of the HPX model file

If you need models that are not available in MVRsimulation’s model library, you can create models in FBX or OpenFlight format with modeling tools such as Autodesk 3ds Max to Presagis Creator and then, as mentioned above, convert the models to MVRsimulation’s HPX model format. You can also edit an HPX file directly to make simple modifications such as changing an assigned texture. HPX files are ASCII files that you can modify easily in a text editor, such as Notepad. If you understand 3D computer graphics concepts, you can create HPX files directly to make simple shapes such as boxes, cross trees, and so on.

The first line in the HPX file is the HPX version identifier, currently HPXV002. The second line in the file identifies the number of entries in the texture palette and the material palette respectively. For example:

```
HPXV002
2 1
```

Following the texture and material palettes is the serialized HPX hierarchy, beginning with the parent (root) node. Each node in the hierarchy, except for geometry nodes, takes one line in the file.

Texture file entries

Each texture is listed on a separate line in the HPX file, starting with the third line. If the model has no textures, the third line begins listing the material palette. VRSG supports the texture formats that are listed in the chapter "Manipulating Textures." Note that .tex is an MVRsimulation proprietary texture format.

The texture file entry in the HPX file supports three texture attributes after the texture name. The attributes are wrapU, wrapV, and backface.

Syntax:

```
texture_file_name.[rgb, rgba, bmp, tex] wrapU wrapV backface
```

For example:

```
4 1
shed-004composite-door.tex 1 1 0
shed-004_walls3.tex 1 1 0
shed-004shng03L.tex 1 1 0
shed-004cncr20L.tex 1 1 0.
.
.
```

Similar to the texture attributes in a .tlb file as described in the chapter "Manipulating Textures", wrapU and wrapV are Boolean values that enable a texture to repeat in the horizontal and vertical direction respectively. A value of 1, the default, allows the texture to wrap in the appropriate direction. A value of 0 restricts the texture to a single repetition. The wrapU and wrapV values can be overridden by an OpenFlight *.attr if the texture is present. Rarely are wrapU and wrapV turned off (0).

The third attribute determines if backfaces should be enabled for geometry textured with this texture. With backfacing enabled, both sides of a polygon are drawn. Backfacing is typically not used, except for objects such as trees and helicopter blades, for which you would want to see both sides of polygon.

If no values are present in a texture file entry, as in some of the entries in the above example, the default values assumed are "1 1 0", which enable texture wrapping on both axes and disable backface drawing.

Material palette entries

The material palette determines how geometry reacts with light sources. Each vertex in the model is attributed with a material code. This code is used to access data in the material palette which in turn is used in the lighting equation for that vertex. Each entry in the material palette is listed on a separate line, following the last texture in the texture palette. The material palette has the following syntax:

```
AR AG AB DR DG DB SR SG SB ER EG EB shininess alpha
```

The following table describes each value:

Value	Description
AR	How much red the material reflects of ambient light, range 0..1
AG	How much green the material reflects of ambient light, range 0..1
AB	How much blue the material reflects of ambient light, range 0..1
DR	How much red the material reflects of diffuse light, range 0..1
DG	How much green the material reflects of diffuse light, range 0..1
DB	How much blue the material reflects of diffuse light, range 0..1
SR	Red contribution to specular highlights, range 0..1
SG	Green contribution to specular highlights, range 0..1
SB	Blue contribution to specular highlights, range 0..1
ER	How much red light the material emits, range 0..1
EG	How much green light the material emits, range 0..1
EB	How much blue light the material emits, range 0..1
Shininess	Controls the tightness of specular highlights, 120 creates a very tight highlight, whereas 10 creates much duller looking surface
Alpha	Transparency of the material, range 0..1 where 0 is fully transparent and 1 is fully opaque

You can try increasing the ambient and diffuse material response (the first 6 numbers) of all materials to 1.0 if a model appears too dark when it is rendered in VRSG. If the model still appears too dark after you make this change, you may need to brighten its textures in an image editing application such as Adobe Photoshop or Corel PaintShop Pro.

If you want to remove the shine from a model and give it a matte finish, you would set the shininess parameter to 0.0.

Levels-of-detail

Following the texture and material palettes for a model is a serialization of the model's hierarchy. To protect the content, the exact syntax of the model format cannot be precisely described. You can make changes however to Level-of-Detail (LOD) ranges encoded in the model. This is often required if a model switches LOD states too quickly for one's preference.

To edit LOD ranges of a model, search the file for occurrences of the text "LOD." The format of an LOD entry is as follows:

```
LOD n minDist maxDist cx cy cz
```

An LOD entry identifies the minimum range and maximum range for which the subordinate geometry will be displayed. When making changes to LODs, it is best to simply scale all minDist and maxDist values in the file by a constant value. For example, to make a model change LOD states less quickly, scale these terms by a value greater than 1. If you change one LOD entry, it is recommended you change all LOD entries in a file, as LOD entries might represent a continuum of LOD states. Failure to do so might result in gaps in the LOD space resulting in periods where the model or portions of the model disappear. As an example, if a given piece of the model and an LOD ranges of (0,1000), and a lower detail version of the model had an LOD range of (1000,4000), you would want to change the 1000 to the same number in both instances to avoid gaps or overlaps in LOD space. By simply scaling the terms in the entire file as described above, these problems can be avoided.

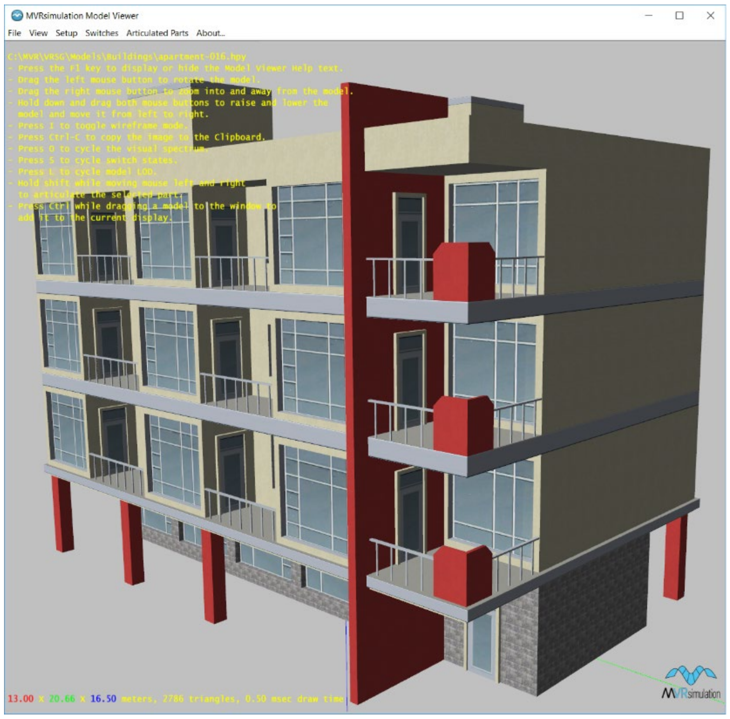
Example

The following example shows the top portion of the contents of a model's HPX file. This model, apartment-016, is one of over 4,000 cultural feature models that are delivered with VRSG:

```
HPXV001
9 1
AP-STUC01.rgb 1 1 0 0 0
ET-Electric01.rgb 1 1 0 0 0
AP-FACADE01.rgb 1 1 0 0 0
AP-BORDER01.rgb 1 1 0 0 0
AP-STUC02.rgb 1 1 0 0 0
AP-WIN01.rgb 1 1 0 0 0
GT-WALLINSIDE1.rgb 1 1 0 0 0
AP-ROOF01.rgb 1 1 0 0 0
AP-DOOR01.rgb 1 1 0 0 0
0.722000 0.722000 0.729000 0.917647 0.917647 0.917647 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 10.000000 1.000000
Parent 1
Switch 2 18 0 18
Parent 4
LOD2 10 0.000000 150.000000 0.000000 0.000000 -8.250000
Parent 1
Geometry 0 472 396 "SH_Mesh_0"
6.000000 10.200012 -3.000000  9.967366 3.907161  255 255 255 1.000000
0.000000 -0.000000 0 255
6.000000 -10.200012 -3.000000  9.967366 -22.404495  255 255 255
1.000000 0.000000 -0.000000 0 255
6.000000 -10.200012 -3.100006  10.096343 -22.404495  255 255 255
1.000000 0.000000 -0.000000 0 255
6.000000 10.200012 -3.100006  10.096343 3.907161  255 255 255
1.000000 0.000000 -0.000000 0 255
-6.000000 10.200012 -13.500000  -15.810656 -11.995148  255 255 255
0.000000 1.000000 -0.000000 0 255
-3.000000 10.200012 -13.500000  -15.810656 -8.115238  255 255 255
0.000000 1.000000 -0.000000 0 255
-6.000000 10.200012 -14.500000  -17.096933 -11.987417  255 255 255
0.000000 1.000000 -0.000000 0 255
6.000000 10.200012 -14.500000  -17.096939 3.532219  255 255 255
0.000000 1.000000 -0.000000 0 255
6.000000 10.200012 -13.500000  -15.810657 3.524487  255 255 255
0.000000 1.000000 -0.000000 0 255
-6.000000 -7.000000 -10.000000  -23.342546 8.488424  255 255 255
```

```
0.000000 1.000000 -0.000000 0 255
-6.000000 -7.000000 -7.000000  -23.288839 4.659811  255 255 255
0.000000 1.000000 -0.000000 0 255
```

Here is how the apartment-016 model appears in the MVRsimulation Model Viewer:


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

```
C:\MVR\VRSG\Models\Buildings\apartment-016.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

13.00 x 20.66 x 16.50 meters, 2786 triangles, 0.50 msec draw time
```

The Model Viewer is described in the chapter “Previewing Models, Effects, and Terrain.”

Constructing light points

Light points are small light source models that emit light from a specific location. These self-luminous points of light do not illuminate anything around them. Light points are mainly used to simulate airfield lighting systems, but can be used simulate other culture lighting.

Once light point models have been built, they can be derived from shapefiles, which are created using GIS tools. For example, using shapefile input, light points can be built into terrain tiles using MVRsimulation’s Terrain Tools.

Light points are not the same as a model’s emissive polygons or light maps. Light points also differ from light lobes, which illuminate their surroundings and must be turned on by the simulation. Headlights and flares are examples of light lobes and are described in the chapter “Configuring Models and Events.”

There are two ways to construct a light point model: by hand in a text editor, or in a modeling tool that can output an FBX or OpenFlight model such as Autodesk’s 3Ds Max or Presagis Creator. FBX and OpenFlight models can be converted to MVRsimulation’s HPX format using the conversion utilities that are delivered with VRSG, which are described in the chapter “Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats.”

This section describes both methods.

Creating light points manually in a text editor

In an HPX file that you create in Notepad or another text editor, you denote a light point by the keyword `LightString`.

A LightString entry has the following form:

```
LightString <num-children> <num-verts> <period> <time-on> <size>
<min-size> <max-dist> <az> <el> <vfov> <hfov> <rot-rate> <flags>
```

Like other HPX nodes, the `LightString` keyword is followed by the number of child nodes in the HPX hierarchy. For light strings, this is typically 0 (zero).

For example:

```
HPXV0010 0
Parent 1
LightString 0 1 1.000000 1.000000 0.250000 1.000000 3000.000000
0.000000 90.000000 180.000000 180.000000 0.000000 00.000000 0.000000
-0.000000  0.000000 1.000000  255 255 178 0.000000 0.000000 1.000000
0
```

The `LightString` entry is followed by a listing of light point vertices, the number of which is indicated by the num-verts field. Each light point vertex has the following syntax:

```
X Y Z P A R G B
```

Where:

X Y Z is the coordinate of the light point in model space. Typically the model is translated to the airfield position by a clt file for the database, allowing light point vertices to be defined in a local coordinate system. If the clt file contains the comment "! clamp_lp=on", the Z coordinate is considered as relative to the ground level at the XY coordinate of the light point. By default, light point ground clamping is disabled. When the comment "! clamp_lp=off" is parsed in the database's clt file, light point ground clamping is turned off, and the Z values are considered to be local to the containing model.

P is the phase delay of an animated light. The phase delay is given in seconds. This parameter is useful for creating a string of lights with a strobe effect such as a runway sequencer.

A is the intensity of the light point. 0.0 corresponds to minimum intensity and 1.0 corresponds to maximum intensity.

R G B defines the color of the light point given as red, green, and blue components. Valid ranges for these parameters are 0 for minimum intensity, and 255 for maximum intensity per component.

The *period* of a light string indicates the length of the animation cycle in seconds. This value applies to all lights in the string. A phase delay is applied to individual lights on a per-vertex basis as described above.

The *time-on* of a light string indicates the duty cycle of the light. For blinking lights, this value will be less than the period. Non-blinking lights will have a time-on equal to their period.

The *size* parameter indicates the real-world diameter of the lights in meters. Lights are drawn in an appropriate size as a function of their distance until they are smaller than *min-size* pixels. Lights will be drawn no smaller than *min-size* pixels, regardless of their distance to the viewer. Note that when the real-world size projects down to smaller than the minimum pixel size, the displayed size is clamped to the minimum pixel size. As result of the resolution dependence, if you run VRSG at a lower resolution on a different display, the light points might appear larger.

The *max-dist* parameter indicates the maximum distance a light is visible in meters. VRSG uses a range roll-off algorithm to attenuate the intensity of a light as a function of its range. sAt its maximum range, the intensity of a light becomes zero making it invisible.

The *az* and *el* parameters define the direction of a directional light string. Both values are provided in degrees. Az is provided relative to North and el is provided relative to the ground plane.

Vfov and *hvov* define the full-angle vertical and horizontal field-of-view (fov) of a light string. For omni-directional lights, a value of 180 should be used for both parameters.

For directional lights, the intensity of a light is attenuated by the angle in which the viewer is within the light's field-of-view. This angular attenuation can be defeated by the *flags* parameter (see below).

Rot-rate describes the rotation rate of a light string in degrees per second. Non-rotating, or omni-directional lights should have a rot-rate of 0.

Flags define a bit mask indicating special treatment of a light string. The value of flags is a logical OR of all desired features. The current set of features includes:

0x1 – do not perform angular attenuation on the top edge of a light's fov

0x2 – do not perform angular attenuation on the bottom edge of a light's fov

0x4 – do not perform angular attenuation on the right edge of a light's fov

0x8 – do not perform angular attenuation on the left edge of a light's fov

0x200 – render light in the visible spectrum

0x400 – render light in the IR spectrum

0x800 – render light in the Near-IR (NVG) spectrum

The following example is an excerpt from a vasi light point model, airport-light-vasi.hpx:

```
HPXV001
2 1
ASPHALT.rgb 1 1 0 0 0
papi.rgb 1 1 0 0 0
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.000000
Parent 12 "g3"
LightString 0 1 1.000000 1.000000 0.400000 0.250000 20000.000000
180.000000 0.000000 6.000000 179.000000 0.000000 0
-0.700000 1.951771 -1.097769  0.000000 1.000000  255 0 0 0.000000 -
1.000000 0.000000 0
LightString 0 1 1.000000 1.000000 0.400000 0.250000 20000.000000
180.000000 46.500004 87.000000 179.000000 0.000000 0
-0.700000 1.951771 -1.097769  0.000000 1.000000  255 255 255 0.000000
-0.688355 0.725374 0
```

```
LightString 0 1 1.000000 1.000000 0.400000 0.250000 20000.000000
180.000000 0.000000 6.000000 179.000000 0.000000 0
-0.700000 2.402193 -1.097769  0.000000 1.000000  255 0 0 0.000000 -
1.000000 0.000000 0
```

For information about how to implement runway light points in a cultural feature file, see the chapter "Configuring Models and Effects."

Creating FBX or OpenFlight light point models for VRSG

In addition to editing or creating HPX files in a text editor to add or create light point models, you can create light point models in a modeling tool that can output an FBX or OpenFlight model such as Autodesk's 3Ds Max or Maya or Presagis' Creator. FBX and OpenFlight models can then be converted to MVRsimulation's HPX format using the conversion utilities that are delivered with VRSG, which are described in the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."

Creating FBX light point models for VRSG

LightPoint nodes are mesh nodes that define a single light point vertex. A light point vertex consists of position (X, Y, Z), phase (P) and intensity (A) and color (R, G, B). X, Y, Z are defined by the Mesh's control points. Phase (P) indicates the phase delay in seconds of the current light point. Both phase (P) and intensity (A) are stored in the Mesh's UV layer. R, G, B are stored in the diffuse property of the mesh's material layer. There are no custom properties for a LightPoint node.

VRSG-specific nodes are modeled in FBX by adding prefixes to node names in the modeling tool. To add light points to an FBX model use the following two prefixes to node names: LIGHTGROUP_ and LIGHTPOINT_. The LIGHTGROUP_ prefix modifies an FBX NullNode. The LIGHTGROUP_ node has one or more child nodes. Each child is an FBX Mesh node with the prefix LIGHTPOINT_. Each LIGHTPOINT_ node defines the location (X, Y, Z), the phase (P) and intensity (A), and the light point color (R, G, B) of one light point. The location coordinates are defined by the mesh's control points. The phase and intensity values are defined in the mesh's UV layer. The color is defined in the diffuse color property of the mesh's material layer. A LIGHTGROUP_ node can have any number of LIGHTPOINT_ child nodes. The LIGHTPOINT_ node itself does not have any child nodes.

You add custom properties to the LIGHTGROUP_ node to define the light point's period, time-on, size, min-size, max_size, az, el, vfov, hfov, rot-rate, and flags as described in the HPX light string parameters in the previous section, "Creating light points manually in a text editor." The syntax/type/units of these properties must be exact as described in the following table:

Custom property	**Description**	**Syntax/type/units**
Period	A floating-point number indicating the length of the animation cycle in seconds.	Period / float / seconds
TimeOn	A floating-point number indicating the duty cycle for the light points in seconds. Blinking lights will have a TimeOn less than the Period.	TimeOn / float / seconds
Size	A floating-point number indicating the real-world diameter of the light points in meters.	Size / float / meters
MinSize	A floating-point number indicating the minimum size in pixels of the light points diameter.	MinSize / float / meters
MaxDist	A floating-point number indicating the maximum distance in meters that the light points are visible.	MaxDist / float / meters
Az	A floating-point number indicating the number of degrees relative to North that light points are visible.	Az / float / degrees
El	A floating-point number indicating the number of degrees relative the ground plane that the light points are visible.	El / float / degrees
VFov	A floating-point number defining the full-angle vertical field-of-view in degrees. For omni-directional lights a value of 180 degrees should be used.	VFov / float / degrees

Custom property	**Description**	**Syntax/type/units** *(cont.)*
HFov	A floating-point number defining the full-angle horizontal field-of-view in degrees. For omni-directional lights a value of 180 degrees should be used.	HFov / float / degrees
RotRate	A floating-point number indicating the rotation rate in degress per seconds. For omni-directional light points a value of 0 should be used.	RotRate / float / degrees
Flags	Flags define a bit mask indicating special treatment of a light string. The value of flags is a logical OR of all desired features.	Flags / string / hex (ex: "0x03")

Creating OpenFlight light point models for VRSG in Presagis Creator

VRSG contains light point features that are not available in OpenFlight, and similarly OpenFlight defines some light point features not supported by VRSG. There is, however, sufficient overlap in capability to meet most requirements. This section describes the light point attributes that will propagate from OpenFlight into exported HPX models. Attribution supported by VRSG that is not part of the OpenFlight standard will require that you edit the resulting HPX model in a text editor to make use of those particular features.

The following settings are taken from Creator's light point animation palette editor, under the Non-Vega Prime Attributes section.

Period – specify this parameter using the *Period* field of Creator's light point animation palette. This field is *only* used by MVRsimulation's HPX export plugin if the *Animation Type* is *Flashing*.

Phase Delay – this field is not read by the HPX export plugin, as it does not allow a per-vertex phase delay control. The HPX export plugin for Creator automatically assigns a per-vertex phase delay that is calculated from the number of lights in the string in conjunction with the period of the animation. The *Animation Type* must be set to *Flashing* for the phase delay to be automatically generated per-vertex. Use this approach to create a sequenced strobe light string.

Time On – this field defines the amount of time a light is on in a flashing animation.

Rotation Rate – if the *Animation Type* is set to *Rotating*, the Period field defines the rotation rate in terms of 360° revolutions per second.

The following settings are taken from Creator's light point appearance palette editor, under the Non-Vega Prime Attributes section.

Size – specifies the actual size (diameter) of the light point in meters. This parameter is taken from the *Actual Size* setting in Creator's light point appearance palette. VRSG will display the light point using perspective projection based on this actual size, until at which point the light becomes smaller than the minimum pixel size.

Min Pixel Size – specifies the minimum pixel size (diameter) that the light point will be displayed. If perspective projection of the actual size causes the light point to be smaller than this minimum pixel size, the light point will be held to the minimum pixel size. This parameter is taken from the *Minimum Pixel Size* setting in Creator's light point appearance palette editor.

The following settings are taken from Creator's light point appearance palette editor, under the Vega Prime Attributes section:

Vertical FOV – defines the vertical FOV of directional lights. This is taken from the *Vertical Lobe Angle* field.

Horizontal FOV – defines the horizontal FOV of directional lights; taken from the *Horizontal Lobe Angle* field.

Visibility Range – defines how far the light is visible (meters).

The orientation of directional lights is inferred from the vertex normal assign to an individual light point vertex. The HPX export plugin will use this information to populate the azimuth and elevation fields of the light string.

The HPX export plugin obtains the per-vertex color and intensity from each OpenFlight light point vertex.

The flags field of the HPX light string has no OpenFlight analog; therefore, if you want to engage any of its features as described above, you must edit the HPX file in a text-editor following its export from Creator.

Adding HPX features to OpenFlight models

This section describes how to add metadata to an OpenFlight model in Presagis Creator before exporting it to HPX format to enhance the dynamic display of a moving model in VRSG. For information about MVRsimulation's HPX plug-in for Creator and MVRsimulation's other OpenFlight conversion tools, see the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."

Shadow map and light map support

For models created in Presagis Creator VRSG includes enhanced multi-texture support, including shadow maps and light maps. For faces with two textures applied, the second texture will be treated as a light map or a shadow map. In daytime viewing, the alpha channel of the second texture will be interpreted as a shadow map. In nighttime viewing, the RGB channels of the second texture will be interpreted as a light map.

The World Reference Model

MVRsimulation's OpenFlight conversion tools support a large subset of the World Reference Model (WRM). WRM is a set of metadata added to OpenFlight models in comment fields that make models "smart" for use in distributed interactive simulations. You can obtain the WRM documentation from the website: www.presagis.com/files/standards/wrm.pdf.

If there is no WRM information present in a model, MVRsimulation's OpenFlight converters will assume:

- All switch nodes are damage states. The first child is the normal state and the second child is the destroyed state.
- All DOFs become transformation nodes in HPX. DOFs that rotate about the Z axis are assumed to be the turret (DIS part code 4096), and all DOFs that rotate about the X axis are assumed to be the gun (DIS part code 4416).

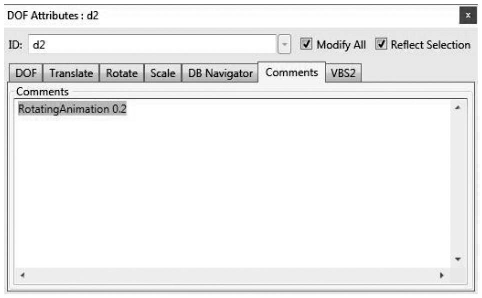
HPX-specific animation enhancements

In addition to supporting a large portion of the WRM, MVRsimulation's OpenFlight conversion tools also provide animation functionality to models that you can invoke by adding HPX constructs in the OpenFlight comment fields. This section describes how to add animation to a model.

For information about how to configure a model for IR animations in Creator, see the chapter "Working with Sensor-Modes and Physics-Based IR."

Rotating and wheel animations

To create an object that rotates about a given axis in an animated manner, you can use the HPX RotatingAnimation construct to avoid the duplicated geometry necessary for a flip-book animation. In addition, rotating animations have essentially an infinite number of states that are derived on-the-fly based on delta time and the desired rotation rate. Thus they yield much smoother animations than flip-book animations. To create a rotating animation, model the object as a Degree of Freedom (DOF) node in Creator. Then add the keyword `RotatingAnimation` as a comment to the DOF node as shown in the following example:

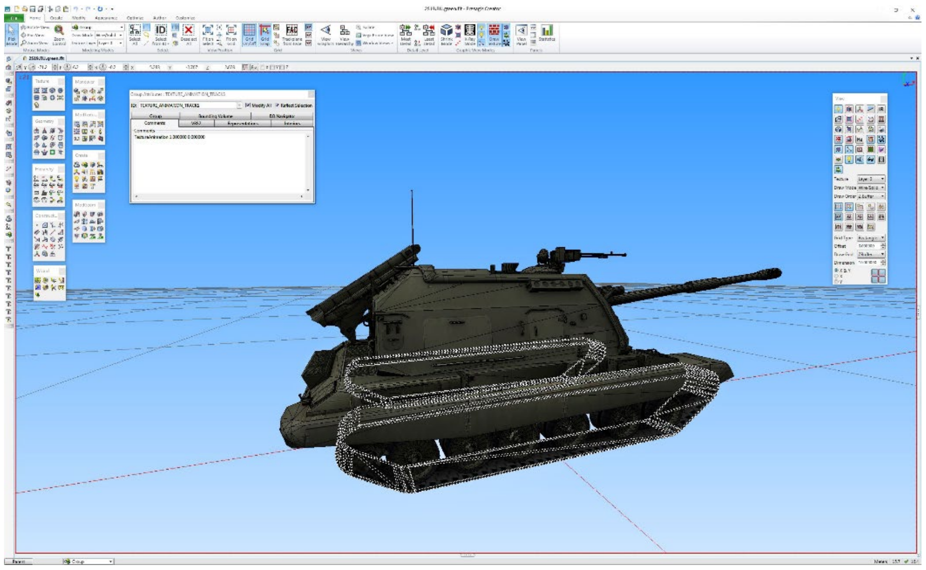


RotatingAnimation 0.2

The parameter following the keyword `RotatingAnimation` is the rotation rate given in seconds per full revolution. Thus smaller numbers produce faster rotations. DOFs that rotate about the Y axis are considered wheel animations by default. With wheel animations, the parameter is interpreted as a wheel diameter. The wheel diameter is used to compute how much to turn the wheel as a function of the vehicle speed. In the HPX file, wheel animations are distinguished from rotating animations by having their axis listed in lowercase (rotation animation is listed in uppercase). To make a rotating animation rotate about the Y axis and not be a wheel animation, you would first export the model as an HPX model and then edit the HPX file to change the axis to be uppercase. For more information about rotating animation nodes, see the discussion earlier in this chapter.

Texture animations

You apply texture animations to Object nodes in Creator only, with the keyword `TextureAnimation`. In the following example, the object containing tank track geometry is selected and the comment *TextureAnimation* is applied:



Group Attributes : TEXTURE_ANIMATION_TRACKS

ID: TEXTURE_ANIMATION_TRACKS

Comments

TextureAnimation 1.000000 0.000000

Texture animations are used to animate the U and V texture coordinates of geometry. The units given to the comment field indicate the amount of U and V offset for every 1 meter of vehicle movement. Texture animations are useful to make tank tracks appear to move with vehicle motion. VRSG animates the textures in a rate consistent with the motion of the vehicle.

CHAPTER 10

Previewing Models, Effects, and Terrain

You can use MVRsimulation's Model Viewer to preview:

- Models in MVRsimulation's HPY or HPX format.
- RGB and MVRsimulation's TEX textures.
- VRSG billboard special effect sequences (EFF).
- VRSG particle system special effects (PAR).
- VRSG cloud files (CLD).
- VRSG terrain tiles (MDS).

Within the Model Viewer you can inspect a model's articulated parts and switch states, try out a weapon model's firing effects, preview character and articulated part animations, copy models to another directory, determine how much land mass of the earth a given set of terrain tiles covers, and take screen captures of anything that is displayed in the Model Viewer window.

This chapter describes these and other previewing tasks.

Accessing the Model Viewer

To preview an MVRsimulation model, an RGB or TEX texture, cloud or effect, or an MDS terrain tile, open the ModelViewer in one of the following ways:

- Launch the Model Viewer from the MVRsimulation program folder in the Windows Start menu, choose File > Open and browse for the model you want to open.
- Double-click ModelViewer.exe, located in the \MVRsimulation\Common\Bin directory, and choose File > Open and browse for the model you want to open.
- Open Windows Explorer, locate the model, effect, or terrain tile of interest and double-click it. The Model Viewer opens, and displays your item of interest.

From within the Model Viewer, you can display another model or other item by choosing File > Open, and from the Open dialog box selecting the model or other item you want to display. Alternatively you can simply drag a model file, texture, cloud or effect, or an MDS terrain tile from Windows Explorer to the Model Viewer.

Choose File > Exit to close the Model Viewer.

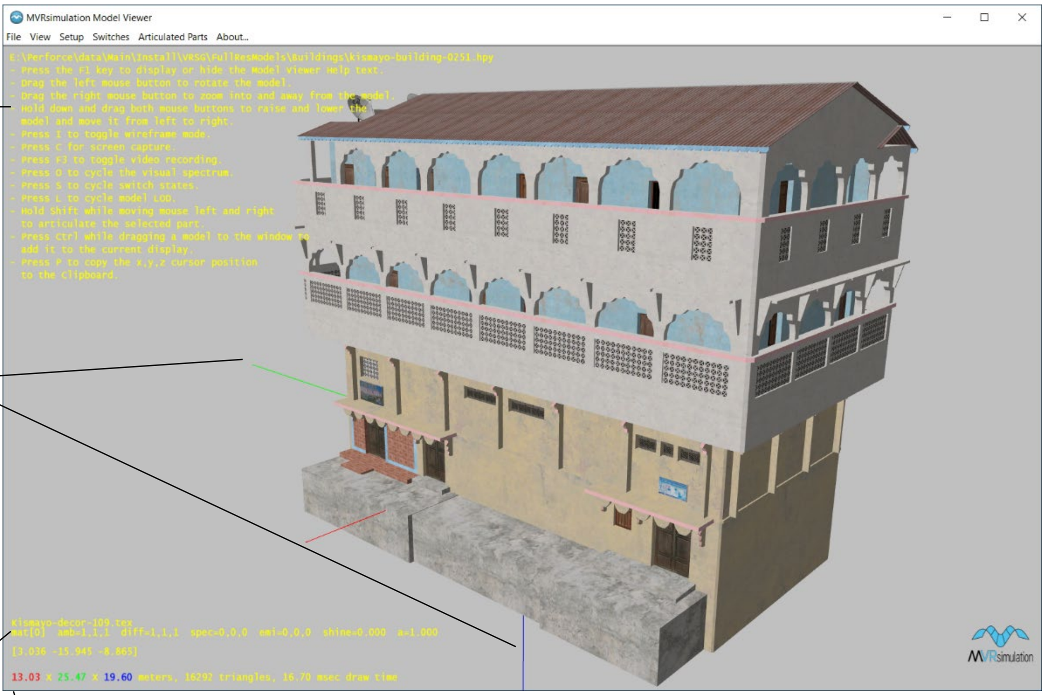
Previewing models

In the Model Viewer you can zoom and rotate the model for a closer inspection, change the display mode to see the model in wire-frame or IR mode, move the articulated parts and cycle through the model's switch states. The Model Viewer has several viewing options you can use by pressing a key on the keyboard. Many of the options are listed in the onscreen Help text. Some options apply to vehicle models only.

Help text that describes how to manipulate the model for inspection. You can remove the display of the Help, the axes, and the polygon count and dimensions by pressing F1.

The X, Y and Z axes. The Model Viewer displays the X (red), Y (green) and Z (blue) axes, which show the origin and orientation of the model. The length of a displayed axis is scaled to the bounding box along that axis.

The texture the cursor is resting on, and information about that texture.



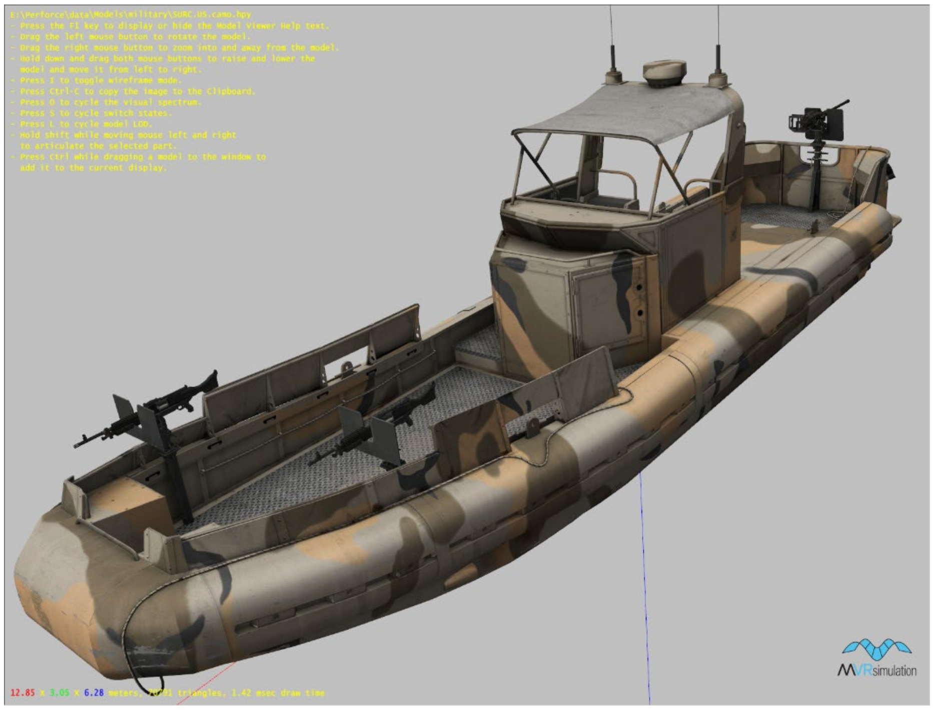
```
E:\Perforce\data\Main\Install\VRSG\FullResModels\Buildings\kismayo-building-0251.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the Clipboard.
```

```
Kismayo-decor-109.tex
mat[0]  amb=1,1,1  diff=1,1,1  spec=0,0,0  emi=0,0,0  shine=0.000  a=1.000
[3.036 -15.945 -8.865]
13.03 x 25.47 x 19.60 meters, 16292 triangles, 16.70 msec draw time
```

Dimensions of the model along the x, y, and z axes in meters, and its polygon count.

In the Model Viewer you can:

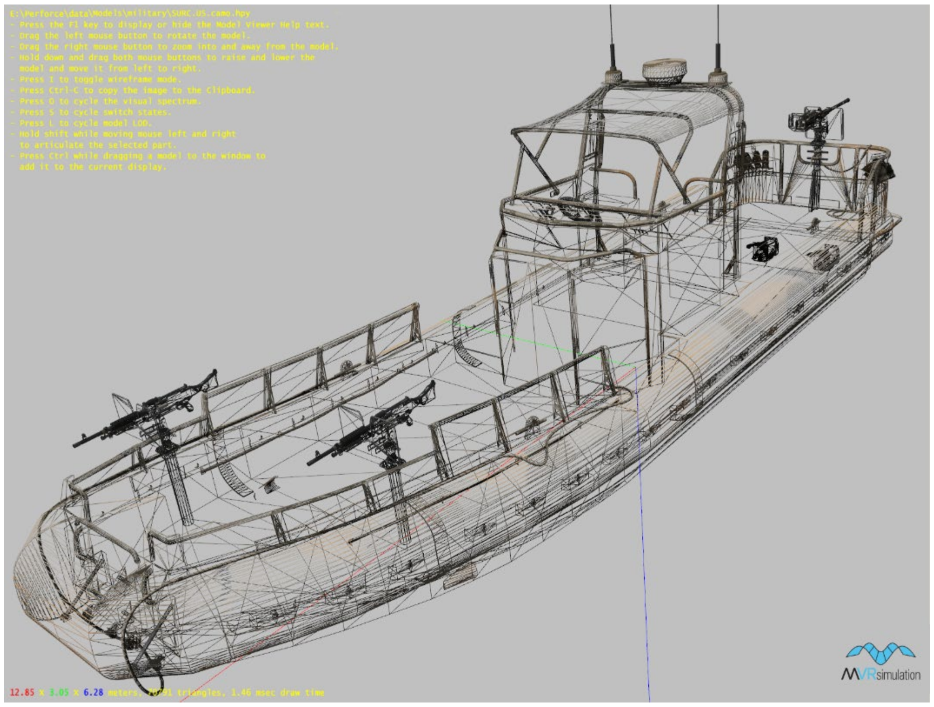
- Inspect whether the model's geometry and textures are correctly assembled.
- Zoom and rotate the model by dragging the left and right mouse buttons, as described in the onscreen Help text. (Press F1 to remove display of the onscreen Help text.)
- Cycle through 6 different fixed orthographic project views by pressing the 'G' key on the keyboard.



```
E:\Perforce\data\Models\military\SURC.US.camo.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

12.85 x 3.05 x 6.28 meters, 70791 triangles, 1.42 msec draw time
```

Displaying a model's wire frame

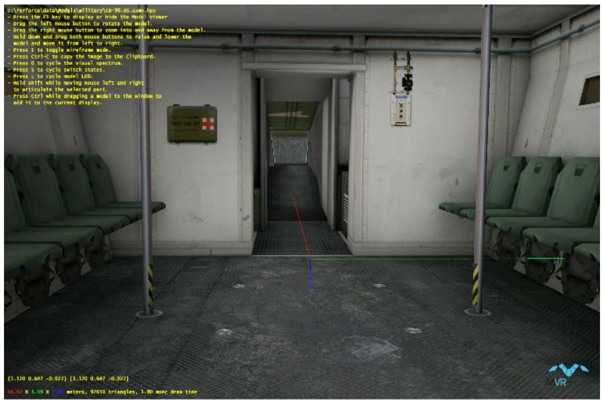
To toggle a display of the model rendered in wire-frame, press the "I" key on the keyboard.



```
E:\Perforce\data\Models\military\SURC.US.camo.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

12.85 x 3.05 x 6.28 meters, 70791 triangles, 1.46 msec draw time
```

Inspecting a model's interior

You can zoom inside the model to inspect its interior, as shown in the following examples of UH-72_MEP and CB-90 models:



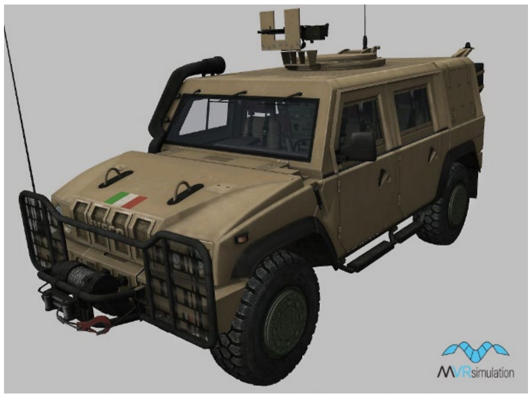
```
E:\Perforce\data\models\military\CB-90.US.camo.hpy
- Press the F1 key to display or hide the Model Viewer
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

(1.120 0.647 -0.022) [1.120 0.647 -0.022]
16.47 X 5.59 X [illegible] meters, 97650 triangles, 1.90 msec draw time
```

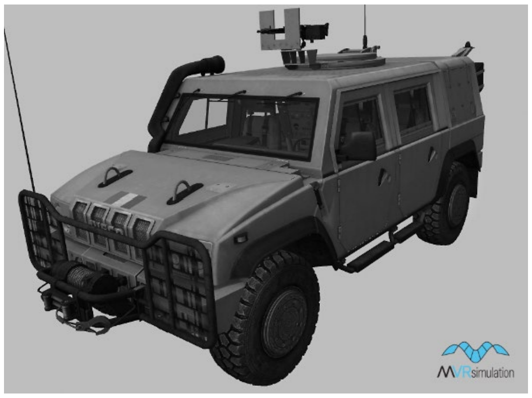
Displaying a model in different sensor modes

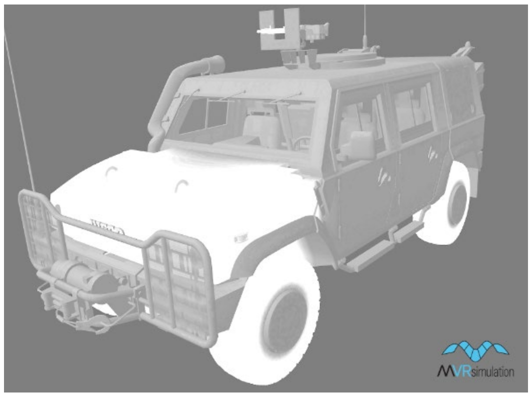
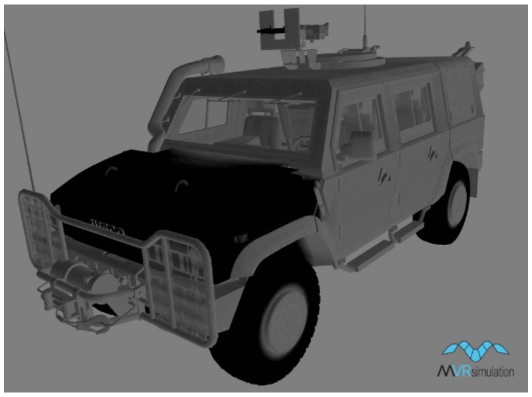
To cycle the model through the visual spectrum, press the letter "O" key on the keyboard. Doing so cycles the model through the standard daylight mode and several sensor modes: monochromatic EO Day TV mode, and two infrared (IR) modes. This is useful for inspecting a model's IR textures. The military vehicle models that are delivered in MVRsimulation's 3D content libraries have built-in IR textures that represent heat signatures or hot spots, and you can preview them in the Model Viewer's IR modes. These IR heat signatures fade in and out in response to simulation events.

You can preview a model in the Model Viewer in the following types of VRSG sensor views:

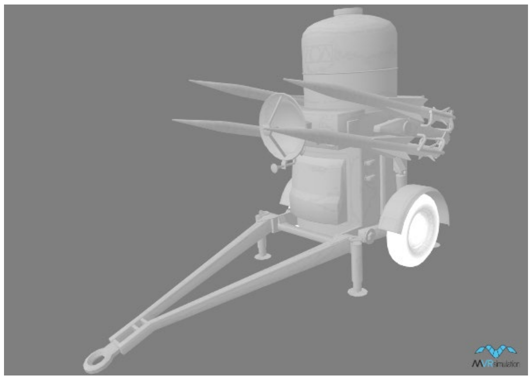
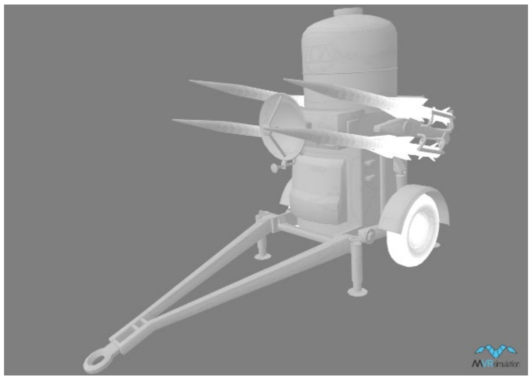
- Normal out-the-window (OTW).
- Daylight television mode (Day TV) sensor.
- IR white hot – where thermally hot areas are gradations of white.
- IR black hot – where thermally hot areas are gradations of black.
- Green IR.





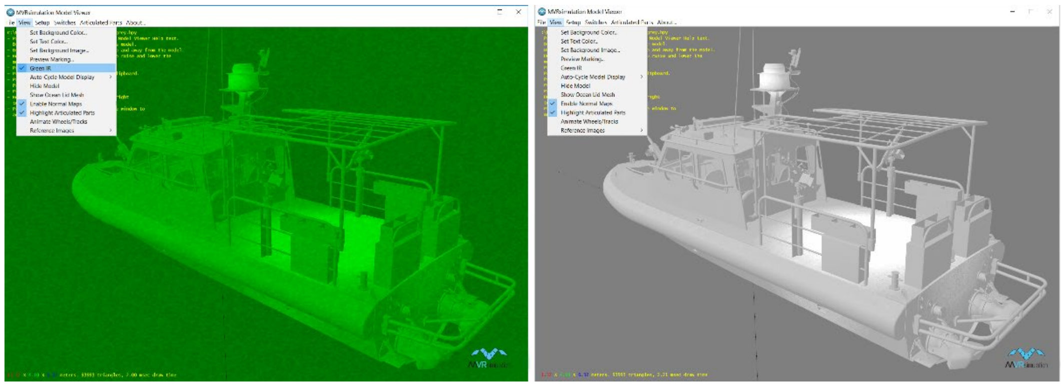



The Model Viewer's IR hot modes initially show all the hot spots on a given model at full intensity. However, you can modulate the intensity of each IR hot spot in the model by rolling the mouse wheel slowly on a thermal area while the model is displayed in IR white hot or black hot mode. You can modulate each hot spot separately, as each hot spot is unique. (In VRSG each hot spot can be stimulated independently for a thermal response.) The following example shows a model that has separate hot spots for the wheels and the munitions:

You can configure the hot spots for a model in Model Viewer and save it as part of the model state in a PRT file as described in the section "Saving a model's switch state, articulated part position, and IR hot spots."

You can display a model in IR mode in monochrome or green, by toggling the Green IR item on the Model Viewer's View menu:



Note: In VRSG, a model's hot spots are not shown as hot by default until there is some event or other stimulation to activate the hot spots (unless you have configured them in a PRT file to always be hot), as described in the chapter "Working with Sensor View Modes and Physics-Based IR."

Measuring the distance between two points in a model

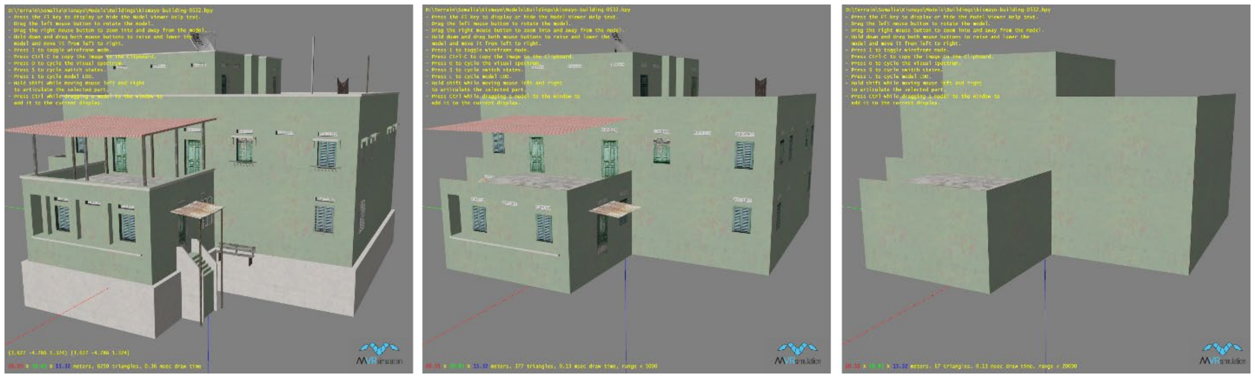
To measure the distance between two points on a model, click the mouse wheel for the first point, and then click it again at the endpoint. The Model Viewer will display the distance between the two clicked points, in meters.

Displaying the model at different texture resolutions

To inspect a model's texture display at different resolutions, press "T" to cycle through full, half, and quarter resolutions. The resolution of the view is listed in the lower left corner of the viewer. This option is useful for assessing the resolution at which models should be included in your virtual world, determining which are crucial to include at the highest resolution. The following example shows a vehicle model at full, half, and quarter resolution:

Displaying a model at different levels of detail (LOD)

To cycle through a model's levels of detail representations, press the "L" key. Each successive display shows the model with a reduced number of polygons; the polygon count is noted in the text in the lower left corner of the viewer. Three of the four LODs for a building model are shown in the following example, in which the polygon count is reduced from 6250 to 377 and then to 67:



All 3D vehicle and character models and most building models in MVRsimulation's 3D content libraries have multiple LODs.

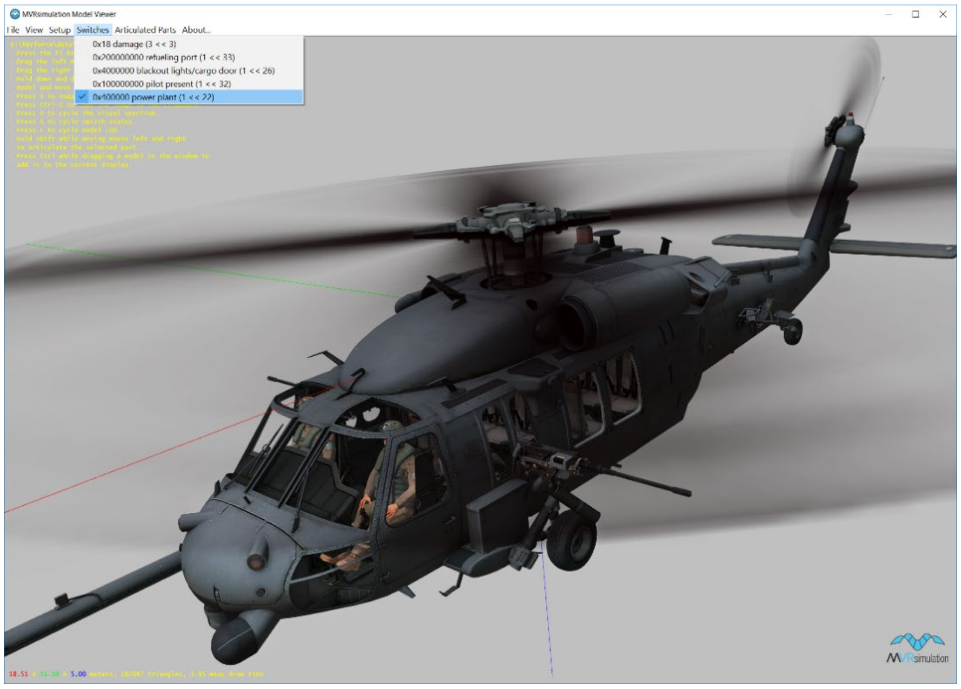
For a character model, you can see the physics model, a lightweight mesh with a negative LOD-range when you press "L" the first time. VRSG uses this mesh for collision detection.

Displaying a vehicle model's switch states

Most MVRsimulation vehicle models have at a minimum two or more possible switch states: a damaged state and a healthy state. Aircraft models typically have an Engine (DIS "power plant") state which, when set, lifts the wheels, stows the landing gear, and spins propellers.

VRSG supports 64-bit appearance masks, which enables advanced features of newer MVRsimulation models to be assigned to bits beyond those specified by the SISO standard.

A model's switch states are listed in the Switches menu, as shown:


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
0x18 damage (3 << 3)
0x200000000 refueling port (1 << 33)
0x4000000 blackout lights/cargo door (1 << 26)
0x100000000 pilot present (1 << 32)
✓ 0x400000 power plant (1 << 22)

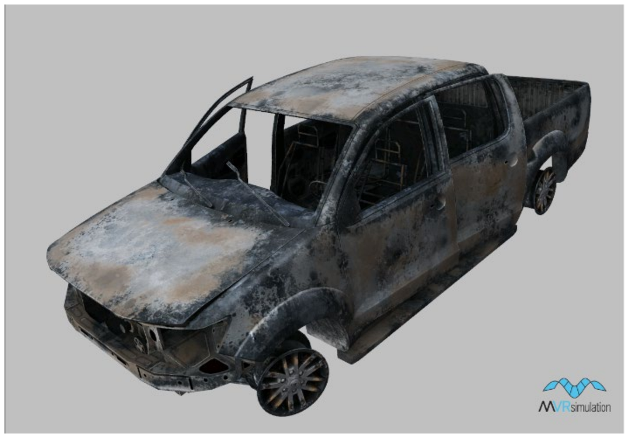
The switch state features vary from model to model, depending on the number of articulated parts and other features that have a state. For instance, many but not all models have lights. Some models have hatches, or refueling ports, or other components that open and close. Ground vehicle models have fire power and mobility states. Fixed wing aircraft have a wheels-down landing state. Where possible, the switch masks follow the DIS standard for the appearance bits. Many models take advantage of unused DIS appearance bits for custom features such as wingstores.

To inspect a model's switch state for an articulated part or damage state, choose it from the Switches menu, and then press the "S" key on the keyboard to toggle the switch state on and off. For example, select Power Plant from the Switches menu and press the S key to cycle the states of the selected power plant/engine switch.

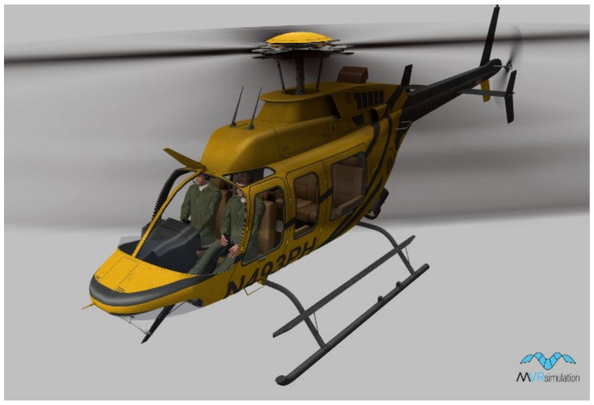
You can also use shortcut keys for displaying these states:

- D to toggle damage state.
- E to toggle engine state.
- H to toggle the hatch open state.

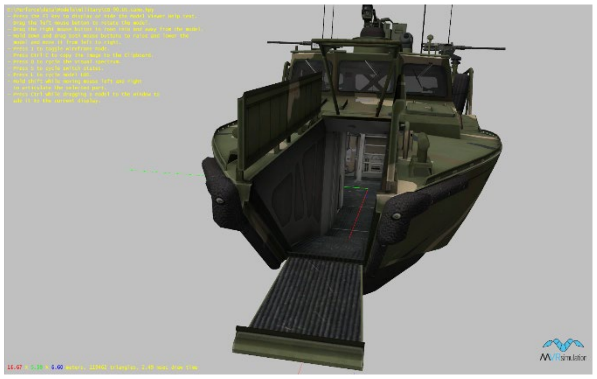
The first switch state in the model hierarchy is the default on the menu. For MVRsimulation models, this will be the damage switch most of the time. The following example shows the damaged state of a Toyota Hilux vehicle:



The next example of a Bell-407 rotary wing model illustrates a model in its normal state and with the Engine switch state turned on, indicated by the whirling propellers:

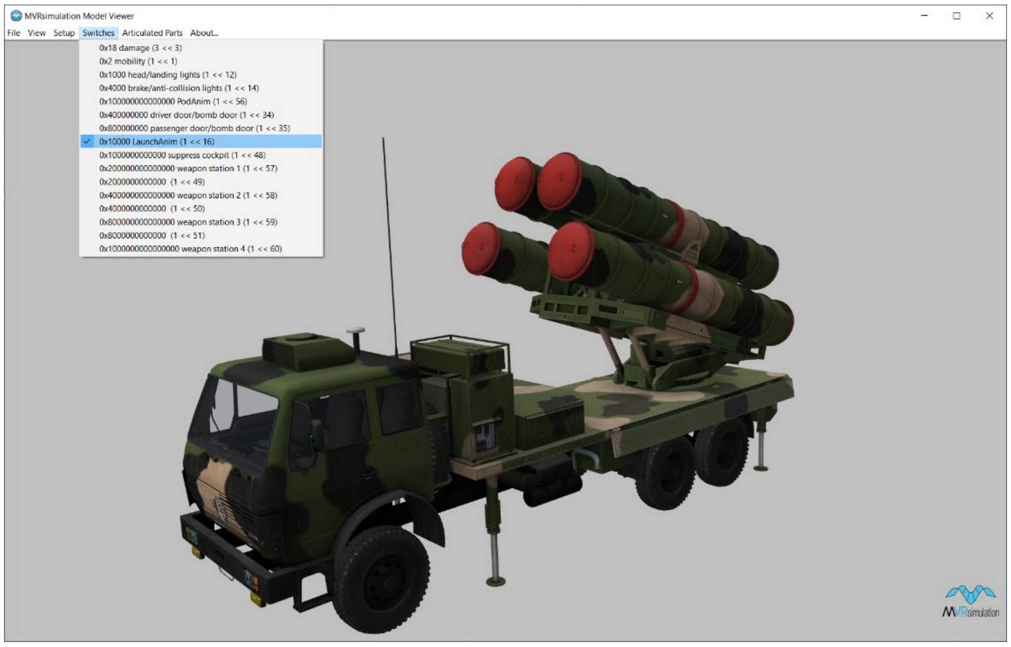


The following example illustrates a CB-90 model in its normal state and with a Hatch switch state turned on, indicated by the open hatch:



Some models, like this one, have multiple hatches that can be opened.

Some models have animations for one or more articulated parts, which can be customized, as described in the section later in this chapter, "Extracting a model's metadata for customization." To preview the animation, choose it on the Switches menu and then press the "S" key, as shown:


MVRsimulation Model Viewer
File
View
Setup
Switches
Articulated Parts
About...
0x18 damage (3 << 3)
0x2 mobility (1 << 1)
0x1000 head/landing lights (1 << 12)
0x4000 brake/anti-collision lights (1 << 14)
0x100000000000000 PodAnim (1 << 56)
0x400000000 driver door/bomb door (1 << 34)
0x800000000 passenger door/bomb door (1 << 35)
✓ 0x10000 LaunchAnim (1 << 16)
0x1000000000000 suppress cockpit (1 << 48)
0x200000000000000 weapon station 1 (1 << 57)
0x2000000000000 (1 << 49)
0x400000000000000 weapon station 2 (1 << 58)
0x4000000000000 (1 << 50)
0x800000000000000 weapon station 3 (1 << 59)
0x8000000000000 (1 << 51)
0x1000000000000000 weapon station 4 (1 << 60)
MVRsimulation

In the example above with the DK-10.MA model, the Launcher animation is selected, and pressing the S key plays the animation of raising and lowering the launcher. (The Pod animation had been played beforehand, hence the articulating ground support pods are already lowered in this example.)

Displaying a culture model's damage state

VRSG's culture model libraries include over 300 models that have the standard slight, moderate, and destroyed damage states. MVRsimulation is standardizing on building damage states into its culture models. In addition to using these models to depict destruction in scenarios, you can use your own models that have damage appearances tagged with DIS appearance bits.

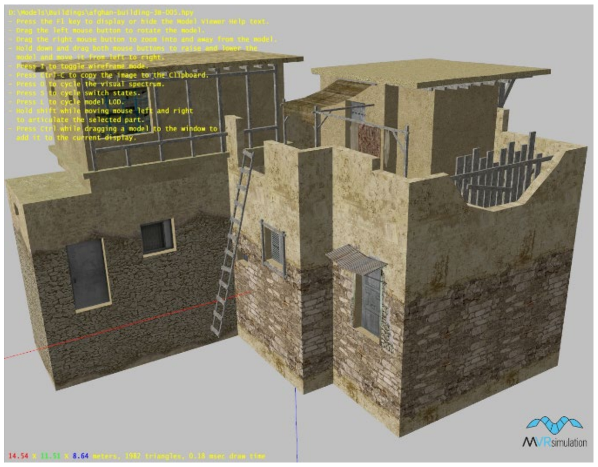
The culture models that currently contain damage appearance states are used in demo scenarios that take place on the virtual Afghanistan, Kismayo, and Hajin, Syria, 3D terrains that are delivered with VRSG. In these scenarios, you can see the resulting damage states from air strikes.

The culture models that currently contain damage appearance states are:

- Afghan-building-3B-001 to Afghan-building-3B-154.
- Kismayo-building-0008, Kismayo-building-0088, Kismayo-building-0107, Kismayo-building-0115, Kismayo-building-0130, Kismayo-building-0445, Kismayo-building-0466, Kismayo-building-0505, and Kismayo-building-0506.

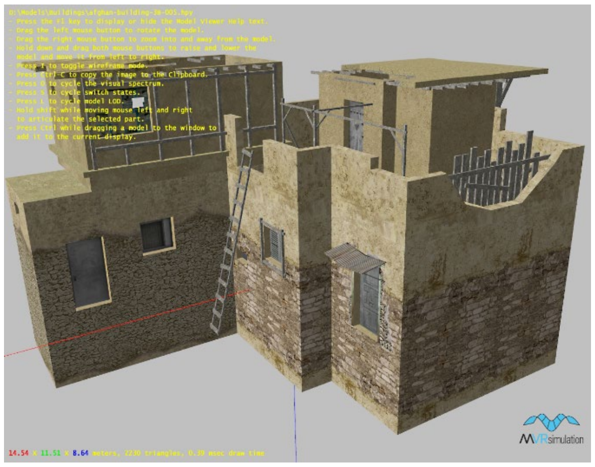
- All building models with a name prefixed “Hajin-Building-”
- Tree-cassia-001, Tree-cassia-005, Tree-elm-002, Tree-elm-006, Tree-poplar-004. These tree models become defoliated when they are fully damaged. These models also have a different appearance for each of the 4 seasons, as described next.

Like previewing the damage states of vehicles in the Model Viewer, to preview the damage states of a culture model (that has damage states), choose the Damage switch state of the Model Viewer and press the “S” key on the keyboard to cycle through the 3 damage states, as shown in this Afghanistan building model:



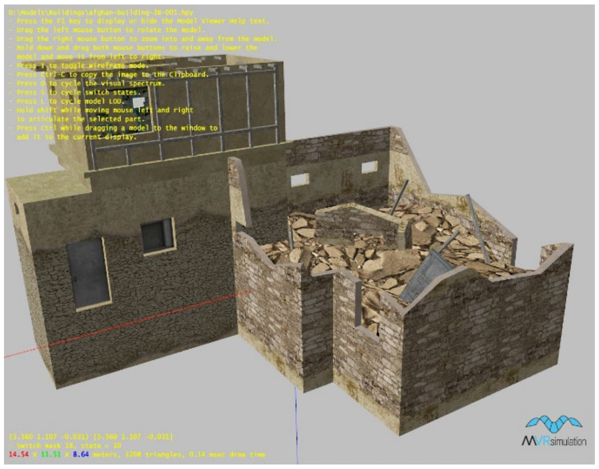
```
D:\Models\Buildings\afghan-building-38-005.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

14.54 x 11.51 x 8.64 meters, 1982 triangles, 0.18 msec draw time
```

Normal state, no damage.



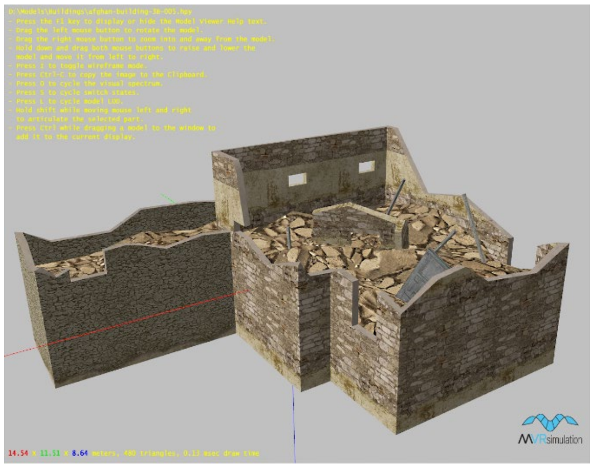
```
D:\Models\Buildings\afghan-building-38-005.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

14.54 x 11.51 x 8.64 meters, 2230 triangles, 0.39 msec draw time
```

Slight damage (affecting the roof and upper story).



```
D:\Models\Buildings\afghan-building-38-005.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

(5.560 1.107 -0.031) [5.560 1.107 -0.031]
- Switch mask 18, state = 10
14.54 x 11.51 x 8.64 meters, 1208 triangles, 0.14 msec draw time
```

Moderate damage.



```
D:\Models\Buildings\afghan-building-38-005.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

14.54 x 11.51 x 8.64 meters, 480 triangles, 0.13 msec draw time
```

Total damage (destroyed).

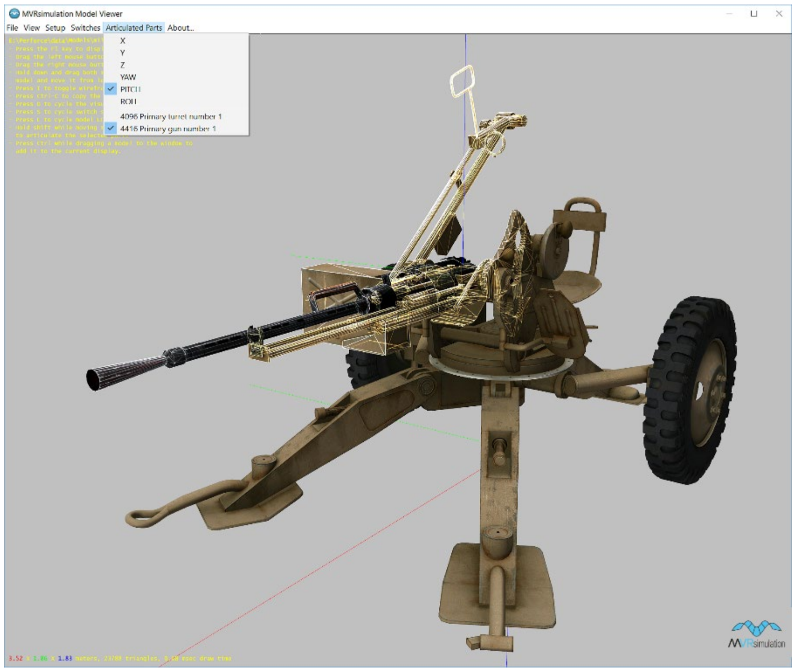
Displaying the season states in tree models

A set of 7 trees in MVRsimulation’s library of tree models have multiple season states. The models: tree-163, tree-164, tree-cassia-001, tree-cassia-005, tree -elm-002, tree -elm-006, tree -poplar-004 all have 4 season states. The season variants of tree-cassia-005 are shown below:

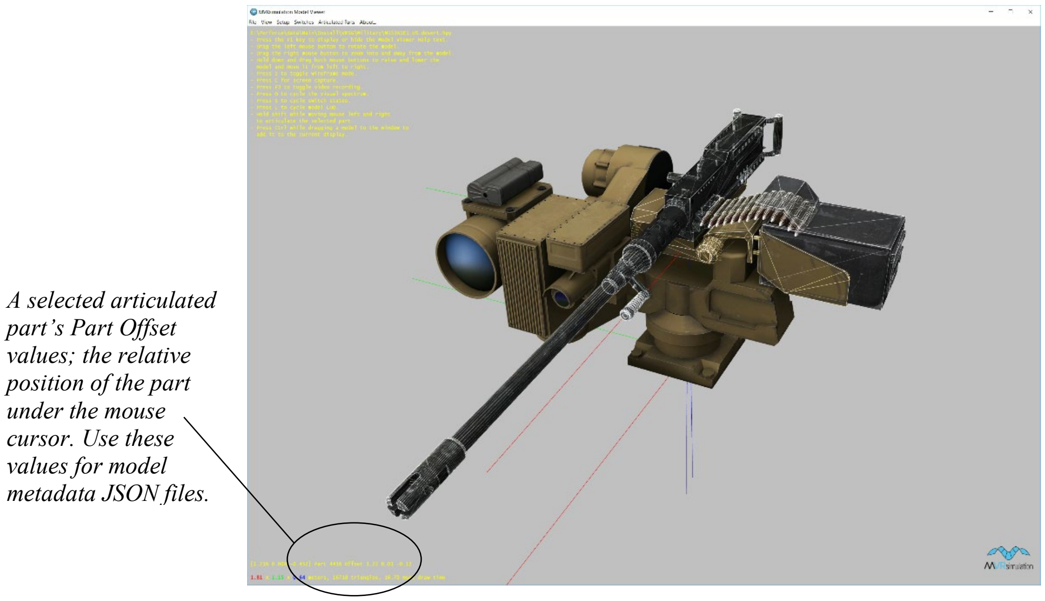
In the Model Viewer, cycle through the tree model's season variants as you would cycle through a switch state, by pressing the "S" key.

Displaying a vehicle model's articulated parts

Many vehicle models have movable parts, such as doors, guns, turrets, camera gimbal, radars, wheels, and so on. These parts will be listed in the Model Viewer's Articulated Parts menu. You can move a movable part by selecting in the menu the part you want to move and the axis or direction in which you want to move it, as shown in the following ZPU-1 model example. Then, move the part by holding down the Shift key and dragging the part with the mouse, moving the cursor left and right. To remove the highlighting on the part, deselect it from the Articulated Parts menu.


MVRsimulation Model Viewer
File
View
Setup
Switches
Articulated Parts
About..
X
Y
Z
YAW
✓ PITCH
ROLL
4096 Primary turret number 1
✓ 4416 Primary gun number 1
MVRsimulation

For a given model, the Model Viewer shows (on the lower left part of the display) a selected articulated part's relative position under the mouse cursor. You can use this "Part Offset" number to specify changes to a model attribute specified in a JSON file (such as where muzzle blast particle effects are attached to a model, or details about an articulated part's animation).

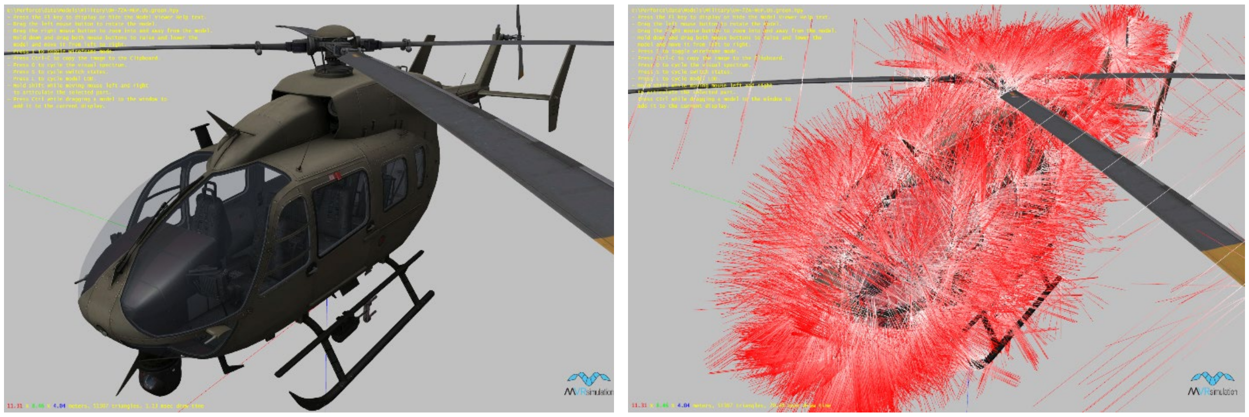

A selected articulated part's Part Offset values; the relative position of the part under the mouse cursor. Use these values for model metadata JSON files.

For more information about using model metadata JSON files, see the chapter "Configuring Models and Events."

Displaying a model's vertex normals

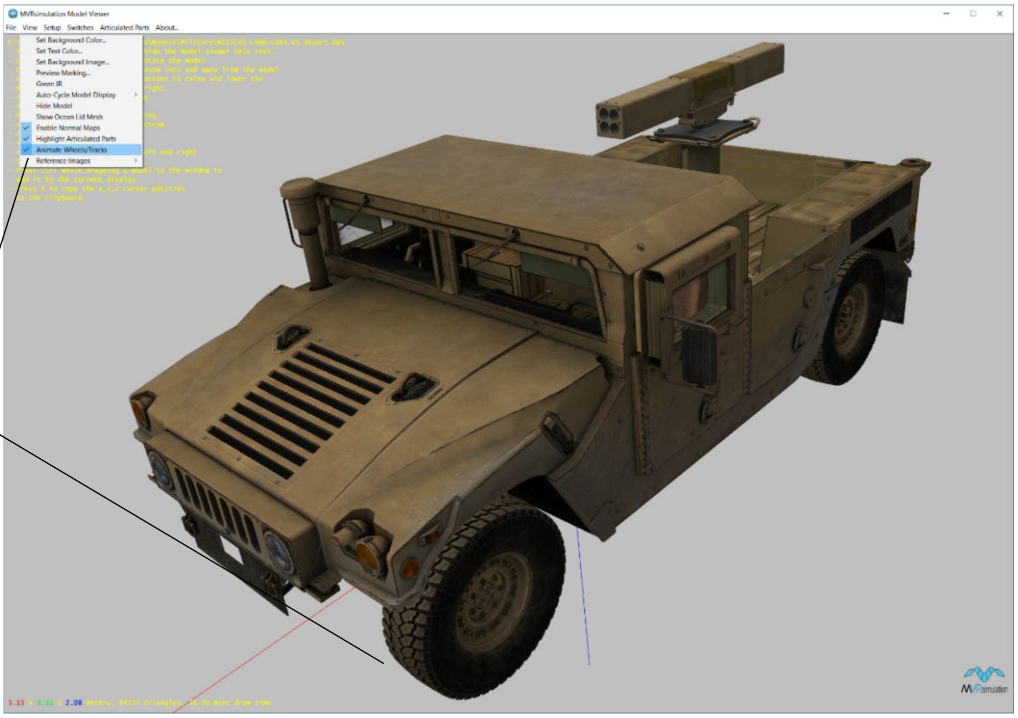
To inspect a model's vertex normals, press keyboard N.

The display is useful for determining whether the normals are oriented correctly, as shown in the following UH-72_MEP model.



Previewing a vehicle model's wheel or track animation

By default, wheels and tracks are stationary in the Model Viewer. To preview a vehicle model's moving wheel or track animation, choose View > Animate Wheels/Tracks.



Select the Animate Wheels//Tracks checkbox to preview the model's wheels or tracks in motion.

Saving a model's switch state, articulated part position, and IR hot spots

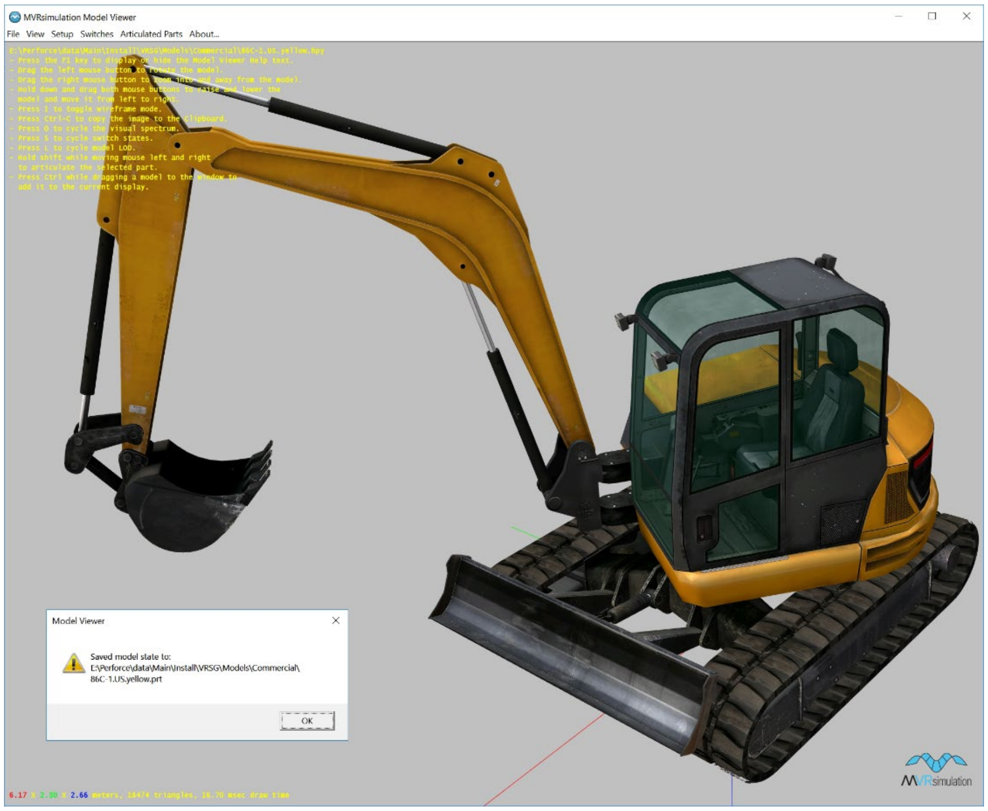
To save a model's switch state, articulated part position and specific modulated IR hot spots for subsequent use in VRSG, you can output these settings to an external file that VRSG loads with model, or generate a text string of the commands of these settings to paste into a cultural feature file.

To save the settings in an external file that VRSG loads with the model:

1. Open a model in the Model Viewer.
2. Change the display to the switch state you want, move the articulated part to the position you want, and initialize or minimize specific IR hot spots.
3. Choose File > Save Model State.

This action saves the switch state, position of the articulated parts, and IR hot spot values to a file, and retains these settings for each instance of the model across VRSG sessions. The Model Viewer creates a file called *model-name*.prt in the same directory as the model. The PRT file contains the metadata that overrides the default values. You can associate multiple PRT files with a model to have multiple saved model states, and associate them in a cultural feature file (CLT) to load a specific PRT file for a model in a scene.

In the following example, the saved .PRT file has captured the switch state of the excavator's boom, and the articulated position of the cab:


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
OK
MVRsimulation

```
E:\Perforce\data\Main\Install\VRSG\Models\Commercial\86C-1.US.yellow.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press 1 to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
```

Model Viewer

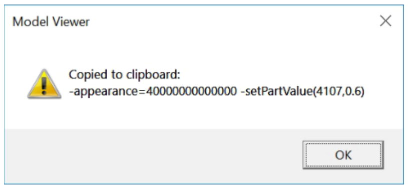
Saved model state to:
E:\Perforce\data\Main\Install\VRSG\Models\Commercial\
86C-1.US.yellow.prt

```
6.17 X 2.80 X 2.66 meters, 18474 triangles, 16.70 msec draw time
```

Model Viewer can generate a command string of a loaded model's switch state, articulated parts positions, BVH animation, and specific IR hot spots for pasting in the model's entry in a cultural feature file. This feature is useful when you want to use these settings for just one or a select few instances of a given model, but not for all instances of the model rendered in VRSG.

To generate a command string that describes the loaded model's switch state, articulated parts position and IR hot spot intensities:

1. Open a model in the Model Viewer.
2. Change the display to the switch state you want, move the articulated part to the position you want, and initialize or minimize specific IR hot spots.
3. Choose File > Copy CLT commands to Clipboard. A message appears, displaying the command string that is copied to the Windows Clipboard, as shown in the following example generated for the excavator model shown above.



Model Viewer

Copied to clipboard:
-appearance=40000000000000 -setPartValue(4107,0.6)

4. Paste the saved command string in the cultural feature file entry for the model.

```
-appearance=40000000000000 -setPartValue(4107,0.6)
```

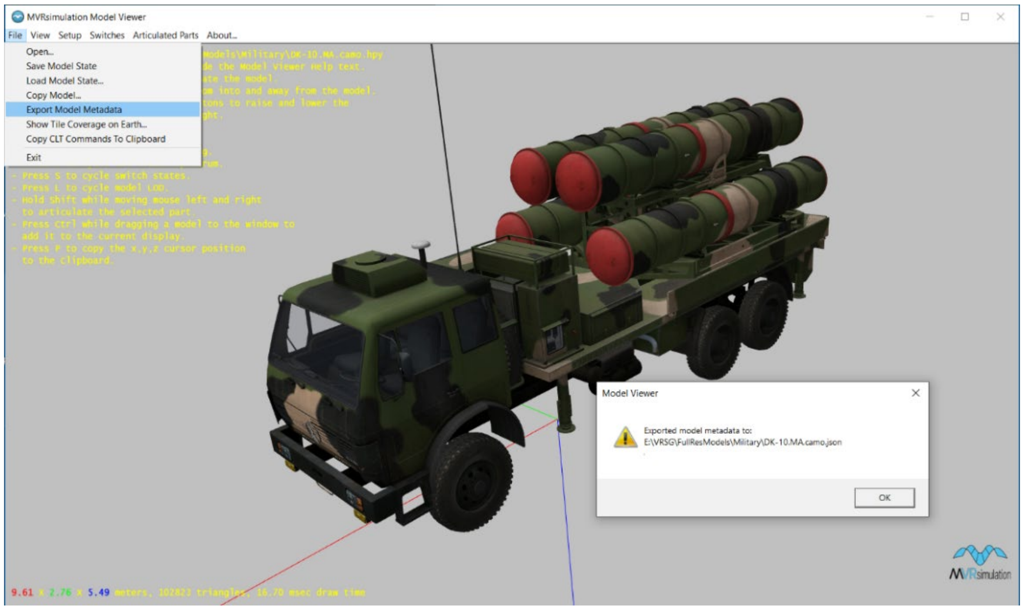
Note: You can use the `-setPartValue` command in the ModelMap.ini file or in a CLT file to specify the initial state of an entity's articulated part before the simulation controls the part (or in case the simulation does not control the part).

For more information about cultural feature files, see the chapter "Configuring Models and Events."

Extracting a model's metadata for customization

In recent models in MVRsimulation military model library, model metadata resides in a JSON file that is embedded in the HPY model. Extracting this metadata is useful for customizing the built-in animations of a model's articulated parts.

To extract the metadata from an HPY model, choose File > Export Model Metadata. This action extracts a JSON metadata file outside the model, which has the same name of the model and a .json extension, as shown in the DK-10.MA example below:



```
Models\Military\DK-10.MA.camo.hpy
de the Model Viewer help text.
ate the model.
om into and away from the model.
tons to raise and lower the
ght.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the clipboard.
```

Exported model metadata to:
E:\VRSG\FullResModels\Military\DK-10.MA.camo.json

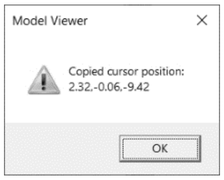
Customizing a model by editing its JSON file is described in the chapter, "Configuring Models and Events." If a JSON file exists inside the HPY model and one also outside as a separate file, the separate file will override the embedded JSON file.

If a model does not have any metadata (meaning it does not have any animations for its articulated parts), the menu item Export Model Metadata will not be available. In this case, you can create your own JSON metadata file for customizing a model's articulated parts, as described in the chapter, "Configuring Models and Events."

Obtaining a specific x,y,z position on a model

To obtain the x, y, and z offset position of a specific location on a model:

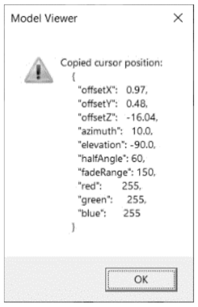
1. Open the model of interest in Model Viewer, and place the cursor over the exact location on the model for which you need the coordinates.
2. Press the letter "P" key on the keyboard. This action copies the x, y, z coordinates of the cursor position to the Windows Clipboard.
3. Paste the x, y, and z values into the file you need, such as, ModelMap.ini or vrsg.clt. This feature is used for obtaining locations of attachment points on a model.



```
Copied cursor position:
2.32,-0.06,-9.42
```

To obtain the x, y, and z offset position and other values needed for positioning a light lobe on a model:

1. Open the model of interest in Model Viewer, and place the cursor over the exact location on the model.
2. Press Shift P on the keyboard. This action copies to the Windows Clipboard not only the x, y, z coordinates of the cursor position, but also the orientation and lobe color attributes.
3. Paste the values from the Clipboard into the model's JSON file.

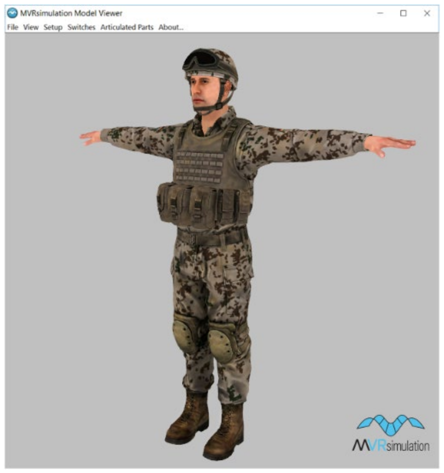


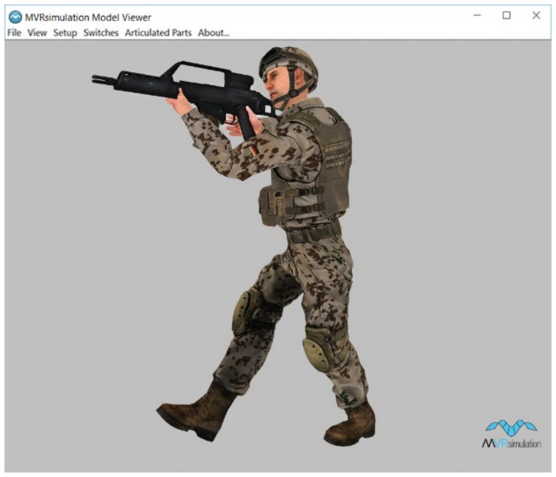
```
Copied cursor position:
  {
    "offsetX":   0.97,
    "offsetY":   0.48,
    "offsetZ":   -16.04,
    "azimuth":   10.0,
    "elevation": -90.0,
    "halfAngle": 60,
    "fadeRange": 150,
    "red":       255,
    "green":     255,
    "blue":      255
  }
```

Previewing characters with animations and weapons

VRSG is delivered with a large set of animated character models and optional weapon and other accessory models. You can use the Model Viewer to preview various combinations of characters with animations and weapons.

To visualize a character model (human or animal) in the Model Viewer, simply double-click the model file, or drag the model file to the Model Viewer window as described earlier. The human characters in VRSG's character model library typically begin with the prefix "human" for example "human-business-001.hpy."


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

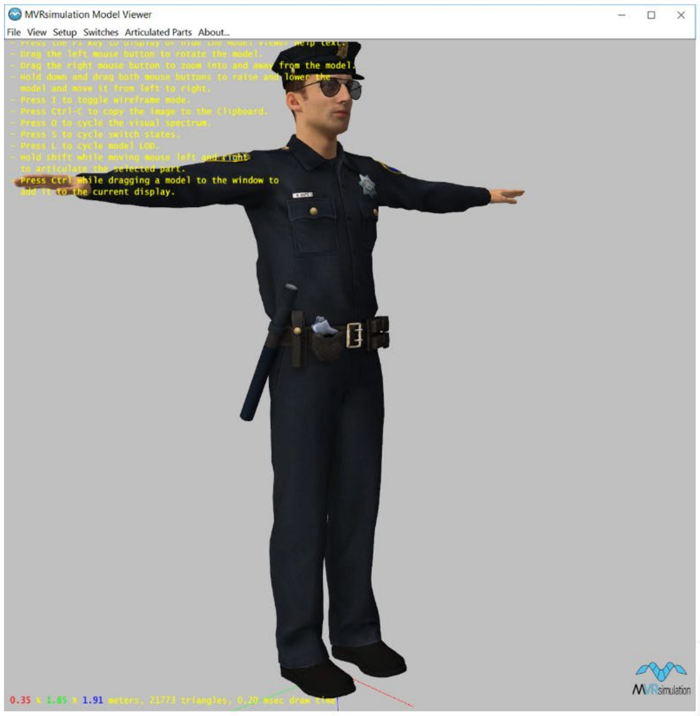
To test a character model with animations and weapons or accessories:

1. Open the character model in the MVRsimulation Model Viewer.
2. Optionally, preview a weapon or other accessory model with the character; simply drag a weapon model (or accessory model, which also has the prefix "weapon-" in the model name) from MVRsimulation's 3D character library to the Model Viewer window. The weapon or accessory model will automatically be added as a secondary model and the Model Viewer will treat it as the character's weapon.
3. Test an animation with your character (with or without a weapon or accessory) by dragging one of the BVH character animations in the \MVRsimulation\VRSG\Animations directory to the Model Viewer window.

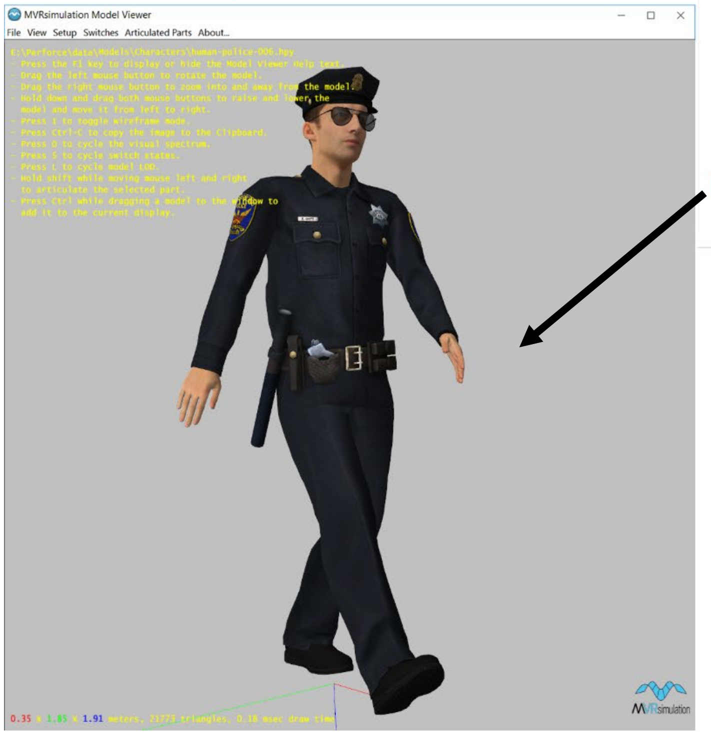
To inspect a weapon or accessory with the character model displayed in the Model Viewer, drag a weapon or accessory model to the Model Viewer window, while the character is displayed. A model name that begins with the "weapon-" prefix is automatically added as a secondary model and Model Viewer treats it as the character's weapon.

If the weapon model does not begin with the "weapon-" prefix in the filename, you can visualize the weapon with the character by opening it through the File menu.

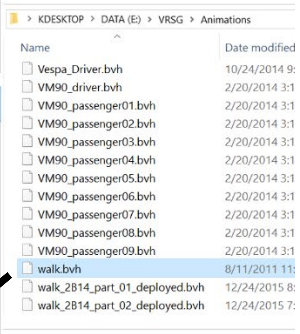
The following example illustrates dragging a BVH animation clip to the Model Viewer to animate the displayed human-police-006 character model. MVRsimulation character animations are located in the \MVRsimulation\VRSG\Animations directory.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

```
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press 1 to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

0.35 x 1.85 x 1.91 meters, 21773 triangles, 0.20 msec draw time
```


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

```
E:\Perforce\data\Models\Characters\human-police-006.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press 1 to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

0.35 x 1.85 x 1.91 meters, 21773 triangles, 0.18 msec draw time
```



Name	Date modified
Vespa_Driver.bvh	10/24/2014 9:
VM90_driver.bvh	2/20/2014 3:1
VM90_passenger01.bvh	2/20/2014 3:1
VM90_passenger02.bvh	2/20/2014 3:1
VM90_passenger03.bvh	2/20/2014 3:1
VM90_passenger04.bvh	2/20/2014 3:1
VM90_passenger05.bvh	2/20/2014 3:1
VM90_passenger06.bvh	2/20/2014 3:1
VM90_passenger07.bvh	2/20/2014 3:1
VM90_passenger08.bvh	2/20/2014 3:1
VM90_passenger09.bvh	2/20/2014 3:1
walk.bvh	8/11/2011 11:
walk_2B14_part_01_deployed.bvh	12/24/2015 8:
walk_2B14_part_02_deployed.bvh	12/24/2015 7:

While viewing a particular character, you can change the animation (by dragging it to the Model Viewer window as described earlier) or change the weapon (also by dragging it to the Model Viewer window) without starting over with the character model.

For more information about characters and animations, see the chapter "Using 3D Characters in VRSG."

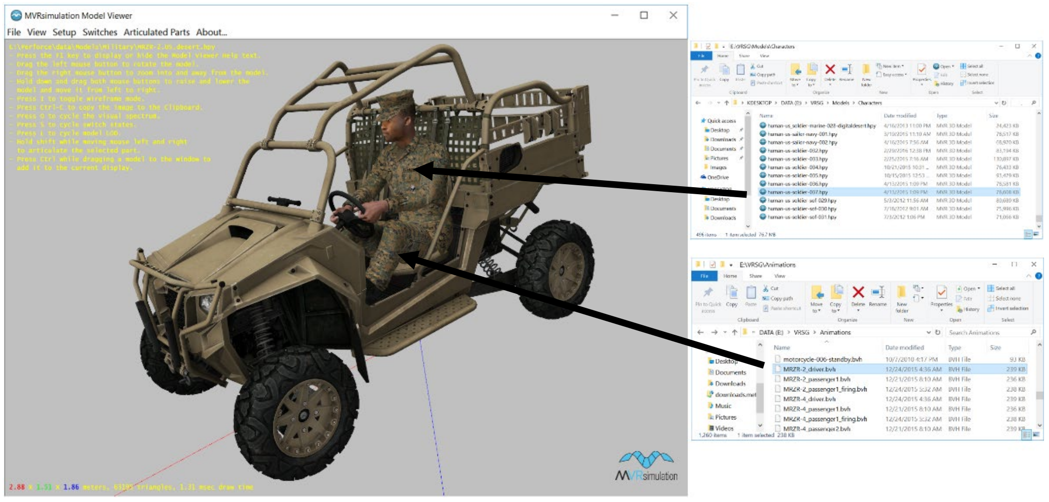
Displaying multiple models

You can examine multiple models together in the Model Viewer, for example to test a driver character animation within a vehicle model, a character holding a weapon, or the size of a model with respect to another one.

To display a model along with one already displayed in the Model Viewer:

1. Open the Model Viewer and display the first model of interest.
2. Press and hold the Ctrl key while dragging the second model from the Windows Explorer to the Model Viewer window.

In the following example, a soldier character was dragged to the Model Viewer while the vehicle model was already displayed. The character model was added in order to inspect the driver animation, which was also dragged to the Model Viewer.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...

```
E:\Perforce\data\Models\Military\MRZR-2.US.desert.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
```

E:\VRSG\Models\Characters — KDESKTOP > DATA (E:) > VRSG > Models > Characters

Name	Date modified	Type	Size
human-us_soldier-marine-028-digitaldesert.hpy	4/16/2013 11:00 PM	MVR 3D Model	24,423 KB
human-us-sailor-navy-001.hpy	3/10/2015 11:10 AM	MVR 3D Model	76,517 KB
human-us-sailor-navy-002.hpy	4/16/2015 7:56 AM	MVR 3D Model	68,970 KB
human-us-soldier-032.hpy	2/29/2016 12:38 PM	MVR 3D Model	83,194 KB
human-us-soldier-033.hpy	2/25/2015 7:16 AM	MVR 3D Model	130,897 KB
human-us-soldier-034.hpy	10/21/2015 10:31 ...	MVR 3D Model	76,433 KB
human-us-soldier-035.hpy	10/15/2015 12:53 ...	MVR 3D Model	93,479 KB
human-us-soldier-036.hpy	4/13/2015 1:09 PM	MVR 3D Model	76,581 KB
human-us-soldier-037.hpy	4/13/2015 1:09 PM	MVR 3D Model	78,608 KB
human-us-soldier-sof-029.hpy	5/3/2012 11:56 AM	MVR 3D Model	80,689 KB
human-us-soldier-sof-030.hpy	7/16/2012 9:01 AM	MVR 3D Model	75,996 KB
human-us-soldier-sof-031.hpy	7/3/2012 1:06 PM	MVR 3D Model	71,066 KB

456 items 1 item selected 76.7 MB

E:\VRSG\Animations — DATA (E:) > VRSG > Animations

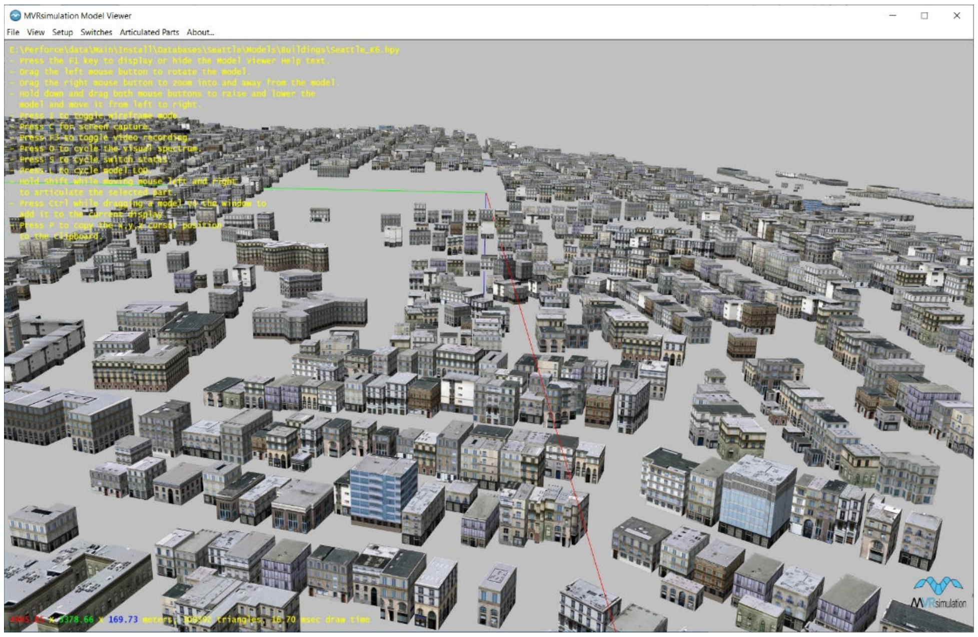
Name	Date modified	Type	Size
motorcycle-006-standby.bvh	10/7/2010 4:17 PM	BVH File	93 KB
MRZR-2_driver.bvh	12/24/2015 4:36 AM	BVH File	239 KB
MRZR-2_passenger1.bvh	12/21/2015 8:10 AM	BVH File	236 KB
MRZR-2_passenger1_firing.bvh	12/24/2015 5:32 AM	BVH File	238 KB
MRZR-4_driver.bvh	12/24/2015 4:36 AM	BVH File	239 KB
MRZR-4_passenger1.bvh	12/21/2015 8:10 AM	BVH File	236 KB
MRZR-4_passenger1_firing.bvh	12/24/2015 5:32 AM	BVH File	238 KB
MRZR-4_passenger2.bvh	12/21/2015 8:10 AM	BVH File	239 KB

1,260 items 1 item selected 238 KB

Alternatively, you can add another model to the Model Viewer by displaying the first model and then choosing File > Open.

When the Browse dialog box appears, locate and select the intended weapon model, click the checkbox "Add this model to the current display" and then click Open.

Models that are a composite of many models can also be viewed in the Model Viewer, as shown in the example below. The example is of a large model originally generated in CityEngine (then output as an FBX model and converted to MVRsimulation's model format) comprised of hundreds of Seattle building models.



```
E:\Perforce\data\Main\Install\Databases\Seattle\Models\Buildings\Seattle_K6.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press 1 to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the clipboard.
[illegible] x 3378.66 x 169.73 meters, [illegible] triangles, 16.70 msec draw time
```

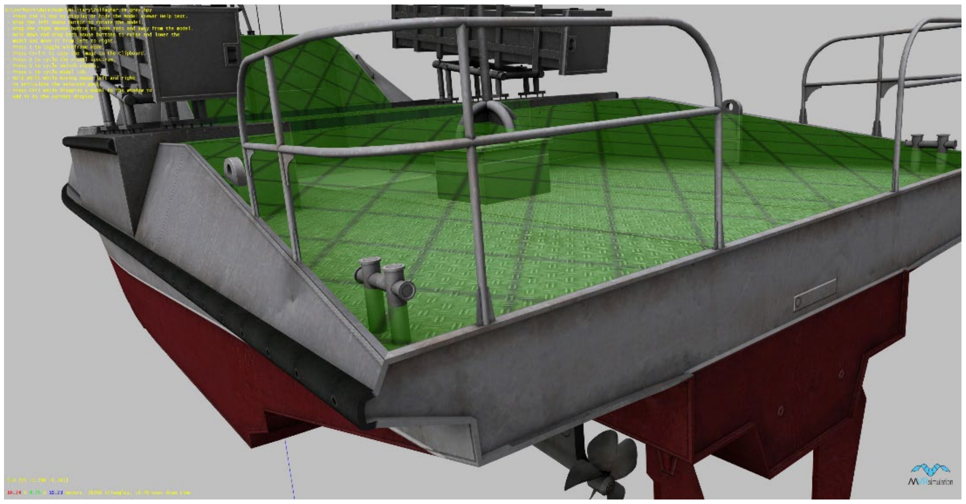
Previewing a vessel model's ocean mask

You can preview the hidden mask or "lid" polygon on top of a surface vessel (domain 3 model) that prevents the 3D ocean from rendering inside the model's open area. Many of MVRsimulation's vessel models were built with this feature.

To preview the ocean lid for a vessel model in the Model Viewer:

Choose View > Show Ocean Lid Mesh.

The lid appears as green translucent area with a grid, as shown:



The lid extends across the open area of the model and masks out the 3D ocean lower than the height of the lid.

Previewing a weapon's firing effect

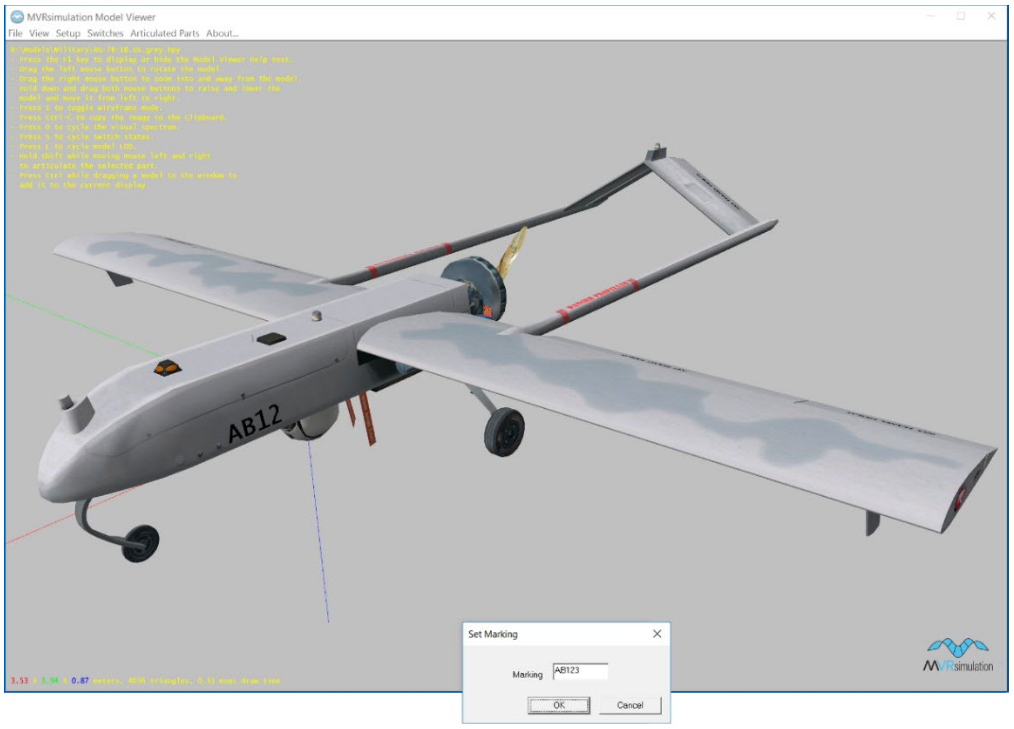
To preview a weapon's firing effect, load the weapon model in the Model Viewer and press the "F" key on the keyboard.

In the same way, press the F key on the keyboard to preview the firing effect of a weapon mounted on a vehicle, as shown:

Previewing a model's markings

Some models in MVRsimulation's military vehicle model library include polygons that can accept textual markings. These models include: T-72-M1.RU.desert-camo.hpy, T-72-

M4.CZ.camo.hpy, T-90S.IN.camo.hpy, T-90S.RU.desert-camo.hpy, and RQ-7B-1B.US.grey.hpy. Although a textual entity marking comes from the marking field in the DIS PDU, you can preview marking changes in the Model Viewer.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
AB12
Set Marking
OK
Cancel
MVRsimulation

```
D:\Models\Military\RQ-7B-1B.US.grey.hpy
- Press the F1 key to display or hide the Model Viewer help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press 1 to toggle wireframe mode.
- Press Ctrl-C to copy the image to the clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

3.53 x 3.94 x 0.87 meters, 4036 triangles, 0.31 msec draw time
```

Marking: AB123

To preview the markings for a model in the Model Viewer:

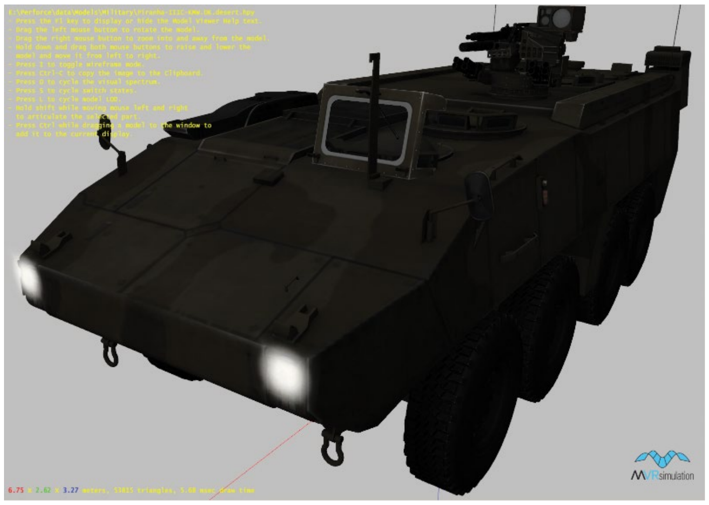
1. Choose View > Preview Marking.
2. In the Set Marking dialog box, type the text you want to preview, and click OK.

The marking text you typed will be previewed in the marking area(s) of the model.

If you need to use markings on other models in the MVRsimulation military vehicle library, contact MVRsimulation at support@mvrsimulation.com to request a modification to the model of interest so that it can accept textual markings (on its side, or tail, or both).

Previewing a model's lights as they would appear in a night scene

To display the emissive properties of a model in night lighting, turn on the headlights, break lights, or landing lights by selecting them the Switches menu and then press Shift-N. The Shift-N action turns the model dark, as it would appear at night, and displays the intensity of the lights as they would appears in a nighttime scene as shown next.



```
E:\Perforce\data\Models\Military\Piranha-IIIC-KMW.DE.desert.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
```

```
6.75 x 2.62 x 3.27 meters, 53815 triangles, 5.68 msec draw time
```

Roll the mouse wheel to modulate the intensity of the lights. Note that you cannot preview light lobes defined for a model via a JSON file.

Changing the background of the Model Viewer window

You can change the background of the Model Viewer window to a different color to improve the visibility of some aspects of a model you are inspecting. You can also change the background image. Loading a background image to a scene is useful for assessing the environment you will eventually use the model in or to inspect the accuracy of a model. Model Viewer provides the ability to associate one or more reference images with a model.

Changing the background color

Some models are easier to inspect against a light color background, others against a dark color. You can change the default grey background color to any color on the Windows color palette.

To change the Model Viewer's background color:

1. Choose View > Set Background Color.
2. From the Windows color palette that appears, select the color you want to use in the background and click OK. You can define a custom color in the palette as you would in the standard color palette in other Windows applications.

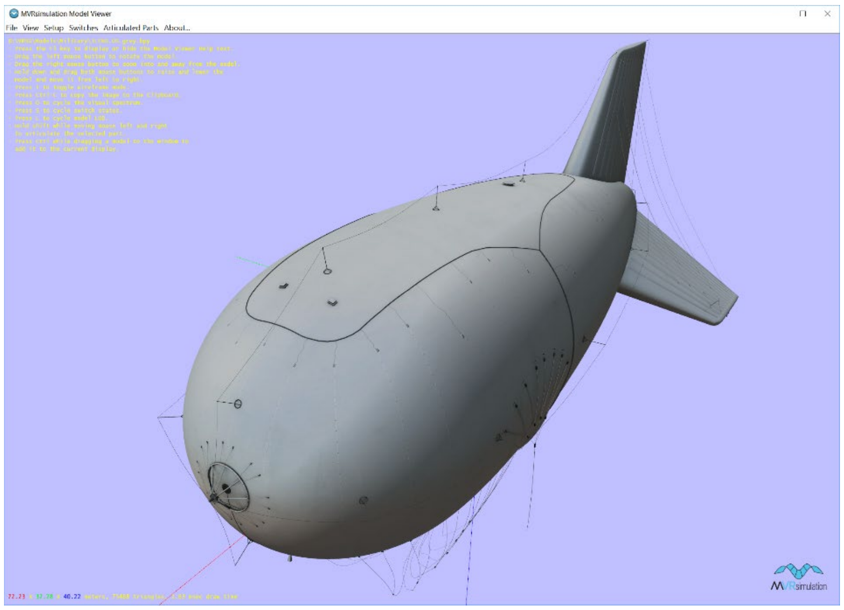
To turn off and on display of the Help text, the model's polygon count and dimensions, and the axes, press F1.

Setting a background image

You can also set an image to use as the Model Viewer's background. Doing so is useful if you are trying to set up the state or articulated part orientation of one or more models in relation to something else.

To set an image to use as the Model Viewer's background:

1. Choose View > Set Background Image.
2. When the Browse dialog box appears, locate the image file you want to use and click Open.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

[illegible]
- Press the F1 key to display or hide the Model Viewer help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the model and move it from left to right.
- Press T to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right to articulate the selected part.
- Press Ctrl while dragging a model to the window to add it to the current display.

The image file can be any file format that VRSG supports: JPG, BMP, TIF, and so on. See the beginning of the chapter, "Manipulate Textures" for a list of image files that VRSG supports.

The image fills the Model Viewer window, behind the model. This option is useful for examining a model you have created against a backdrop of a scene in which it will reside.

If you need to examine the background image without the model displayed in front of it, choose View > Hide Model to toggle displaying and hiding the model.

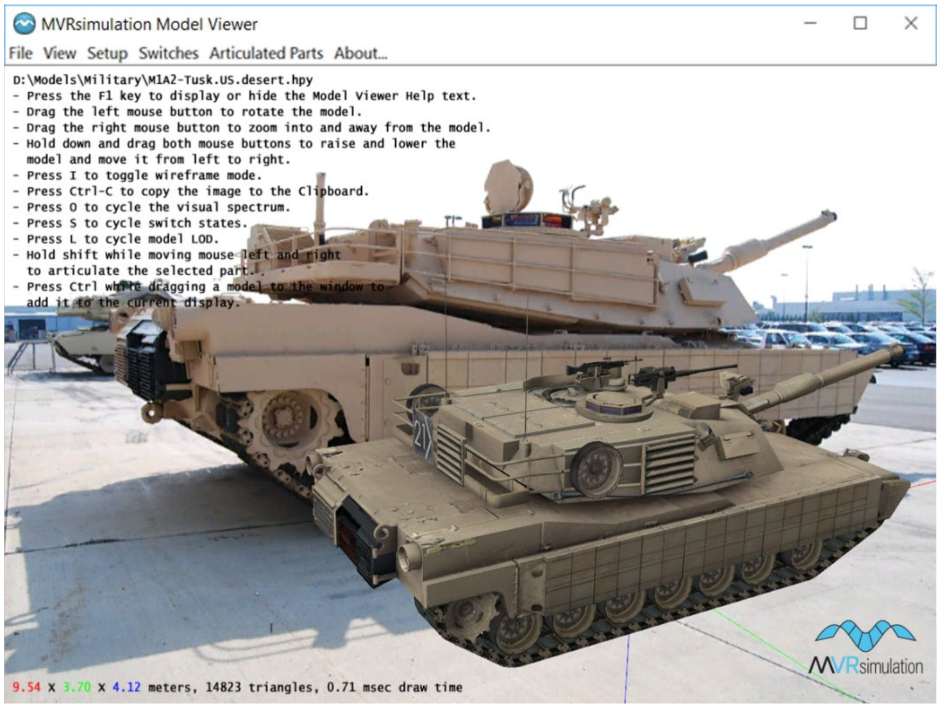
Loading a reference image as a background image

Another use of a background image is the ability to load a reference image (such as a photograph or a CAD diagram) associated with a given model. You can associate multiple images with a model and display each as a background.

This feature can be used as:

- A teaching aid for military vehicle identification for combat recognition.
- An aid for verifying the accuracy of a geospecific model.

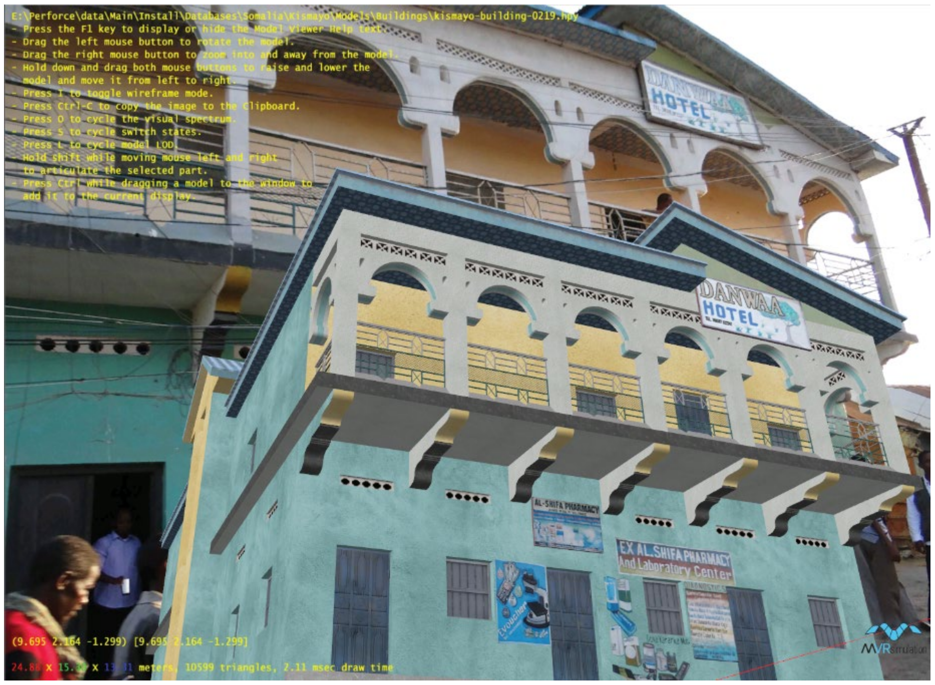
The ability to load reference images for military vehicle models can be used to instruct trainees how to accurately identify and distinguish between different nationalities of military vehicles, as is vital that in any military setting that an operator accurately identify whether a vehicle or entity is part of a friendly force. In the Model Viewer, the 3D model can be compared to a photograph of an actual vehicle. The following example shows a comparison between the M1A2-Tusk.US.desert.hpy model in MVRsimulation's library of military models with a reference image photograph of the actual vehicle.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
21
MVRsimulation

```
D:\Models\Military\M1A2-Tusk.US.desert.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

9.54 x 3.70 x 4.12 meters, 14823 triangles, 0.71 msec draw time
```

In such a close comparison between a photograph and a model, the model can be examined from any angle and with thermal signatures in various sensor modes. By zooming and rotating the model and moving the articulated parts, trainees can learn to recognize a vehicle from different ranges and perspective angles.

As a general means of verifying quality assurance and accuracy of a geospecific model (such as a model that has been newly built and converted to MVRsimulation's format from an FBX or OpenFlight format model) the 3D model can be compared in the Model Viewer to one or more photographs on which the model is based. The following example shows the MVRsimulation model kismayo-building-0219.hpy loaded in Model Viewer with its reference image:



```
E:\Perforce\data\Main\Install\Databases\Somalia\Kismayo\Models\Buildings\kismayo-building-0219.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

(9.695 2.164 -1.299) [9.695 2.164 -1.299]
24.88 x 15.30 x 13.31 meters, 10599 triangles, 2.11 msec draw time
```

This and other examples of Kismayo, Somalia, building model reference images can be found in \MVRsimulation\VRSG\Terrain\Somalia\Kismayo\Models\Buildings.

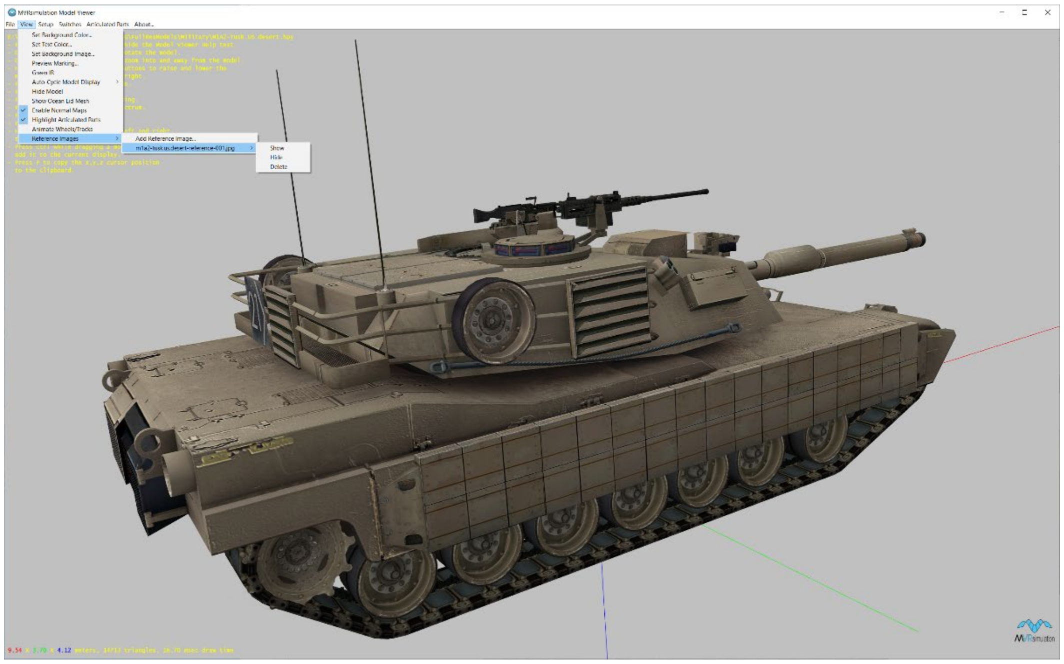
To load a reference image associated with a model as the background image:

1. In the Model Viewer, load the model of interest. If the Model Viewer locates any reference images in the \ReferenceImages subdirectory based on the model's name, a menu option Reference Images appears on the View menu.
2. Choose View > Reference Images > *name-of-reference-image* > Show.
3. Optionally, choose View > Hide Model to examine the background reference image without the model obscuring the image.
4. Optionally, choose View > Reference Images > *name-of-reference-image* > Hide to remove the reference image from the background.
5. Optionally, again choose View > Reference Images > *name-of- reference-image* > Show to load a different reference image associated with the same model.

Model Viewer locates the associated reference image(s) in the \ReferenceImages subdirectory and lists the image filenames in the Reference Image submenu. When you select the image name, Model Viewer loads it as the background image.

To examine the background image without the model displayed in front of it, choose View > Hide Model to toggle displaying and hiding the model.

If you have an image you want to associate with a model to load as reference images in the Model Viewer (such as photographs, illustrations. or CAD diagrams), first resize the image(s) such that the minimum dimension is 1024 pixels on one size, and the maximum dimension is 4096 pixels on one side. An image can be square or rectangular in shape and must be in JPG format.



To associate a reference image with a model:

1. In the Model Viewer, load the model of interest.
2. Choose View > Reference Images > Add Reference Image.
3. When the Add a Reference Image dialog box appears, browse for the JPG image you want to use as a reference image, select it, and then click Open. Model Viewer copies the JPG image to the \ReferenceImages directory associated with the model being viewed and renames the copied image file appropriately. If no \ReferenceImages directory exists, Model Viewer will create one.
4. Repeat steps 2 and 3 for each image you want to associate as a reference image with model loaded in the Model Viewer.
5. When you are finished, load one of the images by choosing View > Reference Images > *name-of-reference-image*> Show.

To delete a reference image, choose View > Reference Images > *name-of- reference-image* > Delete.

Reference images can be deleted through the Model Viewer *if* the image file is not locked as read-only. Because the reference images delivered with VRSG are read-only, the Reference Images menu does not display a Delete option for those images. If you add your own reference images, the Delete option appears on the menu if your images are not read only.

One important aspect in teaching vehicle identification for combat recognition is to show the thermal signatures in the model so that a sensor operator trainee can determine what the vehicle might look like in a thermal white mode from different perspectives and ranges. As described earlier in the chapter, pressing the O key on the keyboard cycles the model through the visual spectrum and enables trainees to preview the simulated thermal signatures.

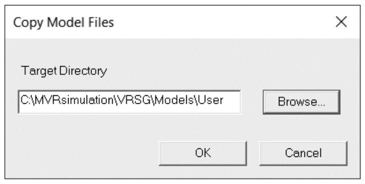
Note: For more advanced combat vehicle identification instruction, you could manipulate a vehicle's articulate part of interest and/or the simulated thermal signatures, save a model's switch state to a PRT file as described earlier in this chapter. Then in VRSG, drag and drop the model onto the terrain and refine the environmental and lighting settings on the Dashboard's Environment tab to bring further context to vehicle identification training. Finally, using these techniques, you could take screen captures of the model in VRSG or in the Model Viewer to add to briefings or course materials.

Copying a model and its textures to another directory

From within the Model Viewer you can copy the displayed model to another directory. Doing so is useful if you are selecting models from MVRsimulation's large model libraries for a particular scenario or another database project and you want to organize them in a directory.

To copy the displayed model to another directory:

1. Choose File > Copy Model.
2. When the Copy Model Files dialog box appears, click Browse to locate and select the directory to which you the model and its textures copied.


Copy Model Files

Target Directory

C:\MVRsimulation\VRSG\Models\User

3. Click OK.

The model file (HPY or HPX) and all of the textures it references are copied to the target directory you specified.

Note: If you create your own models, MVRsimulation recommends you store your own content separately from the content installed with VRSG. Create a subdirectory for your models, in the installed \MVRsimulation\VRSG\Models path called \User, as a location for storing your own models to keep them separate from VRSG's native models: \MVRsimulation\VRSG\Models\User. In a VRSG session, VRSG will search this directory in the same manner it searches its own installed model directories, which means you do not need to explicitly specify this directory in the Alternate Directories list.

Taking screen captures of a model in the Model Viewer

You can take a screen capture of a model that is displayed in the Model Viewer for later review or to use in other applications. Similar to taking screen captures in VRSG, you can have the capture copied to the Windows Clipboard or saved to a JPEG or BMP file in a specified directory. You can then use the image in any application that accepts JPEG or BMP files. This feature is handy for building a reference index of models you have for a particular project or scenario.

There are three ways in which you can take screen captures of models:

- Individual thumbnail image taken in the Model Viewer similar to the thumbnail images that are embedded in the models that are delivered with VRSG.
- Individual larger image capture taken in the Model Viewer.
- A set of images captured in batch mode at the command line.

Creating thumbnail images

For ease of identification, in Model Viewer you can take thumbnail screenshots of models produced or obtained at your site and converted into MVRsimulation's HPX model format. The thumbnail image can either be embedded in the model (in the manner of the image embedded in a model from MVRsimulation's 3D content libraries) or kept in an index of small images of your site-specific models. An image embedded in a model file is displayed in the content palettes of VRSG Scenario Editor and in the contents of a directory when it is viewed in Windows icon mode.

To take a thumbnail image of an HPX model:

1. Open the Model Viewer and display the model, positioned as you want for the thumbnail image.
2. Choose File > Capture HPX Thumbnail.

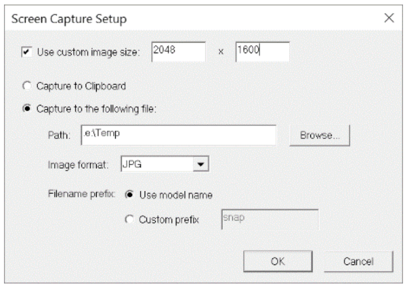
   Model Viewer issues a message asking whether you want to embed the thumbnail image in the model. Embedding the image this way means that you can see the thumbnail image of the model when you look at the model in thumbnail view in Windows Explorer. The image makes it easy to identify the model.
3. Click Yes or No.

If you click Yes, Model Viewer will embed the thumbnail image in the model file and also retain a copy of the unmodified model file. If you click No, Model Viewer will take a small screen capture in .jpg format in the aspect ratio of the Model Viewer window (default image size is 112 x 112 pixels) and saves it in the same directory in which the model is located.

Taking screen captures individually

To set up the manner in which screen captures are handled:

1. Open the Model Viewer and display a model (any model).
2. Choose Setup > Screen Capture Settings. The following dialog box appears:



Screen Capture Setup

☑ Use custom image size: 2048 x 1600

○ Capture to Clipboard

● Capture to the following file:

Path: .e:\Temp

Image format: JPG

Filename prefix: ● Use model name ○ Custom prefix snap

3. Specify whether you want to have captures copied to the Clipboard or to a JPG, BMP, or RGB file in a specified path.
4. If you select Capture to File, browse for the directory path in which the captures should be saved, specify how the captures should be named: with the model name, the default "snap" prefix, or with a prefix you specify, and specify the image format (.jpg, .bmp, or .rgb). Screen captures are numbered in sequence using the specified prefix; in this case, they would be snap.jpg, snap001.jpg, snap002.jpg, and so on.
5. Click OK.

To take a screen capture of a model in the Model Viewer:

1. Open the model in the Model Viewer, and position it the way you want. Optionally, turn off the text display by pressing the F1 key, to ensure the text will not appear in the screen capture.
2. Press the C key on the keyboard. A still image of the area inside the Viewer is copied either to the Windows Clipboard or to a file, depending on the Screen Capture settings.

Taking screen captures in batch mode

You can invoke the Model Viewer from a command-line prompt (or .bat file) to generate a set of screen captures in an automated fashion. Command-line arguments control how the model is displayed and where to direct a screen capture to an external file. When the Model Viewer is used to render screen captures from a batch file, it immediately exits upon completing the image capture. The general Model Viewer command-line syntax is as follows:

```
modelViewer.exe <modelFile> [options…]
```

The Model Viewer command-line options are:

-snapFile=<someFile> specifies a file to save a screen capture too. The format of the image is inferred from the given file extension (.bmp for BMP, .jpg for JPEG).

-IR indicates to take the screen capture in IR white hot mode.

-IRBH indicates to take the screen capture in IR black hot mode.

-EO indicates to take the screen capture in grey-scale (Day TV) mode.

-width=<N>, -height=<N> specifies the dimensions of the screen capture to create. By default, the entire Model Viewer window will be used.

-skyColor=RRGGBB specifies the color of the image background. The values of RR, GG, and BB are hexadecimals in the range 00..FF.

-appearance=<DIS-appearance-mask> specifies the DIS appearance mask to be used to display the model. This is given as a 32-bit hexadecimal with no leading ‘0x’ characters.

Use these arguments to specify an alternate position for the model when viewed.
All values are given in meters. Note the eyepoint is fixed at (0,0,0), so positioning the model moves it away from the eyepoint. By default, Model Viewer will position the model so that the bounding box of the model fits within the field-of-view.

```
-X=<modelXposition>
-Y=<modelYposition>
-Z= <modelZposition>
```

Use these arguments to specify an alternate orientation for a model. All values are given in degrees.

```
-H=<heading>
-P=<pitch>
-R=<roll>
```

-bvh=<animation-file> specifies a BVH animation to apply to a human character mode.

-weapon=<weapon-file> specifies a weapon model to attach to a human character mode.

Capturing video of a model in the Model Viewer

You can record video of models displayed in the Model Viewer for later review or to use in other applications. The video is recorded in H264 MPEG format using the same path and filename as used for screen captures.

This feature is handy for building a reference of models needed for a particular project or scenario. For example, you could set Auto-Cycle Model Display on a set of new models and then record the automatic displays of the models.

To turn on/off recording a model or set of models displayed in the Model Viewer, press the F3 key on the keyboard.

Obtaining information about models

You can invoke the Model Viewer from a command-line prompt (or a BAT file) to generate a report about a model or to automatically report on a directory of models. The information reported includes the bounding box extents of the model in normal ("healthy") and destroyed states, the switch states, articulated parts, the number of IR textures and normal maps, and the IDs of controllable light maps. When the Model Viewer is used at the command line, it immediately exits upon completion.

The Model Viewer command-line syntax is as follows:

```
ModelViewer.exe <modelFile> -stats
```

For example:

```
c:\MVRsimulation\common\bin\modelviewer.exe
c:\MVRsimulation\VRSG\Models\Military \F-16C.US.grey.hpy -stats
```

produces the file:

```
F-16C.US.grey-stats.txt
```

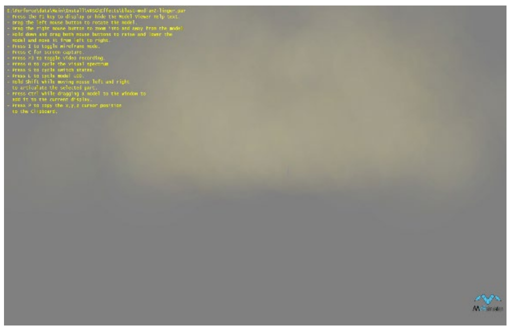
with the following statistics:

```
Maintenance Date:  2010/10/01
Diameter 19.7
Healthy
MinExtents: -7.6 -5.0 -3.6
MaxExtents: 8.6 5.0 1.5
Extents: 16.2 10.0 5.1
Destroyed
MinExtents: -9.8 -4.7 -3.0
MaxExtents: 6.6 5.7 0.2
Extents: 16.4 10.5 3.2
TexturesEO 439995
TexturesIR 18
TexturesNML 0
Switch 1 0 0
Switch 2 18 0 18
Switch 2 10000 0 10000
Switch 2 2000000 0 2000000
LightMaps 1 2 5 6 13
```

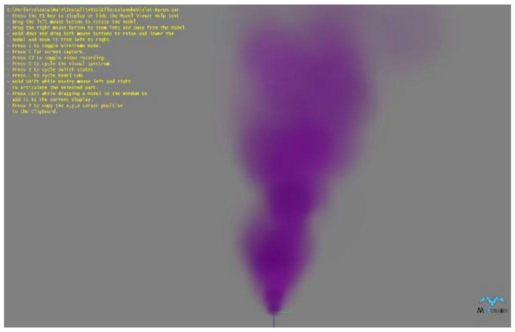
```
ArtParts 1024 1120 1152 1248 1280 1344 1376 3136 3232
ArtPart Part 1024:  pos=(-6.53795,-5e-06,-2.37774)  x=(1e-06,-1,0)
y=(0.557,0,0.831) z=(-0.831,-8.31e-07,0.557)
ArtPart Part 1120:  pos=(-2.69413,-3.16872,-0.129188)  x=(-
0.982,0.188,0) y=(-0.188,-0.982,-0.00259) z=(-0.000487,-0.00255,1)
ArtPart Part 1152:  pos=(-2.69412,3.16906,-0.129184)
x=(0.982,0.188,0) y=(-0.188,0.982,-0.00259) z=(-0.000487,0.00255,1)
ArtPart Part 1248:  pos=(-6.07359,-1.13125,0.019347)
x=(0.0381,0.000232,0.999) y=(-0.0061,1,1e-06) z=(-0.999,-
0.00609,0.0381)
ArtPart Part 1280:  pos=(-5.59002,1.13476,-0.007374)  x=(0.0381,-
0.000239,0.999) y=(0.00628,1,0) z=(-0.999,0.00627,0.0381)
ArtPart Part 1344:  pos=(-0.97079,-3.14078,-0.105366)  x=(-
0.785,0.619,0) y=(-0.619,-0.785,-0.0111) z=(-0.0069,-0.00874,1)
ArtPart Part 1376:  pos=(-0.970791,3.14078,-0.105366)
x=(0.785,0.619,0) y=(-0.619,0.785,-0.0111) z=(-0.0069,0.00874,1)
ArtPart Part 3136:  pos=(-6.08693,0.764231,0.01139)  x=(-1,0,0)
y=(0,-1,0) z=(0,0,1)
ArtPart Part 3136:  pos=(-6.08693,0.764231,0.01139)  x=(1,0,0)
y=(0,1,0) z=(0,0,1)
ArtPart Part 3136:  pos=(-6.08693,-0.760769,0.01139)  x=(1,0,0)
y=(0,1,0) z=(0,0,1)
ArtPart Part 3136:  pos=(-6.08693,-0.760769,0.01139)  x=(-1,0,0)
y=(0,-1,0) z=(0,0,1)
ArtPart Part 3232:  pos=(2.98196,-0.568158,-0.674741)  x=(1,0,0)
y=(0,1,0) z=(0,0,1)
```

Previewing effects

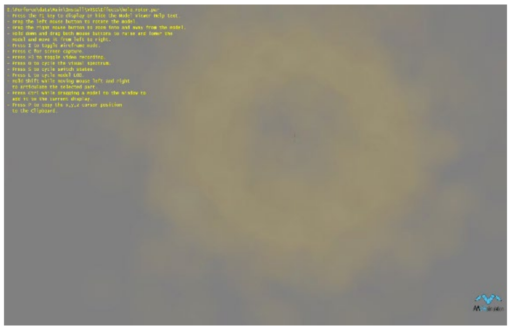
You can preview the effects of PAR, EFF, and CLD files by opening them in MVRsimulation's Model Viewer. The following examples show a rotor wash effect, two smoke effects and a cloud effect previewed in the Model Viewer:



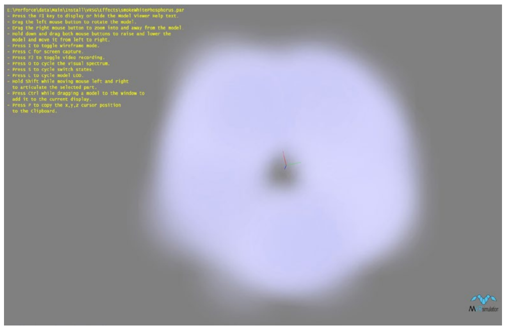
```
E:\Perforce\data\Main\Install\VRSG\Effects\blast-medium2-linger.par
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the Clipboard.
```



```
E:\Perforce\data\Main\Install\VRSG\Effects\smokeViolet-Karman.par
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the Clipboard.
```



```
E:\Perforce\data\Main\Install\VRSG\Effects\helo.rotor.par
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the Clipboard.
```



```
E:\Perforce\data\Main\Install\VRSG\Effects\smokeWhitePhosphorus.par
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the Clipboard.
```

For information about these effects, see the chapter "Configuring Models and Effects."

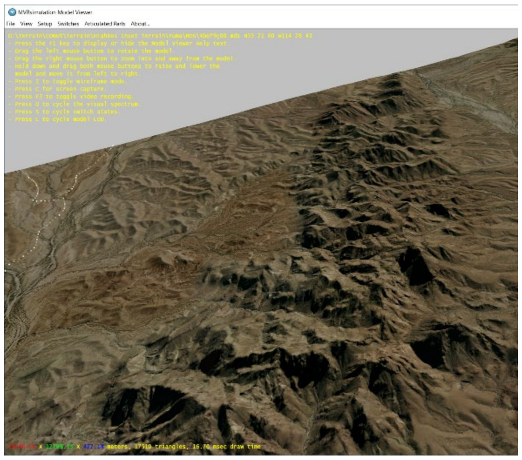
Although previewing effects in the Model Viewer will show how the effect appears and dissipates, it does not give any performance related information.

If your site is developing its own custom effects, MVRsimulation strongly recommends you preview these effects directly in VRSG, by dragging the .par file from Windows File Explorer and dropping it on the rendered VRSG scene. Doing so will help validate the frames-per-second performance of the effect and the duration that was set for the effect.

Note: The flames.flm effect and the solid particle (SPF) effects cannot be previewed in the Model Viewer. You can preview these effects in VRSG by dragging an effects file from Windows Explorer and dropping it in the VRSG visualization window.

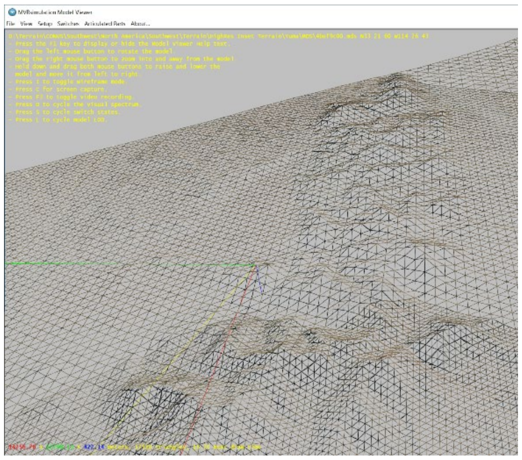
Examining VRSG terrain tiles in the Model Viewer

You can use the Model Viewer to examine VRSG terrain tiles built with MVRsimulation's Terrain Tools. You can inspect both a single tile's textures and geometry, and the earth coverage of a directory of tiles.



MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...

```
D:\terrain\COMMON\terrain\highres inset terrain\Yuma\MDS\40ef9c00.mds n33 21 60 w114 26 43
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
```



MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...

```
D:\Terrain\COMMON\Southwest\North America\Southwest\Terrain\highres inset terrain\Yuma\MDS\40ef9c00.mds n33 21 60 w114 26 43
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
```

Inspecting a terrain tile's texture and geometry

To inspect an MDS terrain tile in the Model Viewer:

1. From the Windows Start menu, choose MVRsimulation > Model Viewer.

2. When the Model Viewer appears choose File > Open and then from the Files of Type drop-down list, select MVRsimulation VRSG Tiles (MDS).
3. Browse for the MDS terrain tile you want to open and click OK.

Alternatively, you can open the Model Viewer and drag an MDS tile file from the Windows Explorer to the Model Viewer. Simply double-clicking the MDS file also opens it in the Model Viewer.

Upon opening the file, the Model Viewer detects the geographic coordinates encoded in the tile's filename and displays the coordinates along with other information about the tile.

You can pan, zoom, and rotate the tile to inspect its textures and geometry as you would any other object in the Model Viewer. You can use the many viewing options by pressing a key on the keyboard as listed in the onscreen Help text as shown below. (Note that some model-specific options do not apply to terrain tiles.)

Displaying terrain tile coverage on the earth

In addition to examining a single VRSG terrain tile in the Model Viewer, you can look at how much land mass of the Earth a given set of tiles covers.

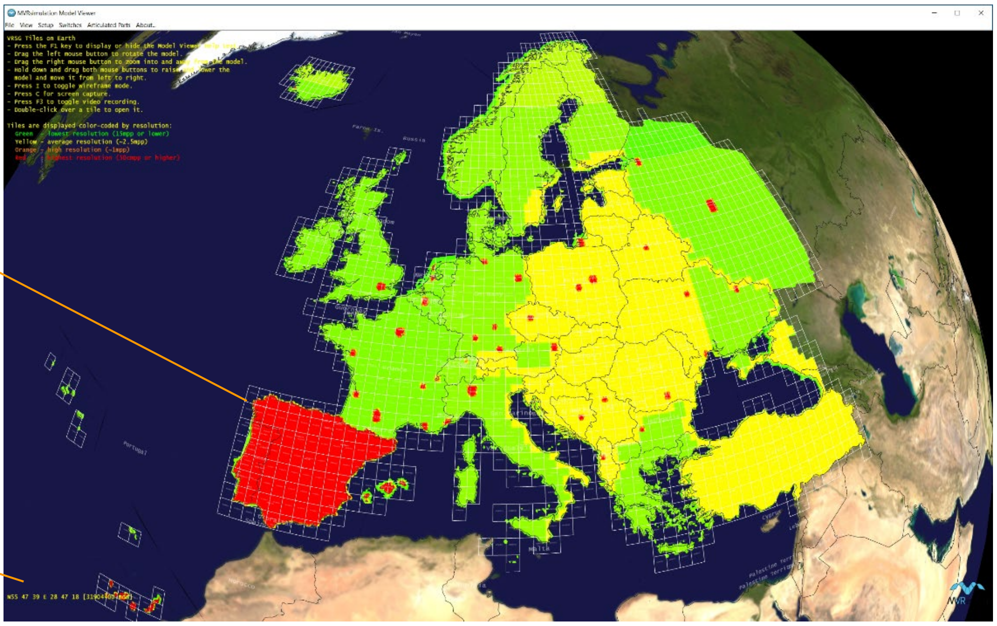
To display the Earth coverage of a set of terrain tiles:

1. From the Windows Start menu, choose MVRsimulation Model Viewer.
2. When the Model Viewer appears, choose File > Show Tile Coverage on Earth.
3. When the Browse dialog box appears, locate the directory of tiles you want to display and then click OK.

A grid representing the tile coverage appears, geo-located on a model of Earth. Each tile is colored to represent its resolution.

You can pan and zoom to move around Earth coverage in the Model Viewer just as you would for a model. In this case, you could zoom in for a closer look at the resolution of a given tile.

As shown in the next image, tile colors range from green for low resolution tiles (15 mpp imagery or lower) to red for high-resolution tiles (50cm per pixel or better). The yellow tiles indicate imagery with 1.0 - 2.5 mpp resolution imagery, and the orange-brown areas indicate imagery resolution of less than 1 mpp.


Grid of tile coverage on the earth model.
Coordinates and name of the tile on which the cursor is resting.
MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About..

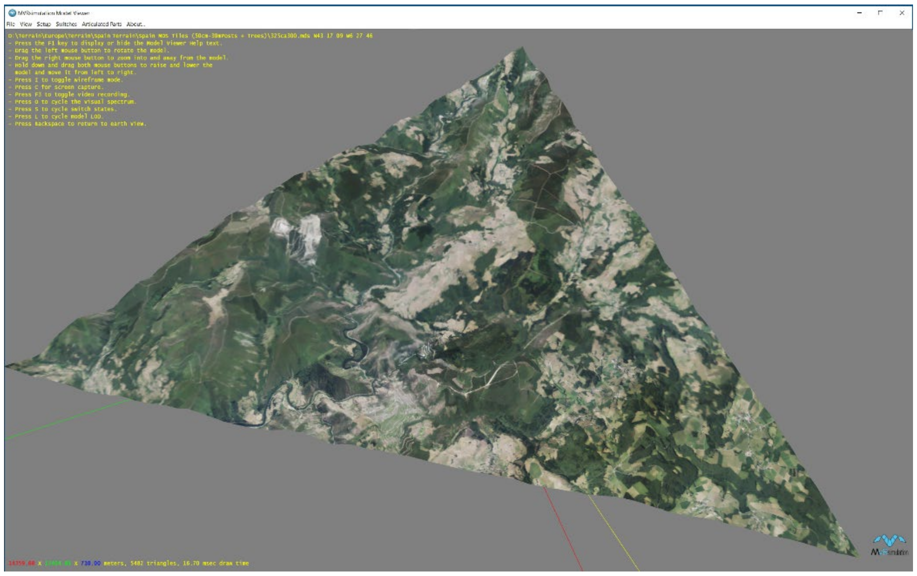
```
VRSG Tiles on Earth
- Press the F1 key to display or hide the Model Viewer help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Double-click over a tile to open it.

Tiles are displayed color-coded by resolution:
  Green  - lowest resolution (15mpp or lower)
  Yellow - average resolution (~2.5mpp)
  Orange - high resolution (~1mpp)
  Red    - highest resolution (30cmpp or higher)

N55 47 39 E 28 47 18 [31904400[illegible]]
```

From this display you can identify where the highest resolution data is concentrated and locate any gaps in tile coverage for a given area. This display can also provide feedback about the tiles that have been compiled during a compilation in progress.

Once you have opened the Model Viewer in the tile coverage view, you can open additional directories of tiles and have them also displayed on the earth model.

Also from the tile coverage view, you can double-click a tile to load it and inspect it in the Model Viewer.

The tile coverage view is oriented with the location of the last tile loaded facing you. This orientation can be helpful for determining the location of a single tile. When you put a tile in a directory of its own, open it in the Model Viewer and zoom in on it directly without rotating, the Model Viewer zooms to the single tile; providing its location information.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...

```
D:\Terrain\Europe\Terrain\Spain Terrain\Spain MDS Tiles (50cm-30mPosts + Trees)\325ca300.mds N43 17 09 W6 27 46
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Press Backspace to return to earth view.
```

```
14359.00 x [illegible] x 710.00 meters, 5482 triangles, 16.70 msec draw time
```

To switch back to the tile coverage view from the single-tile view, press the Backspace key.

Determining a tile that matches a set of coordinates

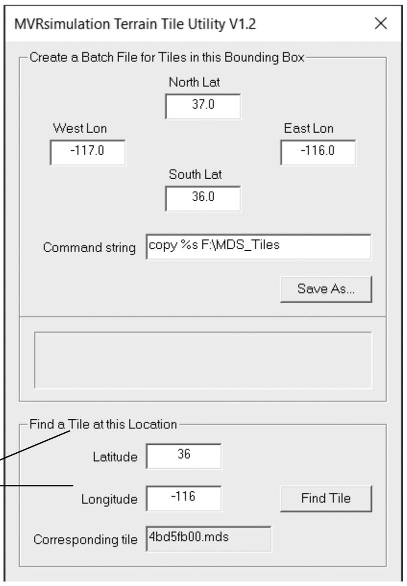
You can find out the name of the tile that matches a set of coordinates of interest with MVRsimulation's Terrain Tile Utility. Doing so can be useful for inspecting the terrain of the area of interest (by examining the tile in the Model Viewer), locating a tile on which you want to build up content, or deleting a tile you want to regenerate with updated imagery.

To find out the name of the tile for a given geographic area:

1. Double-click the executable tileMgr.exe, located in \MVRsimulation\Common\Utilities.
2. When the Terrain Tile Utility dialog box appears, enter the latitude and longitude of the tile you want to locate in the appropriate fields at the bottom half of the dialog box, and then click Find Tile.

The utility then displays the name of the corresponding tile that covers the region you specified. Note that this name is generated on the fly using the VRSG terrain convention for encoding a unique filename for a tile; the tile itself might not actually exist on your system.

(The filename is a bit pattern that records the recursion history of the tessellation process leading to any given tile.) The naming convention ensures that unique and repeatable filenames will be generated, even if the tile itself has not yet been generated.


MVRsimulation Terrain Tile Utility V1.2
Create a Batch File for Tiles in this Bounding Box
North Lat
West Lon
East Lon
South Lat
Save As...
Find a Tile at this Location
Find Tile

North Lat: 37.0
West Lon: -117.0
East Lon: -116.0
South Lat: 36.0
Command string: copy %s F:\MDS_Tiles
Latitude: 36
Longitude: -116
Corresponding tile: 4bd5fb00.mds

The latitude and longitude values of the tile you want to identify.

Creating a batch file for manipulating a set of tiles

To move, copy, or delete a set of terrain tiles of a specific geographic region, you can set up a batch file to manipulate them using MVRsimulation's Terrain Tile Utility. This alleviates the labor of manually entering a command-line Copy command for every tile.

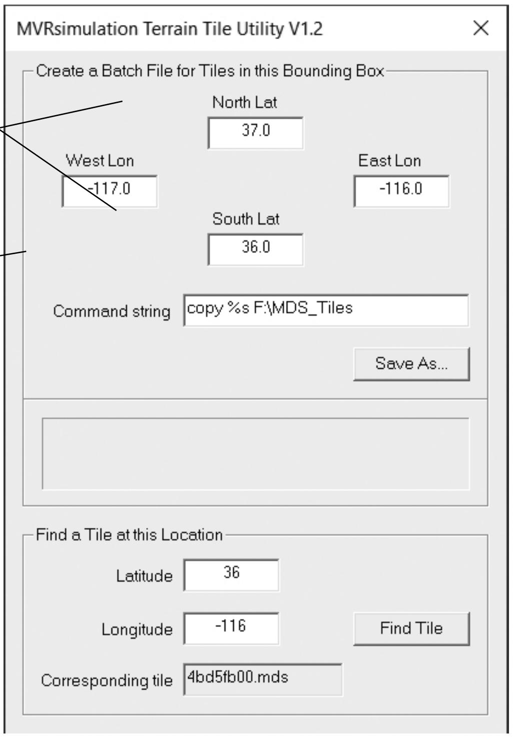
Tile Utility makes a set of commands for copying, moving, or deleting a set of tiles, using a template command string you provide. The utility figures out what set of tiles are inside a bounding box you provide, and then for each tile that overlaps the bounding box creates a command from that template to operate on the tile. When you set up the template command, you provide directory paths that are meaningful to your system, and embed them into command string.

To specify a square bounding box (region) of the tiles of interest and the command string to operate on each tile:

1. Double-click the executable tileMgr.exe, located in \MVRsimulation\Common\Utilities.
2. When the Terrain Tile Utility dialog box appears, enter in the top section:
   - The latitude-longitude coordinates of the geographic area of the target set of tiles. This creates the bounding box.
   - The command string to perform on each tile, such as copy, move, or delete. Use the percent symbol and the letter "s" (%s) as a wildcard to represent the filename of each tile.

The latitude and longitude coordinates of the region covered by the tiles of interest.

The command string that you want to operate on each tile in the bounding region.



MVRsimulation Terrain Tile Utility V1.2

Create a Batch File for Tiles in this Bounding Box

North Lat: 37.0
West Lon: -117.0
East Lon: -116.0
South Lat: 36.0

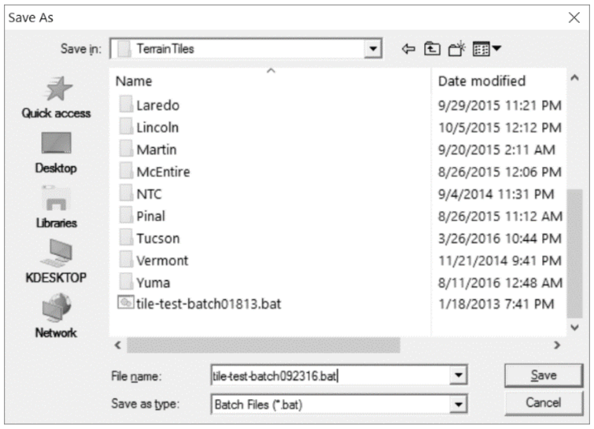
Command string: copy %s F:\MDS_Tiles

Save As...

Find a Tile at this Location

Latitude: 36
Longitude: -116
Find Tile
Corresponding tile: 4bd5fb00.mds

3. Click Save As to save the output to a named batch file.
4. When the Save as dialog box appears, type a name for the BAT file and click Save.



Save As

Save in: TerrainTiles

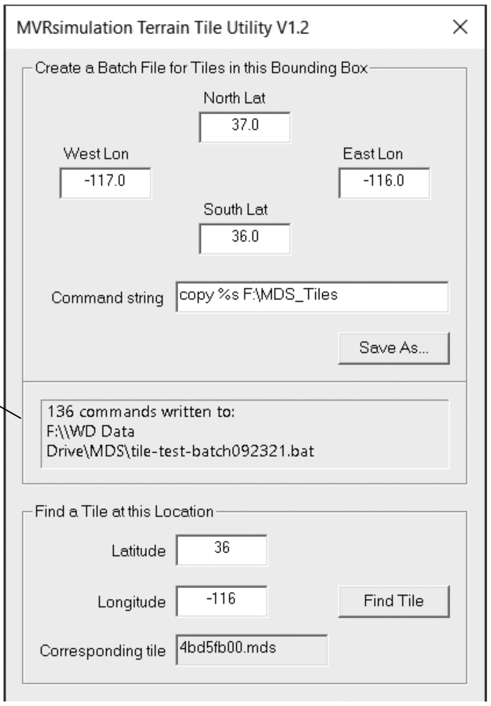
Name	Date modified
Laredo	9/29/2015 11:21 PM
Lincoln	10/5/2015 12:12 PM
Martin	9/20/2015 2:11 AM
McEntire	8/26/2015 12:06 PM
NTC	9/4/2014 11:31 PM
Pinal	8/26/2015 11:12 AM
Tucson	3/26/2016 10:44 PM
Vermont	11/21/2014 9:41 PM
Yuma	8/11/2016 12:48 AM
tile-test-batch01813.bat	1/18/2013 7:41 PM

File name: tile-test-batch092316.bat
Save as type: Batch Files (*.bat)

Save
Cancel

Next, the utility displays the total number of commands (one per tile) that it saved to the BAT file you specified:

The number of commands – one per tile – that were saved to the batch file you specified.


MVRsimulation Terrain Tile Utility V1.2
Create a Batch File for Tiles in this Bounding Box
Save As...
Find a Tile at this Location
Find Tile

North Lat: 37.0
West Lon: -117.0
East Lon: -116.0
South Lat: 36.0
Command string: copy %s F:\MDS_Tiles

136 commands written to:
F:\\WD Data
Drive\MDS\tile-test-batch092321.bat

Latitude: 36
Longitude: -116
Corresponding tile: 4bd5fb00.mds

After computing all possible tiles covering the geographic area of the given coordinates, the utility generates a list of command entries in the .bat file. Each entry contains a tile filename, with the command you entered.

For example, the following command string:

```
copy%s F:\MDS_Tiles
```

yields the following .bat file:

```
copy 4bd44000.mds F:\MDS_Tiles
copy 4bd44100.mds F:\MDS_Tiles
copy 4bd44200.mds F:\MDS_Tiles
copy 4bd44300.mds F:\MDS_Tiles
copy 4bd44400.mds F:\MDS_Tiles
copy 4bd44500.mds F:\MDS_Tiles
copy 4bd44600.mds F:\MDS_Tiles
copy 4bd44700.mds F:\MDS_Tiles
copy 4bd44800.mds F:\MDS_Tiles
copy 4bd44900.mds F:\MDS_Tiles
…
copy 4bdab800.mds F:\MDS_Tiles
```

Now you can either use the .bat file to perform the tile manipulation task, or edit the file further as necessary before using it.

Run the .bat file by double-clicking it, or by executing it from a command-line window.

CHAPTER 11

Manipulating Textures

This chapter describes various techniques for modifying textures to satisfy various scenarios needs, how to control which textures are loaded with a database, and how they are loaded. It also describes how VRSG handles texture swapping.

Supported texture formats

MVRsimulation products support texture images in the following formats:

- Windows BMP
- Graphic Image Format GIF
- Joint Photographic Expert Group JPEG
- PBMPlus Portable Grayscale Map PGM
- Portable Network Graphics PNG
- PBMPlus Portable Pixel Map PPM
- SGI RGB, RGBA, INT, and INTA
- MVRsimulation's proprietary texture format TEX
- Tagged Interchange File TIF, TIFF*
- VistaTGA,VST

*Although VRSG supports TIF image files as textures, TIF is a large standard encompassing many different possible image representations and compression schemes. Therefore, not all TIF formats may be supported as texture files. If you have problems with a TIF texture in VRSG, consider converting the file to another supported format, such as RGB or BMP.

Assigning a texture library path

You typically place the textures associated with a particular database in their own subdirectory located in the database's \Textures directory. Doing so makes moving or deleting the database easy, and it avoids naming conflicts with other databases that use a texture with the same name. Such textures can be those used in models or used directly on the terrain (such as a road texture or a microtexture).

Editing textures

You can easily modify textures with image-editing applications such as Adobe Photoshop or Corel PaintShop Pro. Situations that require modifying textures include:

- Reducing noise in the texture
- Reusing a texture for multiple scenarios of disparate regions
- Resizing a texture

- Creating alternate textures for sensor-view mode

Techniques for these modifications are described in this section.

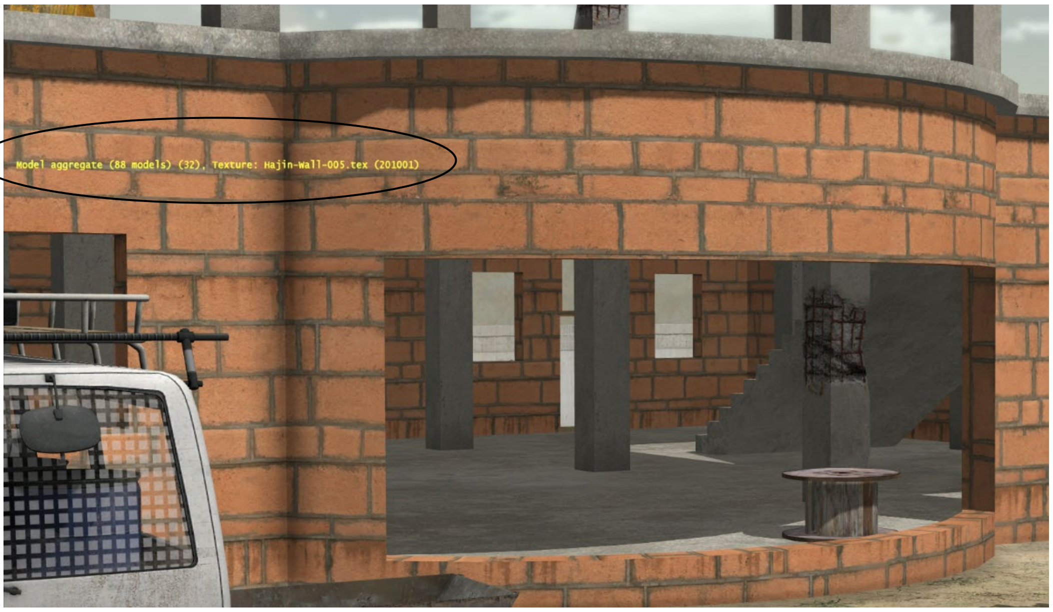
Identifying textures used in a database

From within VRSG you can find out the name of a texture that is used in a database. Doing so is useful for identifying a texture you want to modify, using it in other virtual worlds, and so on.

To identify a texture while visualizing a database in VRSG:

1. Run VRSG in windowed mode or Desktop Cover mode so that you have access to the mouse cursor.
2. Position the cursor over the texture area of interest. Press the Shift key and click the left mouse button. VRSG displays the name of the texture that is applied to the face you clicked as shown in yellow text in the following example:

The name of the model and texture on which the left mouse button was clicked.



```
Model aggregate (88 models) (32), Texture: Hajin-Wall-005.tex (201001)
```

Plugins for editing images in RGB format

If you need textures in SGI RGB and RGBA format (for alpha channel support), you can use Adobe Photoshop's optional SGI RGB plugin (available from Adobe's Support website and from third party sites such as www.telegraphics.com.au/sw/product/SGIFormat) outputting textures in RGB format. Two shareware applications, XnView (www.xnview.com) and Irfanview (www.irfanview.com), can also output an image file in RGB format.

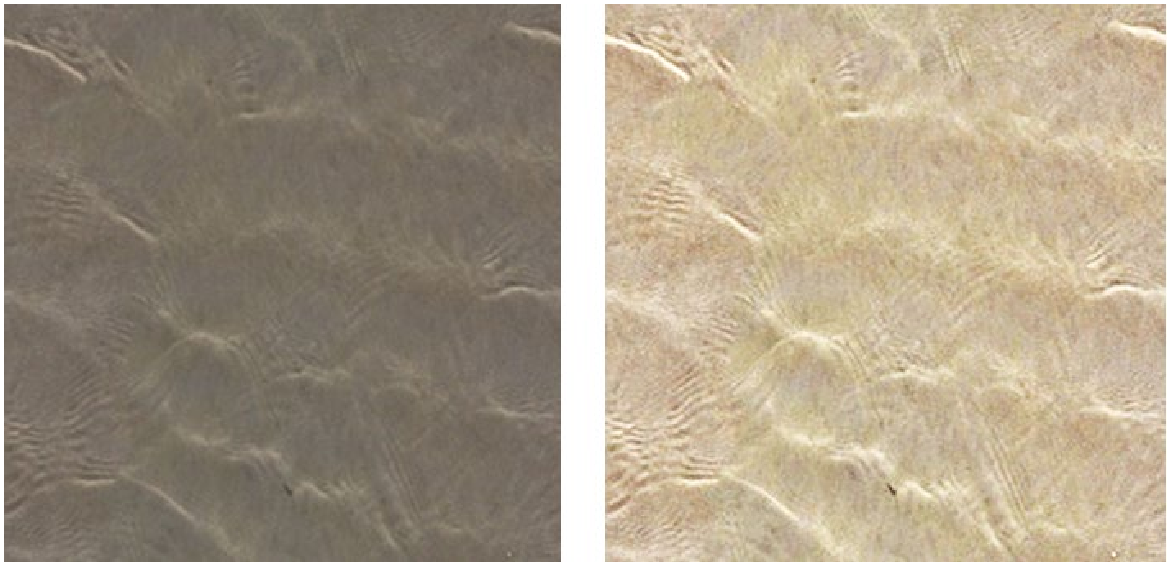
Modifying textures for a scenario

One way to manage alternate textures of a cultural feature is to have multiple textures in different layers of a single texture file. For example in a Photoshop or PaintShop Pro file of a building or a road you could have multiple layers of building façades or road textures, for quickly creating textures for a scenario.

Another way of creating a texture quickly is to simply copy a texture used in another scenario and modify it to suit the needs of the scenario.

Adjusting the color

The following example illustrates modifying the color of the sand in a texture file to match the sand in a different geographical region, with the original texture on the left and the modified texture on the right:



Using the Adjust > Curves feature in an image editor, you can make the sand texture a lighter color. As result, when you display the terrain database that uses this texture, the virtual world will display in VRSG with a lighter color of sand.

Reducing the noise

The noise in images is defined as unwanted artifacts that are introduced into the image, making it less realistic. Realism in the depiction of photo realistic textures is crucial for the suspension of disbelief required in a realistic, simulated virtual world. Visual artifacts are often created by the efficiency of the compression algorithm used to store the source image.

To remove problem areas in a texture that has random spots, use noise filters available in an image-editing application such as Adobe Photoshop or Corel PaintShop Pro. In particular, the Noise > Despeckle option in each product can clean up unwanted artifacts in the imagery.

The noise in the example above stands out in the areas that do not have much detail.

Using a Noise Despeckle filter, the resulting image looks like the above.

Resizing textures to VRSG's required dimensions

Texture files are typically subject to 3D-accelerator hardware constraints. Although VRSG can support any texture size, the practical limitation of your graphics card will ultimately dictate the maximum size and dimension of the image. VRSG requires texture length and width dimension to be powers of 2. This means that allowable texture dimensions are 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096 pixels per side. No texture should exceed 4096 pixels on a side. VRSG will accept textures that do not conform to these dimensions and will resize them at runtime to the next power of two, but doing so will increase load time.

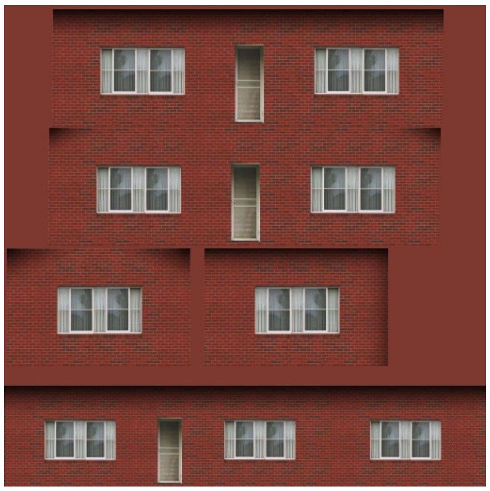
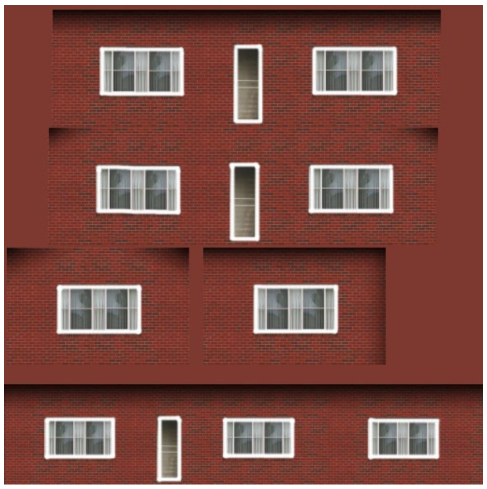
A texture with a length of 256 pixels and a width of 128 pixels would be of the correct dimensions for use in VRSG. When you resize a texture to maximum power of 2 per side you would typically scale down the dimension. But if the dimension is close to the next acceptable higher number, you could scale up. For example a texture of 954 x 1450 pixels would be resized to 1024 x 1024, a texture of 610 x 500 pixels would be resized to 512 x 512. Many graphics cards only support textures up to a maximum size. At the time of this writing, most graphics cards support a maximum length and width dimension 4096 x 4096 pixels. Make sure the texture size and dimensions are supported by the 3D accelerator you use to run VRSG.

You can resize textures in an image editor to comply with VRSG's required dimensions. Additionally, to make a texture swappable or *pageable*, as described later in this chapter, you can resize the texture file to be 512 x 512 pixels, or as of VRSG 7.0 and terrain built in Terrain Tools v2.0, resized to 1024 x 1024, or 2048 x 2048 pixels. If a texture has an alpha channel, resizing it to one of the sizes listed above will not make this texture pageable because textures with an alpha channel are not pageable. Therefore, if a texture is smaller than 512 x 512 and contains an alpha channel, do not resize this texture to 512 x 512, rather leave the texture in the smaller size.

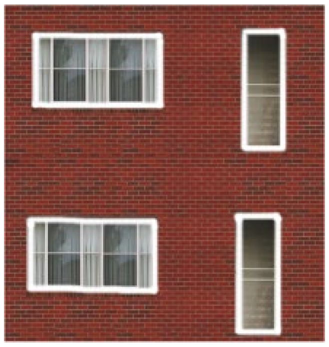
Creating textures for sensor-view mode

When running a scenario in sensor-view mode, VRSG loads texture files in real time that have the extension *_IR.*. You can modify any texture used in the database to make alternative sensor view textures. You would create these alternative IR textures for static models used in the database scene and store them in the same directory as all other model textures used in the database.

The following example illustrates modifying texture files to use in a sensor-view scenario. This example uses an image-editing application to make the IR texture; you could also modify the texture by using a modeling program such as Presagis Creator. To make an IR texture from an existing texture, first copy the original file, in this case house-011_wall_512.rgb, to a new file house-011_wall_512_IR.rgb. Then open the _IR.rgb file in an image-editing application such as Adobe PhotoShop or Corel PaintShop Pro.

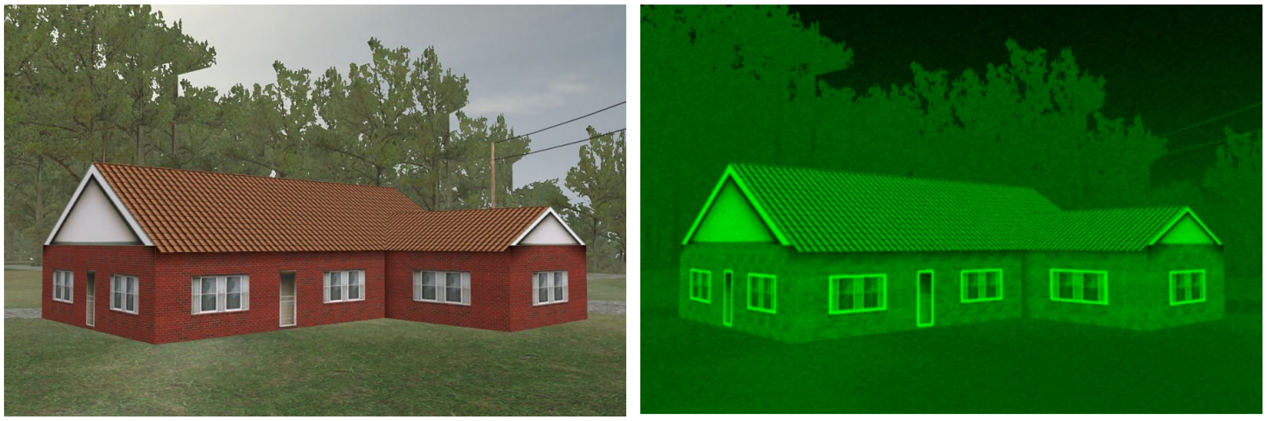



Using a dodge or airbrush tool, you can apply white hot spots where thermal bleed-through might occur based on your knowledge of thermodynamics (in this case, on the windows and doors, as shown in the edited texture on the right). Note that an image could also be the result of physics-based models using material properties. This example illustrates a hand-editing method for situations when you do not have material codes for the images.



Do not change the dimensions of the image; the dimensions must match those of the original texture.

After you finish modifying the texture, save the IR texture in the same directory as the original texture, with "_IR" appended to the filename. When you run VRSG again in sensor mode the building's IR depiction will show the modifications you made to the texture:



Notice the thermal bleed-through around the windows and doors of the house in the IR scene on the right.

Creating normal maps for entity models

Many entities in the MVRsimulation military vehicle library have normal maps. For your own models that you have converted to MVRsimulation's format, you can associate a normal map with any color texture with the suffix "_NML" in the filename (not the extension). The texture must be either TEX or RGB formats. The TEX format is MVRsimulation's proprietary texture format .tex, which is used for distributing 3D model and texture libraries in a non-editable format.

For example:

```
A10_COCKPIT_1024x512_NML.tex
```

The RGB channels of the image contain the normal vector, quantized to 8 bits. For example:

```
red = x*127.0 + 127.0
green = y*127.0 + 127.0
blue = z*127.0 + 127.0
```

This obtains the normal components that are in the range -1.0 .. 1.0 in the unsigned char range of 0..255.

If an alpha channel is present, it is interpreted as a "gloss map," which gives per-pixel attenuation of the specular response. If the texture has no alpha channel, there will be no specular attenuation.

Normal maps use the same UV mapping as the color maps, so no editing of the associated model is required.

Microtextures

Detailed terrain textures, or *microtextures*, are high-resolution, geotypical ground textures. At runtime, VRSG blends one or more microtextures with the geospecific terrain imagery to add further detail to the terrain when viewed at ground level. Microtextures add realistic details when the source imagery is not of sufficient resolution to provide detail of the terrain when it is viewed at close range. Generic textures for sand, grass, and gravel are examples of geotypical microtextures that can blend with the geospecific terrain imagery to produce the most realistic visual depiction of the virtual world.

Each directory of terrain tiles can have its own unique microtexture, placed in a \Textures subdirectory within the directory that contains the terrain tiles. VRSG will apply the microtexture present in that subdirectory to all the terrain tiles in that directory. Examples of such a directory can be found in any of the 3D terrain datasets installed with VRSG, such as:

\MVRsimulation\VRSG\Terrain\Syria\Hajin\MDS\Textures

Terrain tiles built with MVRsimulation Terrain Tools that reference external textures for roads or other features, are also stored in the \Textures subdirectory.

A microtexture can be in any image format that VRSG supports (such as .jpg, .bmp, .rgb, and so on) and must be a square image in dimensions described earlier (2048 x 2048, 1024 x 1024, or 512 x 512). The filename must be in the following form:

vrsg_microtexture_*X_Y_Z.<file-extension>*

The X, Y, and Z parameters control the microtexture mapping:

- X – mapping frequency, in units of texture repetitions per meter. A value of 0.1 indicates that the microtexture pattern covers an area of 10x10 meters.
- Y – fade-out distance in meters. The microtexture fades out completely by this range.
- Z – blend bias. The minimal percentage of base terrain imagery that shows through the microtexture at close range.

For example:

```
vrsg_microtexture_ 0.1_800_0.15.tex
```

Microtextures must be located the \Textures subdirectory of the terrain tiles directory. When VRSG encounters a microtexture of name in the form "vrsg_microtexture_ X_Y_Z.tex", VRSG will look for an IR texture of the form vrsg_microtexture_ X_Y_Z_IR.tex.

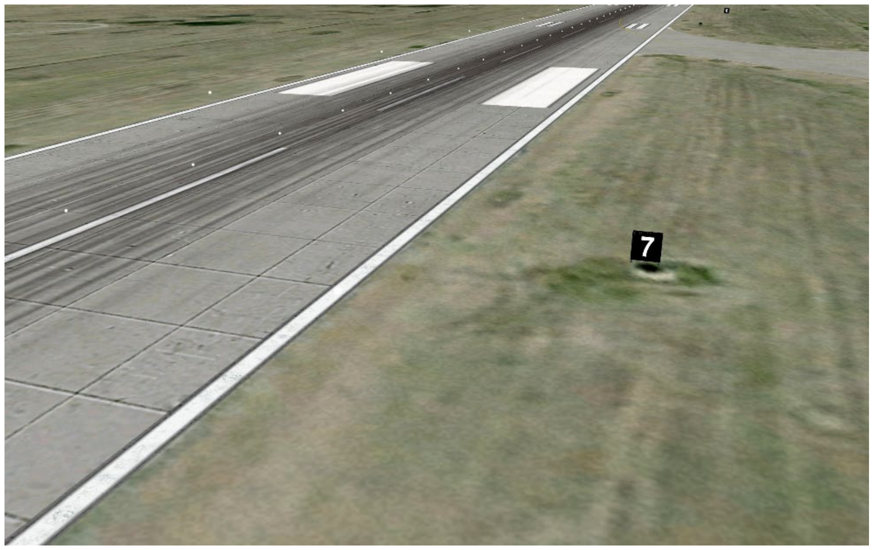
When VRSG renders terrain tiles from tile directories that contain no microtexture, it applies a generic high frequency noise pattern microtexture.

VRSG is delivered with a set of high-resolution dirt, grass, and sand microtextures you can use with your terrain. These textures are located in MVRsimulation\VRSG\Textures\Microtextures. To use one of these microtextures with your terrain, simply copy the texture, rename is as described above and place it in the \Textures subdirectory of the terrain tiles directory.

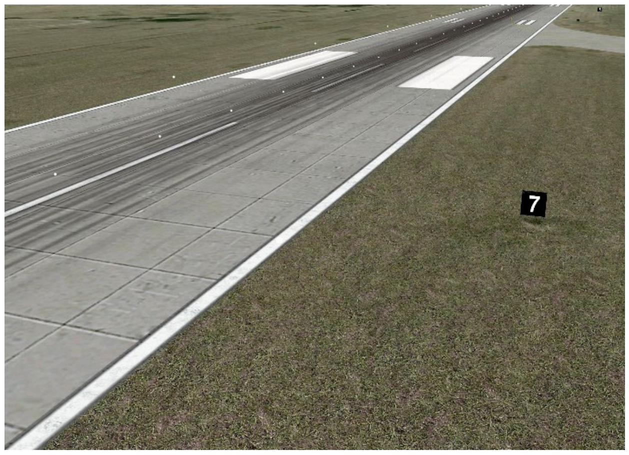
The following images show three versions of an airfield runway scene with:

- No microtexture.
- A grass microtexture in use, using the parameters described above.

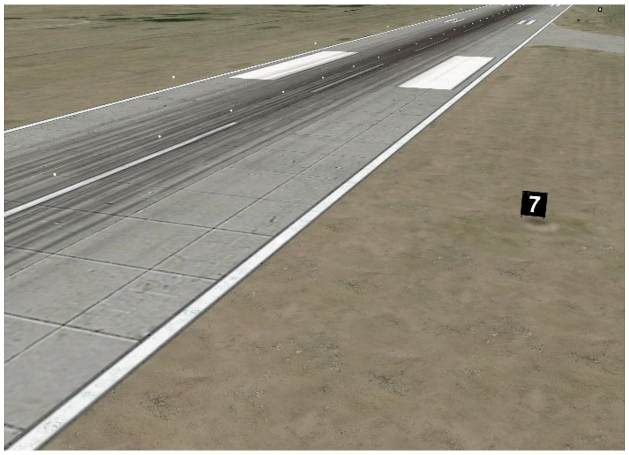
- A different microtexture.



VRSG screenshot of a scene without a microtexture.



Same scene using a grass microtexture.



Same scene using a dirt microtexture.

Texture swapping

VRSG supports software-based texture swapping. Texture swapping is useful for cases in which you have a large amount of texture imagery whose total size exceeds the hardware memory that your 3D graphics card supports. VRSG can use your system's hard drive to swap textures in and out of the hardware texture memory on the graphics card. This texture-swapping feature provides unlimited texture memory, which is useful primarily for PCI-based graphics cards with limited texture memory.

Pageable textures

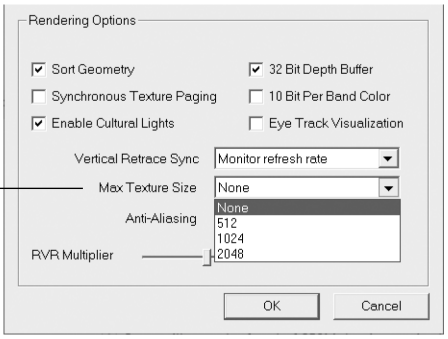
VRSG can support textures up to 4096 pixels on a side if your video card can support it. However, the textures that VRSG can swap to disk, called *pageabl*e textures, must be 512 x

512 pixels, or now as of VRSG v7.0 and terrain built in Terrain Tools v2.0, 1024 x 1024, or 2048 x 2048 pixels.

Pageable textures cannot contain an alpha channel. If a texture image has an alpha channel or any side is not one of the dimensions listed above, the texture file is non-pageable and is always video-memory resident. Bear in mind that nonpageable textures take away from the pool of video memory, thus leaving less room for pageable textures. Therefore, there is a limit on nonpageable textures, which is the video memory limit. Pageable textures are limited only by disk. As described earlier in this chapter, you can resize the textures you want to be pageable in an image editing program.

You can select the Max Texture Size value in the Rendering Options section of the Advanced Graphics dialog box (accessed from the Dashboard's Graphics tab) to force VRSG to limit all texture sizes to a maximum size.

Downsizes all loaded textures to the specified size.


Rendering Options

- [✓] Sort Geometry
- [ ] Synchronous Texture Paging
- [✓] Enable Cultural Lights
- [✓] 32 Bit Depth Buffer
- [ ] 10 Bit Per Band Color
- [ ] Eye Track Visualization

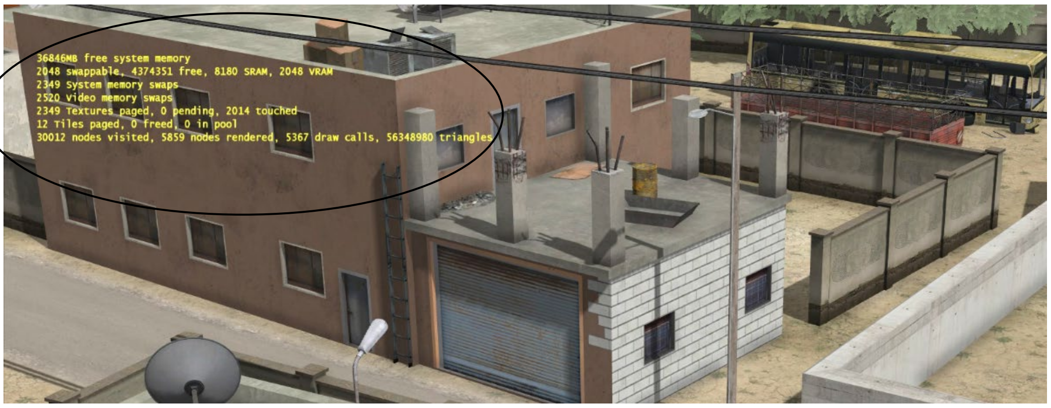
Vertical Retrace Sync: Monitor refresh rate

Max Texture Size: None

This feature is useful for running large databases with a lot of high-resolution models on systems with minimal texture memory. As described in the *MVRsimulation Product Installation Guide*, MVRsimulation recommends that you run VRSG on a system that has an Nvidia GeForce RTX 3080 Ti graphics card with at least 11 GB of memory.

You can display texture-loading information in VRSG by pressing the T key on the keyboard, as shown in the example below:

System memory, texture memory, and other texture loading statistics.



```
36846MB free system memory
2048 swappable, 4374351 free, 8180 SRAM, 2048 VRAM
2349 System memory swaps
2520 video memory swaps
2349 Textures paged, 0 pending, 2014 touched
12 Tiles paged, 0 freed, 0 in pool
30012 nodes visited, 5859 nodes rendered, 5367 draw calls, 56348980 triangles
```

You can override the selected Max Texture Size option on a per-model basis by creating a file \Models\max_texture_sizes.txt, in which you list one model per line, followed by the maximum texture size for that model, as follows:

```
CVN-77.US.grey.hpy 4096
```

You do not need to know the maximum texture size for a given model; the 4096 entry shown in the example above would load the full-resolution model, even if its largest texture was only 2048 x 2048.

Note: With a VRSG installation, 3D content libraries of over 10,000 high-resolution models are installed. The Max Texture Size option, used with a max_texture_sizes.txt are useful tools for running VRSG on an older machine with an older video card with limited VRAM and you want to limit the resolution of models that are less important in your simulation session.

CHAPTER 12

3D Model Content

MVRsimulation VRSG is delivered with substantial libraries of 3D content. All models in the libraries are in MVRsimulation's HPY or HPX model format, described in the chapter "MVRsimulation 3D Model Formats." You can use these models to populate your virtual worlds with cultural features and to interact with networked character and vehicle entities in real time to carry out scenarios. Most of the military entity models are created from high-resolution photographic textures and detailed dimensions of the actual model.

This chapter provides some information about what goes into the making an entity model. Using an example from the MVRsimulation 3D military entity library, this chapter describes the components of a model, the consideration and tradeoffs that goes into building a model, and levels of details. The image below shows the Shadow-200 IE UAV model (RQ-7B-IE.US.grey.hpy) from the MVRsimulation 3D military entity library. This model is used as an example throughout this chapter.

Overview

Adding models to a terrain database as part of a real-time simulation requires trade-offs. In the ideal case, models would have infinite geometric detail and infinite image resolution. Current computing capability does not yet support infinite model complexity in real-time, therefore one often has to balance the proportion of the model that is photographic-based or polygon-based. When you create a model entirely from raw source data you can control the level of detail that the model contains. Using high-resolution photographs, you can optimize the 3D models, minimizing the number of triangles and maximizing the rendering speed of the computer graphics subsystem.

The choice of how to model a 3D object to be as realistic as possible is determined by the type and limitation of the simulation. The ideal 3D database is one that supports a variety of virtual scenarios, from the ground activity of vehicles, individual, and special effects, such as those found in urban warfare training, to high-speed fixed-wing flight simulations. With a single terrain representation, models that need to be realistic at the ground level typically require large textures and significant geometry -- on the order of 5,000 to 50,000 polygons. Many models of tracked vehicles have 20,000 polygons at the highest resolution. This case can potentially be a bottleneck for processing the 3D geometry in order to maintain real-time performance. Each model in the VRSG 3D content military entity library has, at a minimum, a primary 4096 x 4096 or 2048 x 2048 pixel-resolution color texture map; many models have additional subsection texture maps as well. Geometric complexity of a model varies according to the vehicle on which is based; many wheeled ground vehicles have 50,000 polygons or more at the highest level of detail. The VRSG rendering engine has no fixed limit on the number of models, model types, or model effects that can be displayed for a given visual channel.

Rendering models in real time

When VRSG is running on a dual CPU PC-based high-end system, it can render over 1 million triangles per second and still maintain 60 Hz in a dense urban scene, but you are limited in the number of models you can model at such fidelity. In this case you would need to put levels of detail (LODs) in the models that fade out aggressively, at for instance 4 kilometers, to minimize the geometry processing for a given scene. At 4 kilometers, the model is switched out for a very low resolution model. However, high-speed fighter aircraft pilots need to discern detail of an aircraft often from a significant distance. In this case, geometry of the vertical tail surface and its orientation in conjunction with the light reflected off of the canopy are important for the pilot to accurately acquire targets out to as far as 20 kilometers. In this case, you need to maintain a high LOD for the models out to a significant range.

Models of buildings or other simple structures at the lowest level of detail seen from above in a flight simulation could be modeled with a simple box-like geometry and small textures on each side. For example, an architecturally typical repeating pattern could be used on all horizontal surfaces for buildings. This means a single texture could be used to model all sides. This approach contrasts with the case in which buildings are created with high resolution unique textures for each face; this case requires large amounts of texture memory. Texture compression offloads the graphics card from some of the load. The Microsoft DXT11 hardware accelerated texture compression will reduce the texture storage requirement onto the graphics card memory by a factor of 4. When you use more texture memory that the graphics card can support, you may get an error message from the application.

MVRsimulation VRSG provides a texture paging capability that greatly offloads the texture memory limitations of a given graphics card. Textures that are sized as 512 x 512 pixels on a side will be software-paged by VRSG. This means that VRSG can load and offload textures from the graphic card, which allows your scenario to exceed the card's hardware graphics memory capacity when rendering real-time object models. See the chapter "Manipulating Textures" for more information about texture paging.

About the models in MVRsimulation's 3D content libraries

The libraries of 3D content that are delivered with VRSG contain run-time models with full-resolution textures of 4096 x 4096, 2048 x 2048, or 1024 x 1024.

The models contain a great deal of detail which is valuable in cases where the eyepoint in a simulation is very close to the entity, such as 10 meters or less. If you find that viewing a large number of models with full-resolution textures simultaneously in a scene affects the rendering performance, you can have VRSG limit the texture size it renders by reducing the Max Texture Size value in the Rendering Options section of the Advanced Graphics dialog box (accessed from the Dashboard's Graphics tab).

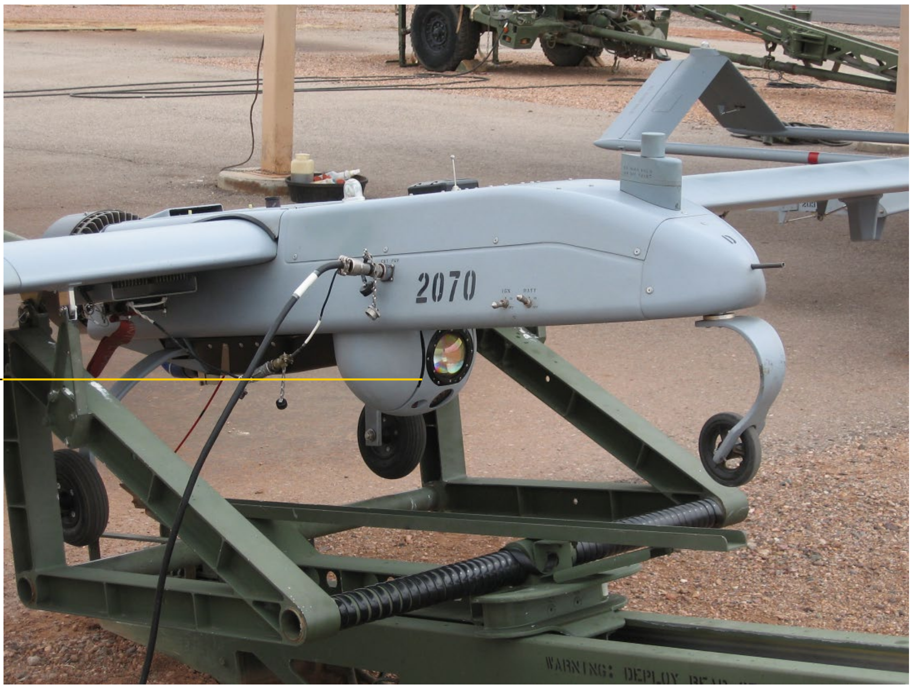
How MVRsimulation vehicle models are created

This section describes how a typical vehicle from the MVRsimulation library of military vehicles was built. Initial source data collection for the Shadow 200 Unmanned Arial Vehicle (UAV) model, used as an example in this chapter, started by taking close-up and as much as possible, perspective-less digital camera photographic images of the actual vehicle. Many 6-or-more megapixel digital cameras can provide images of sufficient quality to use for 3D modeling. Trade shows provide an excellent forum for gathering source data imagery under good lighting conditions.

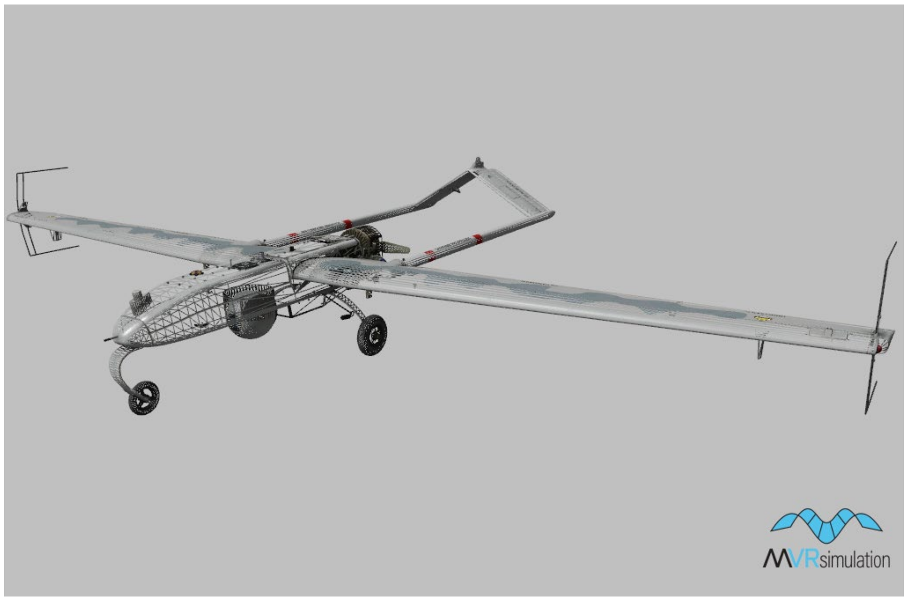
Geometry and texture maps

The following images are among many photographs MVRsimulation took of the Shadow 200 UAV. Multiple images were taken from as many vehicle perspectives as possible. Context images of the UAV were also taken to help show location and operation of any articulated components of the vehicle.

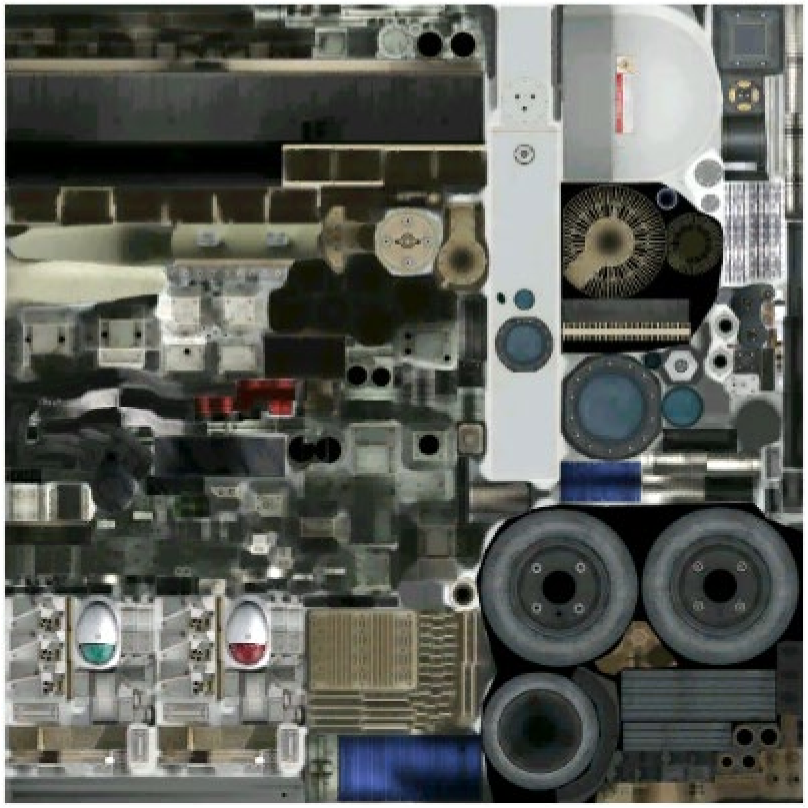
This next image shows close-up detail of the UAV video camera surveillance payload:



There are many ways to establish 3D geometry for a given vehicle. In this case, CAD data was used as the foundation of the model geometry as shown in the RQ-7B-IE model below:

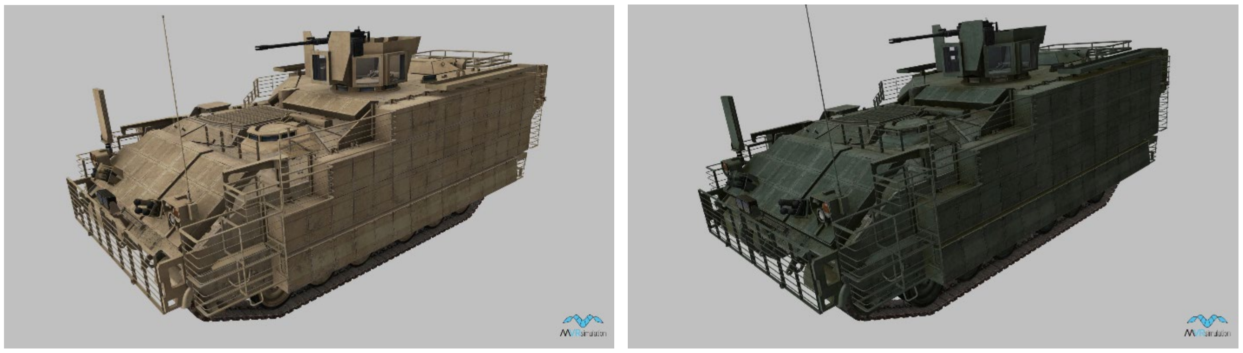


Subsections of the vehicle images are cut out; then a 3D modeling tool is used to apply the graphic to the model geometry. The resulting texture map shows how the vehicle graphics are situated to map to the model geometry.



Paint schemes and derivative models

Where appropriate, vehicles in MVRsimulation's 3D military vehicle model library may have multiple color schemes that represent camouflage schemes and levels of use. The following example shows the green and desert paint schemes for the XM1283.US model:



Levels of detail

Models often have multiple representations of varying detail embedded within them. These different representations of varying detail are commonly referred to as Levels of Detail (LODs). The basic idea behind levels-of-detail is that when a model is viewed from a sufficient distance, some features of the model become too small to contribute to the scene. When a model reaches some predetermined range, the model is said to "switch out" to a lower level of detail. Most MVRsimulation vehicle models contain at least 3 LODs. A model's LODs are all contained in the single model file.

LODs are a way to reduce the model complexity in discrete states such that when the vehicle is seen from further and further away, only the discernable detail in terms of the human perception to recognize and identify the model are maintained. Essentially, models seen from a far distance do not need to have as many polygons as a model seen up close in order for the model to accurately represent the real vehicle. A similar technique is used with the texture map but modern PC-based graphics cards automatically modify the representation of a texture map using a technique called mipmapping, in which discrete states of the texture map are computed and stored by the graphics card. As the model is viewed from increasing distances, lower resolution texture representations are substituted for the original higher resolution texture. Both the LOD-swapping and mipmapping techniques are ways to reduce the resources used by the system in order to increase the overall complexity of the rendered scene and thus ultimately better approximate reality.

The Shadow-200 UAV model (RQ-7B-IE.US.grey.hpy) uses three LODs, which are shown below. The model can cease to be rendered after 15 kms as a fourth LOD, which approximates the distance at which human visual perception no longer can see the vehicle.

MVRsimulation

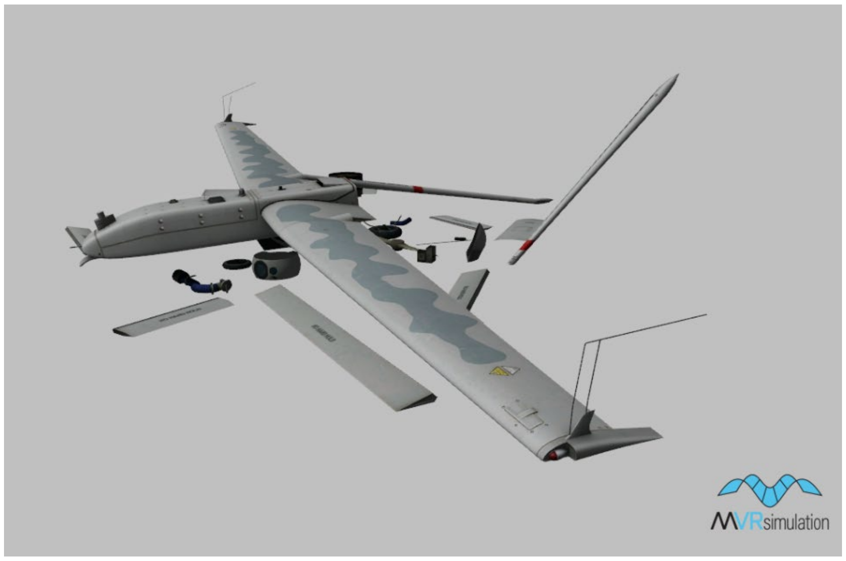
RQ-7B-IE.US model with 302 triangles at the lowest level of detail used to view the model from a distance of 10,000 kms and beyond.

MVRsimulation

RQ-7B-IE.US model with 6881 triangles at an intermediate level of detail used to view the model from a distance of 300 km.

RQ-7B-IE.US model with 45,643 triangles at the highest level of detail used to view the model from a distance up to 2 kms.

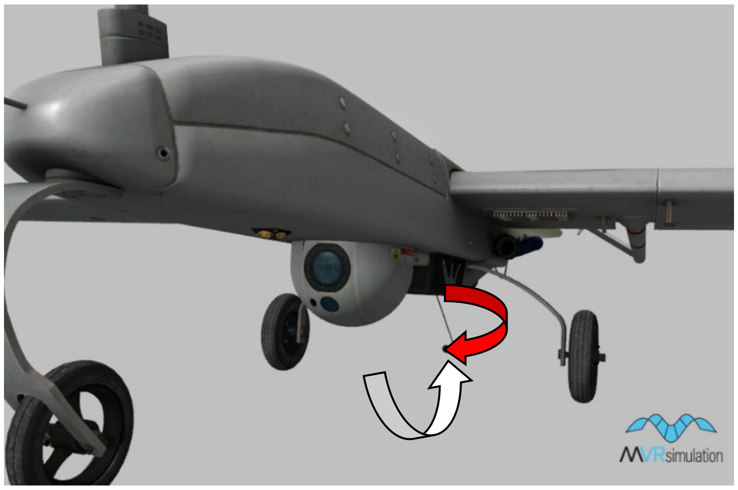
The model's damage state, as shown next, is modeled with 11,782 triangles.



Some LOD ranges supplied for a model are not suited to all viewing circumstances. For example, if you are running VRSG at a higher screen resolution, highly detailed components of a model will be visible at further ranges than if you are viewing the same model at the same range, but at a lower screen resolution. Therefore, higher screen resolutions call for larger LOD ranges. Likewise the field-of-view (magnification) level affects when the detail of a model becomes too insignificant to contribute to the scene. Given a 3 degree field-of-view (a high magnification) and a given range, the projected size of a model will be much larger than the same model viewed at the same range but with a 60 degree field-of-view (a low magnification). Thus smaller fields-of-view call for larger LOD ranges.

Articulated components

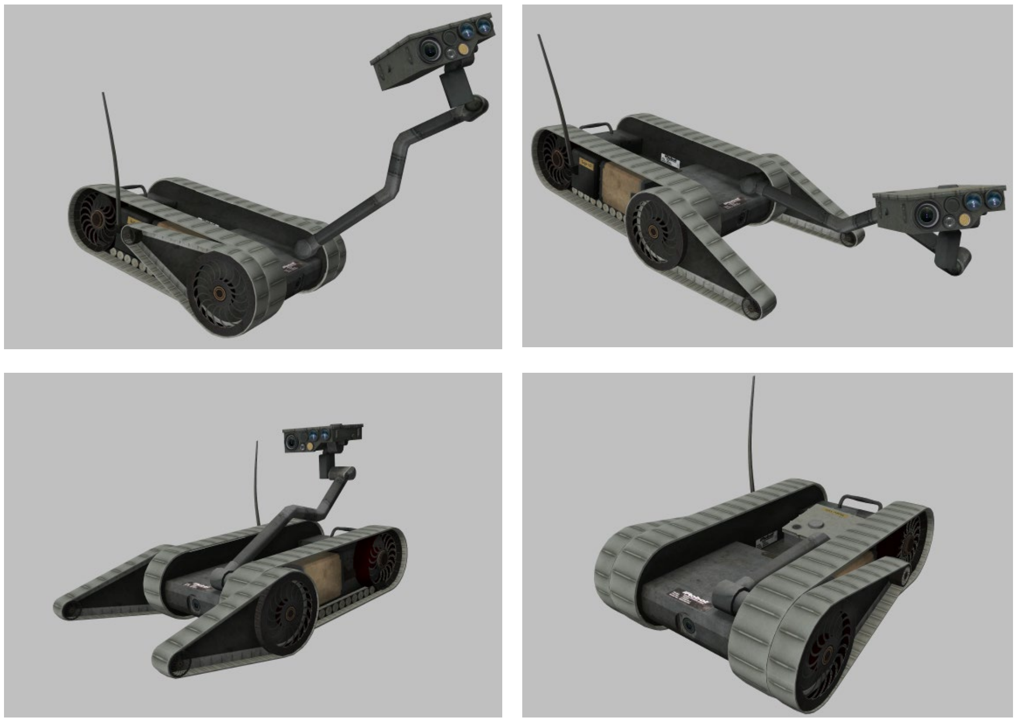
Most VRSG entity models have one or more articulated components (degrees of freedom) that follow the World Reference Model (WRM) metadata format as described in the chapter "MVRsimulation 3D Model Formats." The number of articulated components depends on the use of the vehicle and the model's complexity.



Simple models, such as missiles, may not have any articulations. In the RQ-7B-IE.US model, the video camera payload shown in the image above is modeled as a constrained component capable of 180 range of movement (as depicted by the white arrow) mounted on a fully rotating turret (as shown by the red arrow).

This movement enables the camera to move in the direction that a UAV operator would designate by using a gamepad or joystick in the ground control station remote cockpit.

Another example is the XM1216 SUGV model, with its articulated camera head, neck, and rotating flippers:



For information about how to use the MVRsimulation Model Viewer to inspect the articulated parts, geometry, and other characteristics of a model, see the chapter "Previewing Models, Effects, and Terrain."

CHAPTER 13

Integrating 3D Sounds

VRSG's sound generation feature enables you to add sounds to events in the VRSG virtual world. The integrated 3D sound support uses the Microsoft DirectSound and DirectSound API (as part of the DirectX SDK) to achieve integrated visual and audio capability in a virtual world. This means that if you associate a sound with an entity such as a vehicle, when you attach to the vehicle in a scenario, VRSG plays the sound associated with that entity.

DirectSound provides low-latency sound mixing, hardware acceleration, and direct access to the sound device in a hardware-independent manner. With VRSG, Direct3D and DirectSound provide 3D sound in a VRSG scenario without any reduction in frame rate, as many hardware vendors are incorporating the API at the silicon level.

VRSG provides these advanced DirectSound features:

Perception of sound positions	**Listeners**
Sound cones	Minimum and maximum distances
Position versus velocity	Integration with Direct3D
Units of measure and distance factors	Mono and stereo sources

For information about these features, see the description at Microsoft's website at http://www.microsoft.com/directx.

Requirements

To use VRSG's integrated 3D sound support, you must have the following:

- DirectX (including the DirectSound API).
- A sound card.
- Sound sampling software (typically available with the sound card).
- Either one or more .wav sound files or an audio CD sound effect library.

Adding sounds to a virtual world

You can easily add sounds to events in the VRSG virtual world. You can use sounds from sound libraries, or you can record your own. Using sound sampling software you can record sounds digitally from an audio CD, a .wav library, or from field recordings.

The sound segment you record for use in VRSG must be in the following format:

- One second
- 11,025 Hz
- Mono
- 16-bit format
- Symmetrical wave forms

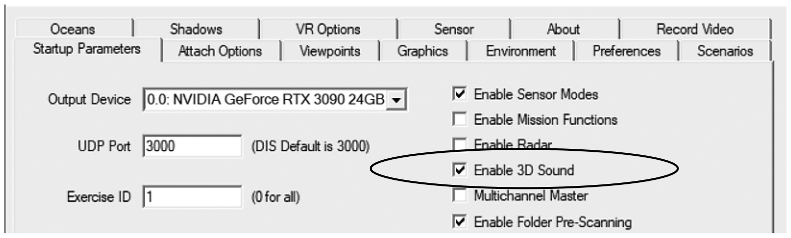
These sounds can be applied to entities, events, or to simulating rapid-firing weapons.

VRSG is fully configured to generate sounds in a DIS virtual world. These default sounds in .wav format are included in the \MVRsimulation\VRSG\Sounds directory.

Once you have a sound, you can associate it with a particular event or entity in a virtual world.

To associate a sound with an event or entity:

1. Add a reference to the .wav sound to a VRSG initialization (.ini) file. For information about how to do this, see the description of editing the ModelMap.ini file in the chapter "Configuring Models and Events."
2. Associate the sound with a DIS enumeration event.
3. Run VRSG with the Enable 3D Sound option selected on the Startup Parameters tab.



Output Device 0.0: NVIDIA GeForce RTX 3090 24GB
UDP Port 3000 (DIS Default is 3000)
Exercise ID 1 (0 for all)

[✓] Enable Sensor Modes
[ ] Enable Mission Functions
[ ] Enable Radar
[✓] Enable 3D Sound
[ ] Multichannel Master
[✓] Enable Folder Pre-Scanning

Generated sounds are attenuated using an exponential roll-off between the minimum and maximum distance. At the minimum distance, the sound is at zero attenuation. At the maximum distance, the sound is at its maximum attenuation.

When you create a customized sound, you must cut the sound off so that the end of the sound clip is also the beginning of the clip. This means that the end of a cycle should be the same as the beginning of the cycle, or symmetrical.

In this example, a user has recorded a sound segment from an audio CD that contains a sound effects library for aircraft sounds.

Once you have created the .wav sound clip with the guidelines described above, you can insert the file into the VRSG sound mapping initialization file.

To associate a sound with an event, you edit the appropriate VRSG mapping initialization file. Sounds are mapped to events or entities by the DIS enumeration standard.

In the following code fragment from the VRSG sound initialization file (EntityMap.ini), 10 meters is the minimum distance and 1500 meters is the maximum for the attenuation of a helicopter (helo.wav) entity sound:

```
! Helo sound
1 2 225 20 * * * helo.wav 10 1500
```

Notice the DIS enumeration for a US air vehicle as described by the 1 2 225 20 * * * enumeration components.

Mapping sounds

The file EntityMap.ini, located in the \MVRsimulation\VRSG\Sounds directory with the *.wav files, has default settings for a limited set of entity sounds. You can use asterisks as wildcards in the enumeration mapping process to account for abstractions in the DIS designations. For more information about mapping events and entities using the DIS enumerations, see the chapter “Configuring Models and Events.” The enumeration integer values preceding each sound are shown in the following example:

EntityMap.ini

```
! EntityMap.ini

! Copyright 1997 - 2025 MVRsimulation Inc. All rights reserved.
! URL: http://www.mvrsimulation.com  Email: support@MVRsimulation.com
!

! This file can be used only with MVRsimulation’s Virtual Reality Scene
! Generator (VRSG).
!

! The receipt of this file and your use of the information contained
! herein is subject in all cases to your agreement to the provisions
! governing “Additional Materials” of the current MVRsimulation Inc.
! license agreement found at:

! https://www.mvrsimulation.com/howtobuy/license_agreement_policy.html.
!

! Syntax:

! <Kind>  <Domain>  <Country>  <Category>  <Sub-category>  <Specific>
<Extra>   <Wav file>  <Min distance>  <Max distance>
!

! Refer to the VRSG User's Guide for further details
!

! Helo sound

1 2 225  7 * * * helo.wav 10 1500
1 2 225 20 * * * helo.wav 10 1500
1 2 225 21 * * * helo.wav 10 1500
1 2 225 23 1 * * helo.wav 10 2000
1 2 225 50 * * * atr2.wav 10 1500
1 2 78  20 2 * * helo.wav 10 2500
1 2 13  20 2 * * helo.wav 10 2500
!

! Default air-vehicle sound
* 2 * * * * * jet.wav 50 2000
!

! Default dismount sound
3 * * * * * * infantry.wav 1 20
```

```
!
! Leschi Town demo - wheeled vehicle sound (no treads) for Strykers and
HumVees
1 1 225 6 1 * * WheeledGroundVehicle.wav 10 1000  ! Small wheeled utility
vehicles – HMMWV
1 1 225 2 5 * * WheeledGroundVehicle.wav 10 1000  ! Light Armored
Vehicles – Strykers
!
! Default tank sound
1 1 225 2 5 2 12 trckstrt.wav 10 500
1 1 225 1 * * * tank.wav 10 1000
1 1 222 2 * * * tank.wav 10 1000
1 1 222 4 * * * tank.wav 10 1000
1 1 222 * * * * tank.wav 10 1000
!
! Default catch-all
* * * * * * * silent.wav 10 1000
```

CHAPTER 14

Using 3D Characters in VRSG

VRSG has integrated support for high-quality animated 3D characters. The character animation and rendering system can draw hundreds of characters within the field of view while maintaining a high frame rate.

This chapter describes how to animate 3D character models in VRSG.

Overview

VRSG is delivered with a substantial model library of characters and weapons in MVRsimulation's model format. You can immediately configure and use these models in VRSG. Over a thousand .BVH animations for the characters are also included in the library; they portray all commonly used appearances described by the DIS protocol (including multiple deceased positions). VRSG smoothly blends between animations automatically. In addition you can create your own custom characters, weapons, and animations and use them with VRSG.

MVRsimulation's 3D characters consist of an inner skeleton and an outer "skin" mesh. The skin mesh is deformed in real-time according to the position of the skeleton. This technique prevents cracks or interpenetrating surfaces; it also creates realistic looking animations without requiring any excess geometry. In VRSG, the deformation is computed entirely by the graphics card to maximize performance. VRSG computes transitions between animations by smoothly blending from one animation to the next. This approach prevents jarring discontinuities and avoids the burden of creating hand-authored transitions when creating new animations.

VRSG has the ability to animate a character's individual fingers using 16 bone hand models. These hand models with articulated fingers can be used with our character models to simulate the gestures of non-verbal commands and communication, such as tactical combat hand signals. Supporting this feature are 30 BVH animations for the skinned hand models, which were created using Autodesk Motion Builder animation software. The two FBX rigs for these hand models are available upon request.

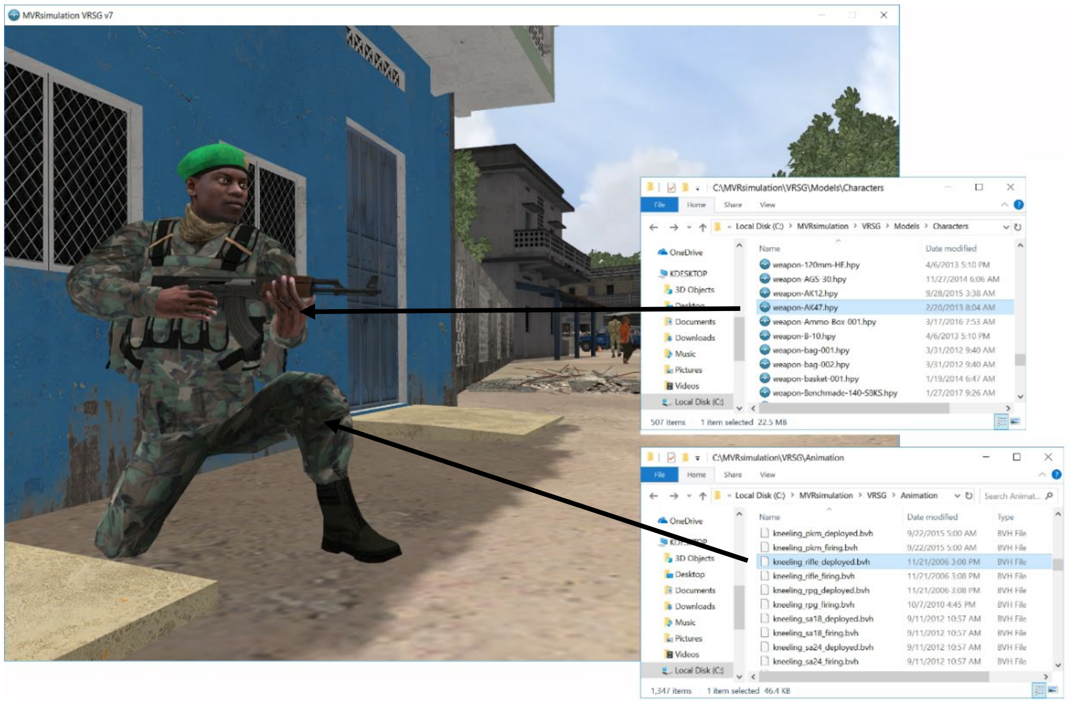
Adding characters to VRSG

As with vehicle entities, you configure 3D character models as entities in VRSG through the ModelMap.ini configuration file.

Animations for these models support the most common life form states used by the DIS protocol. VRSG is delivered with over 1,000 character animation clips in the standard BVH format. Each clip is stored in a separate file in the \MVRsimulation\VRSG\Characters\Animations subdirectory with the .bvh extension. The configuration file AnimationMap.ini is used to associate DIS lifeform entity appearances with specific animation clips.

A weapon or other character accoutrements can be associated with a character. A character with a weapon can respond to DIS protocol messages to deploy or fire its weapon. When an entity's appearance is updated over DIS, an animation is automatically assigned. The character animations are supported with a standard skeleton hierarchy based on MotionBuilder's skeleton hierarchy. For more information about configuring character and their animations in the ModelMap.ini file, see the chapter "Configuring Models and Events." You can also use 3D characters in scenarios you create in VRSG Scenario Editor.

As with cultural features, you can add static characters to a scene in VRSG Scenario Editor or by dragging them from the Windows Explorer to the VRSG visualization window, as described in the chapter "Configuring Models and Events." You can assign weapons and animations to them in the same manner, in either Scenario Editor as part of scenario activity or by dragging the weapon or animation file of interest to the character in the visualization window.



C:\MVRsimulation\VRSG\Models\Characters

Local Disk (C:) > MVRsimulation > VRSG > Models > Characters

Name	Date modified
weapon-120mm-HE.hpy	4/6/2013 5:10 PM
weapon-AGS-30.hpy	11/27/2014 6:06 AM
weapon-AK12.hpy	9/28/2015 3:38 AM
weapon-AK47.hpy	2/20/2013 8:04 AM
weapon-Ammo-Box-001.hpy	3/17/2016 7:53 AM
weapon-B-10.hpy	4/6/2013 5:10 PM
weapon-bag-001.hpy	3/31/2012 9:40 AM
weapon-bag-002.hpy	3/31/2012 9:40 AM
weapon-basket-001.hpy	1/19/2014 6:47 AM
weapon-Benchmade-140-SBKS.hpy	1/27/2017 9:26 AM

507 items 1 item selected 22.5 MB

C:\MVRsimulation\VRSG\Animation

Local Disk (C:) > MVRsimulation > VRSG > Animation

Name	Date modified	Type
kneeling_pkm_deployed.bvh	9/22/2015 5:00 AM	BVH File
kneeling_pkm_firing.bvh	9/22/2015 5:00 AM	BVH File
kneeling_rifle_deployed.bvh	11/21/2006 3:08 PM	BVH File
kneeling_rifle_firing.bvh	11/21/2006 3:08 PM	BVH File
kneeling_rpg_deployed.bvh	11/21/2006 3:08 PM	BVH File
kneeling_rpg_firing.bvh	10/7/2010 4:45 PM	BVH File
kneeling_sa18_deployed.bvh	9/11/2012 10:57 AM	BVH File
kneeling_sa18_firing.bvh	9/11/2012 10:57 AM	BVH File
kneeling_sa24_deployed.bvh	9/11/2012 10:57 AM	BVH File
kneeling_sa24_firing.bvh	9/11/2012 10:57 AM	BVH File

1,347 items 1 item selected 46.4 KB

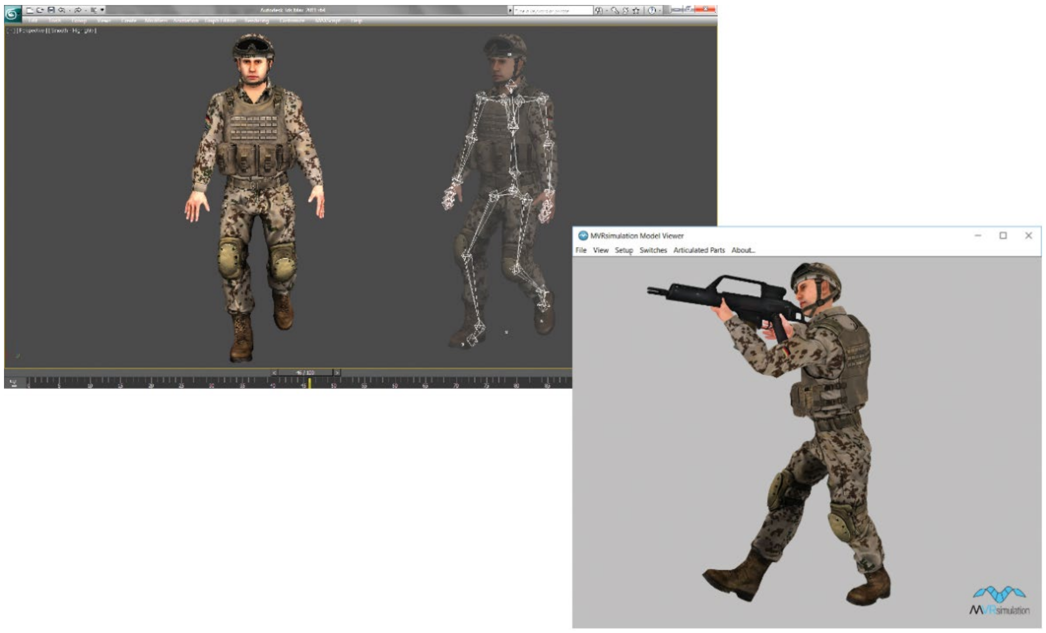
Drag weapon and animation files of interest to the character model in the visualization window.

Creating your own character models

In addition to using the models in MVRsimulation's 3D character library, you can create and texture your own human characters in well-known modeling tools such as Autodesk 3ds Max or Autodesk Maya using MVRsimulation's character template and conversion utility that converts character models from the popular FBX format into MVRsimulation's model format. MVRsimulation's character template includes an armature object containing a hierarchy of bones that matches MVRsimulation's BVH rig, a low polygon mesh that is used by VRSG's physics.

The chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats" describe how to use MVRsimulation's conversion utility to convert FBX models to MVRsimulation's HPX model format. This section describes how to create or modify human character models to use with BVH motion capture files in VRSG.

The following images show the preparation of a model in Autodesk 3ds Max prior to exporting it in FBX format for conversion to MVRsimulation's model format, and subsequent inspection of the converted model in the Model Viewer.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

In the Model Viewer you can test your character with a weapon and with an animation by dragging a weapon and a BVH animation from the MVRsimulation character model library to the Model Viewer window. You can then inspect the animation playing in real time.

About MVRsimulation’s character template

The file mvr_control_rig.fbx included with MVRsimulation's FBX conversion utility contains two assets:

- An armature object containing a hierarchy of bones that match MVRsimulation’s BVH rig.
- A low poly mesh, only used for VRSG's physics, never displayed.

Note: MVRsimulation also has two hand rigs for character animations that require finger articulations, like hand signals. MVRsimulation’s FBX rigs for hand models are available upon request.

Preparing the model for conversion

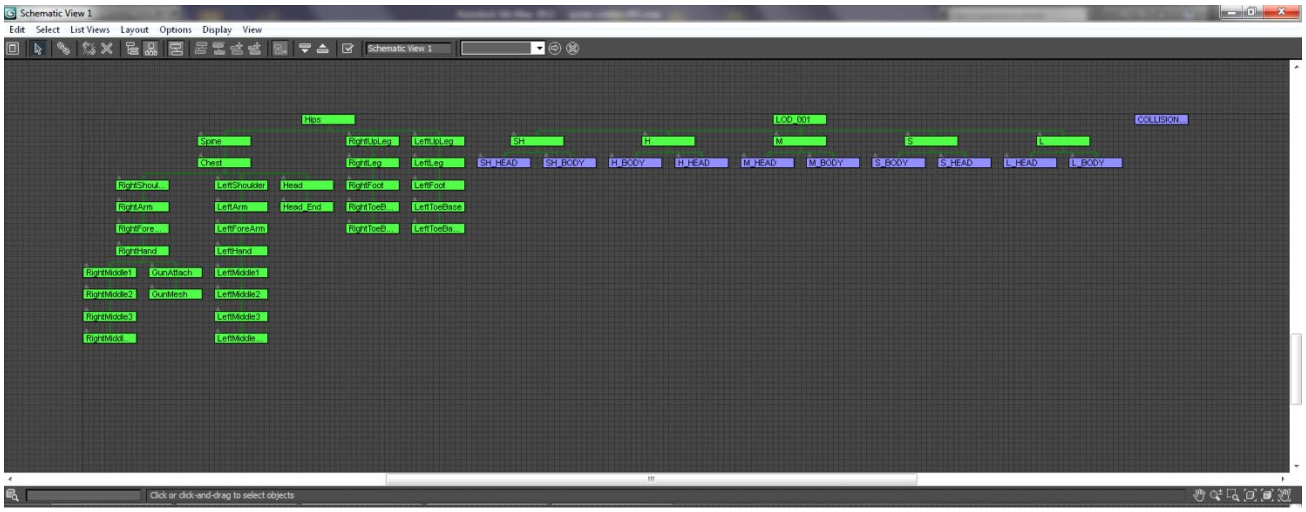
Preparing a model for conversion to HPX format entails opening the model in MVRsimulation’s character template, ensuring the textures imported correctly, and refining the model to match the MVRsimulation armature. There are no special geometry requirements. Character models may contain single or multiple meshes.

To prepare a character model in your modeling tool:

1. Import MVRsimulation's character template into your character's scene.
2. Scale the geometry to the dimensions of the template’s skeleton.
3. Skin the character model to the control rig in the model tool's standard manner. Initially, the mesh(es) should be assigned to the model's highest LOD.

4. Assign vertex weights to the skeleton's bones; do not assign more than three bones per vertex.
5. After you test the character at the highest LOD, copy the skin data to the lower LODs.
6. Create the node hierarchy.
7. Export the character model to FBX format.

The following image shows the typical node hierarchy for a character. This examples shows the node hierarchy in Autodesk 3ds Max of the German soldier character depicted elsewhere in this section.



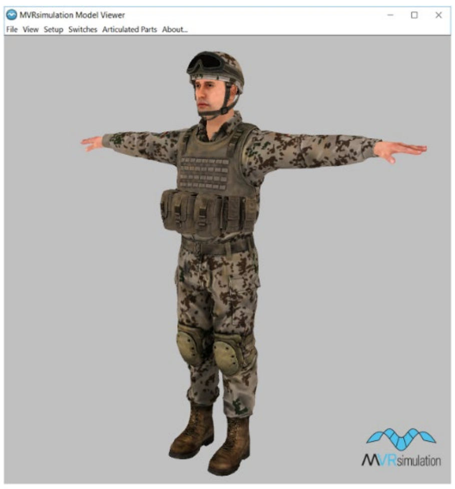
Converting the model to HPX format

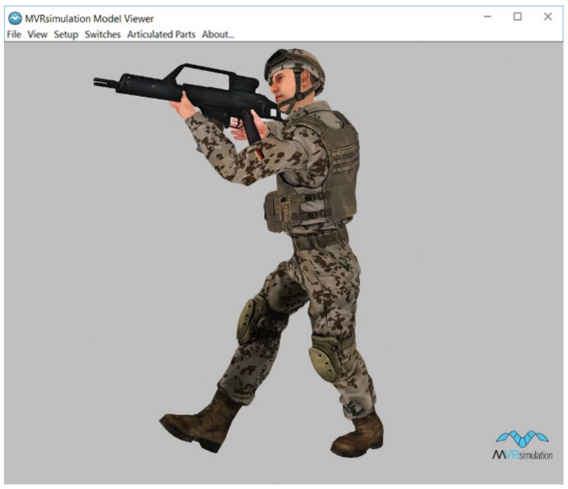
To convert the FBX model to MVRsimulation’s HPX format, run the FBX2HPX utility described in the chapter, “Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats.”

Previewing the model in the Model Viewer

Next you inspect the model in MVRsimulation’s Model Viewer:

1. Open the new HPX model in the MVRsimulation Model Viewer by starting the Model Viewer from the MVRsimulation folder on the Start Menu and then browsing for the model, or by simply double-clicking the model.
2. Test an animation with your character by dragging one of the provided .BVH animations to the Model Viewer window. The animations are located in the \Animations subdirectory of MVRsimulation’s 3D character library.
3. Optionally, preview a weapon or other accessory model with the character; simply drag a weapon model from MVRsimulation’s 3D character library to Model Viewer window. The weapon model will be automatically added as the secondary model and Model Viewer will treat it as the character’s weapon.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

Using MVRsimulation's hand models to animate a character's fingers

VRSG supports animations for finger articulations with six new 16-bone hand models in the character model library. The models, located in \MVRsimulation\VRSG\Models\Other are: skinnedHandLeft-01.hpy, skinnedHandRight-01.hpy, skinnedHandLeft-02.hpy, skinnedHandRight-02.hpy, skinnedGloveLeft-01.hpy, and skinnedGloveRight-01.hpy.

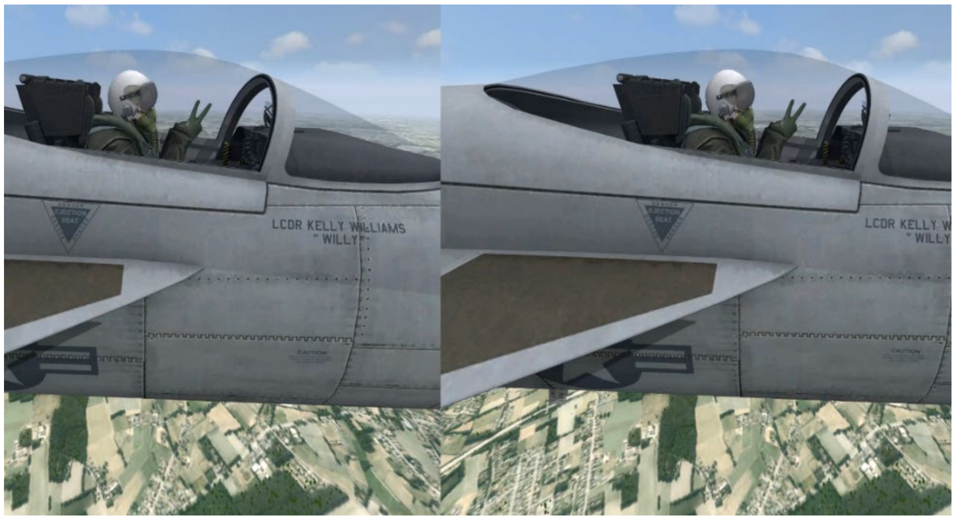
These hand models with articulating fingers can be used with character models to simulate the gestures of non-verbal commands and communication, such as tactical combat hand signals. Supporting this feature are 30 BVH animations for the skinned hand models, which were created using Autodesk Motion Builder animation software. The animations are located in \MVRsimulation\VRSG\Animations\ and are named HandRight*.bvh and HandLeft*.bvh. HandleRightNull.bvh and LeftNull.bvh position the hands in the dealt open-hand position.

You can use the hand models with finger animations by:

- Providing an appearance command in the ModelMap.ini file.
- Sending an appearance EntityStatePDU to trigger the animation.

To use these skinned hand models, a character .hpx or .hpy model must have a switch state 0x20000000, which switches off the hand meshes as the model's hand meshes cannot be rendered along with the skinned hand meshes.

At present two MVRsimulation character models are set up to accept the hand models with articulated fingers: human-pilot-014 and human-us-soldier-sof-033.



VRSG real-time screenshot taken in an HTC-VIVE Pro headset of character finger-based signal animation. The pilot character model is giving a hand signal for the Naval Air HEFOE code indicating the aircraft is having trouble with the electrical system.

For example, the above image is taken from a scenario that has the following lines in its ModelMap:

```
  1   2 225   0   0   0  23 Vehicle FA-18E.US.grey.hpy
  3   1 225   0   0   0  15 Human human-pilot-014.hpy -skinnedGloves
```

The `-skinnedGloves` command associates the specified character model with the ungloved hand models with articulated fingers

Two FBX rigs for these hand models are available from MVRsimulation upon request for creating your own finger/hand animations.

Editing an MVRsimulation animation

Sometimes you might want to edit the syntax of an MVRsimulation .bvh animation to make small adjustments, for example to reposition a gunner animation to have a character use it as a second gunner or to apply the animation for use in another vehicle.

To do this, you can make a copy of the animation of interest, give it a new name for the new vehicle and position.

The BVH files are human-editable ASCII files, which means that making a simple edit like moving the character position is relatively easy.

For example, in the bottom of each character animation for use with a vehicle are the following three lines:

```
Frames:    5
Frame Time:    0.0333333
-60 75 -50 0.117927 0.301847 [lots more numbers...]
```

Most animations have more than 1 frame even for a stationary character. These frames are for subtle motions like moving the character’s head around slightly. You could change the Frames number to 1, so you only have to edit one row, or you can edit every row if you want to retain that motion in there.

The first three numbers are the hip joint position; they position the character's relative to the vehicle. The units are in centimeters, and the axes are:

positive X - points to the right side of the vehicle.

positive Y - points to the up of the vehicle.

p*ositive Z* - points to the back of the vehicle.

By editing these three numbers you can reposition the character. You can use Model Viewer to preview the resulting animation with both the vehicle and the character. (After you edit the animation file, open Model Viewer and display both the vehicle and character, and then drag-n-drop the .bvh onto Model Viewer to see the result.) For more information about previewing models and animations in the Model Viewer, see the chapter "Previewing Models, Effects, and Terrain."

Creating your own animations

MVRsimulation characters use the MotionBuilder standard skeleton hierarchy for animation. MotionBuilder is one of the strongest animation tools available, ideal for building character animation for simulations and games.

This approach enables you to import motion-capture data or use software package to create custom animation, and in turn use MotionBuilder to transfer your data to MVRsimulation's skeleton hierarchy.

MVRsimulation's 3D character model library includes a control rig in MotionBuilder's FBX format, which contains the standardized skeleton used in all of MVRsimulation's character models. You can use this .fbx file as the starting point for any new animations. Once your data is in MotionBuilder, you can easily create and optimized looped animation and export it in BVH format for use in VRSG. Then you list your .BVH animation in the file AnimationMap.ini, which is located in \MVRsimulation\VRSG\Animations, to tie it with an appearance code. (See the chapter "Configuring Models and Events" for information about using Animation.ini.)

This section describes, in step-by-step form, how to create a character animation for VRSG version 5 in MotionBuilder Pro, using the MVRsimulation control rig. The example shows how to create a "brake signal" hand signal animation for one of the carrier-deck-crew character models in MVRsimulation's 3D character model library. The animation uses for a reference the web site http://navysite.de/cvn/catcom.htm, which contain video clips of the most common signals used by personnel on an aircraft carrier's flight deck.

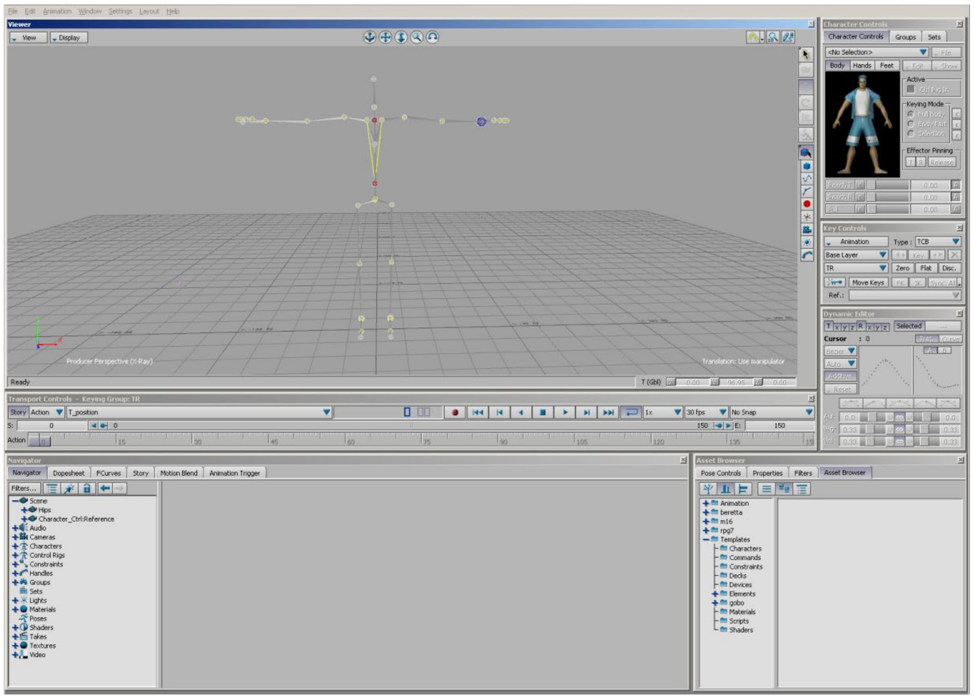
The steps cover basic keyframe animation manipulation of the control rig using MotionBuilder features such as automatic floor detection and finger controls. This example assumes you have basic knowledge of MotionBuilder.

By following the steps in this example, you should be able to use MVRsimulation's control rig to create a basic animation, export the animation to BVH format, and preview the result in MVRsimulation Model Viewer with any character model distributed with VRSG.

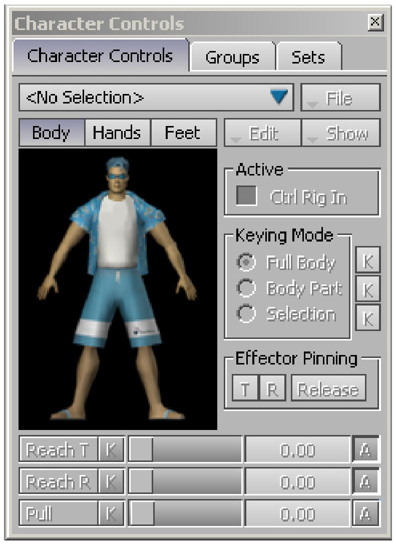
Getting started

To start creating the animation:

1. Locate the file \MVRsimulation\VRSG\Animations\mvr_control_rig.fbx and open it in MotionBuilder Pro. The MVRsimulation standard skeleton appears; this is the skeleton used for all character models in MVRsimulation's 3D character library.



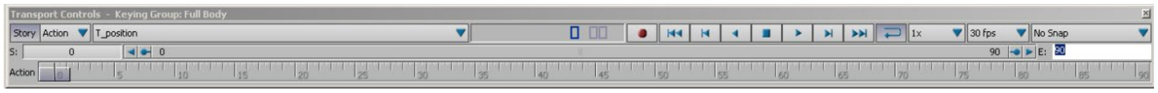
2. Choose Layout > Editing, or press Ctrl-Shift-3.



3. From the Character Controls drop down menu, select Character. A circle appears on each part of the model that can be animated. You should only use the Character Controls to

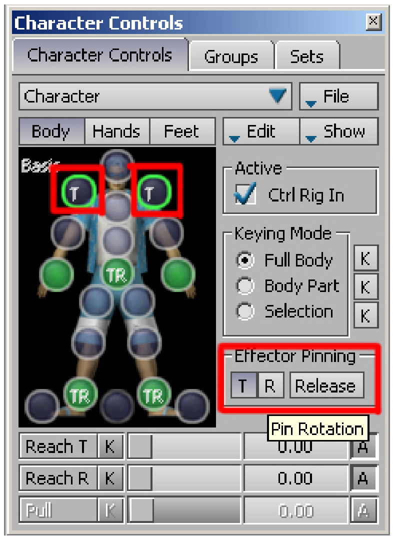
select the part of the skeleton you want to animate — to minimize the chance of putting keyframes on the wrong body parts or on a skeleton joint not used in VRSG.

4. In the Transport Controls menu, type the number of frames required to make the entire animation. This example requires 3 seconds for 90 frames (1 second = 30 frames).



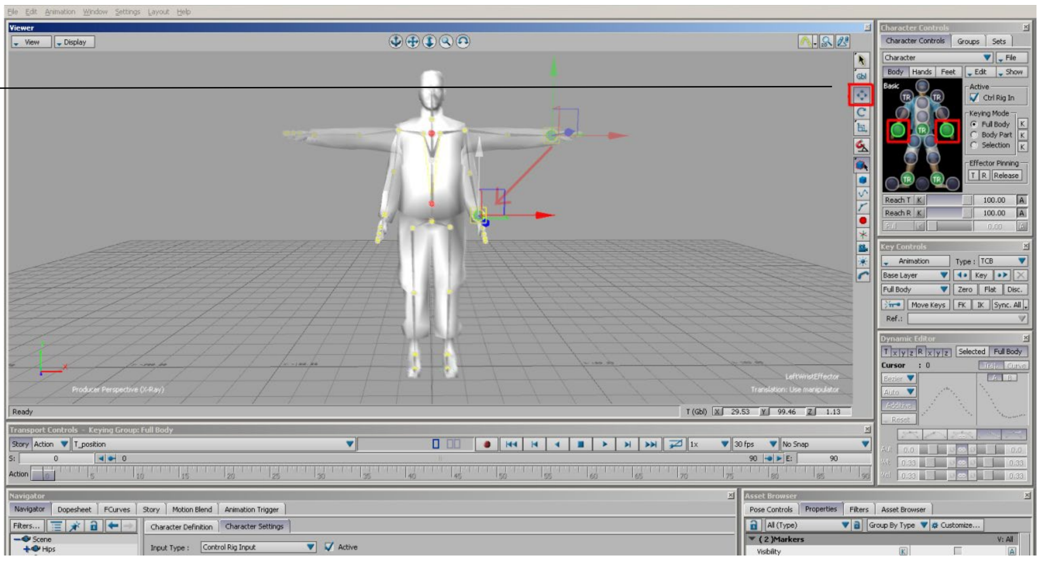
The animation will have the character standing in a natural pose and then performing the hand signal. Before animating both arms, you must lock the shoulder position.

5. From the Character Controls menu, select both shoulders (using the Shift key) and choose Effector Pinning. (In MotionBuilder, you can lock translation and rotation on selected joints so that they are not affected by the transformation.)



6. Select both hands and lower the arms with the Translate tool as shown in the following image:

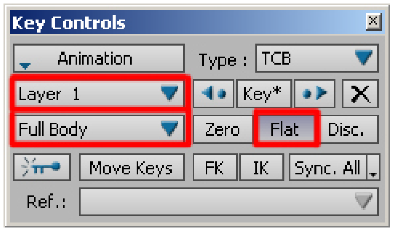
The translate tool.



Building the animation

Now you can assign the first keyframe, which will consist of the starting pose of your character animation.

1. On the Key Controls menu, select “Layer 1” and “Full Body” and then click Flat to place a keyframe for the entire control rig. (MotionBuilder‘s concept of layers simplifies the modification of motion capture data.)

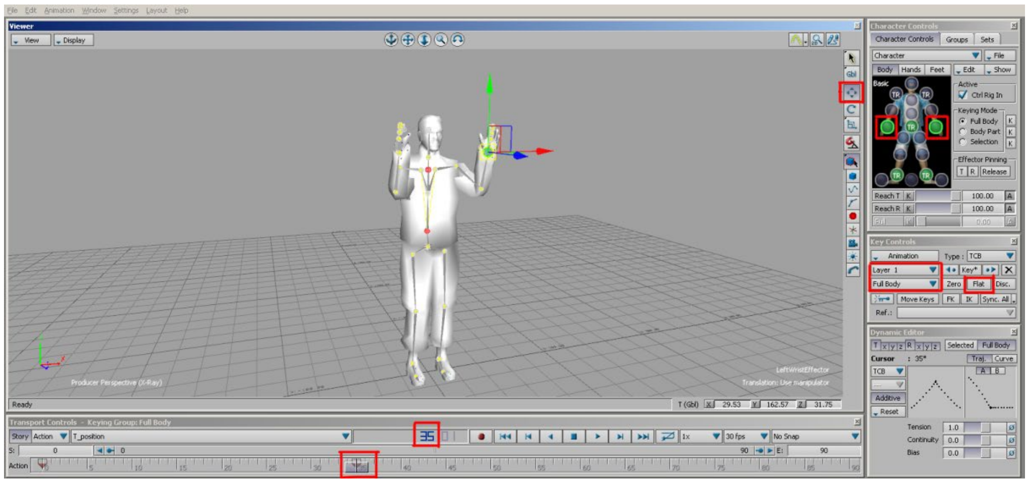


The “brake” signal consists of bringing both hands in front of the character’s face and forming a circle with the thumb and finger. The next keyframe will consist of bringing both arms forward and raising the hands at head level.

2. Using the Transport Controls, move the time slider to the keyframe 30.

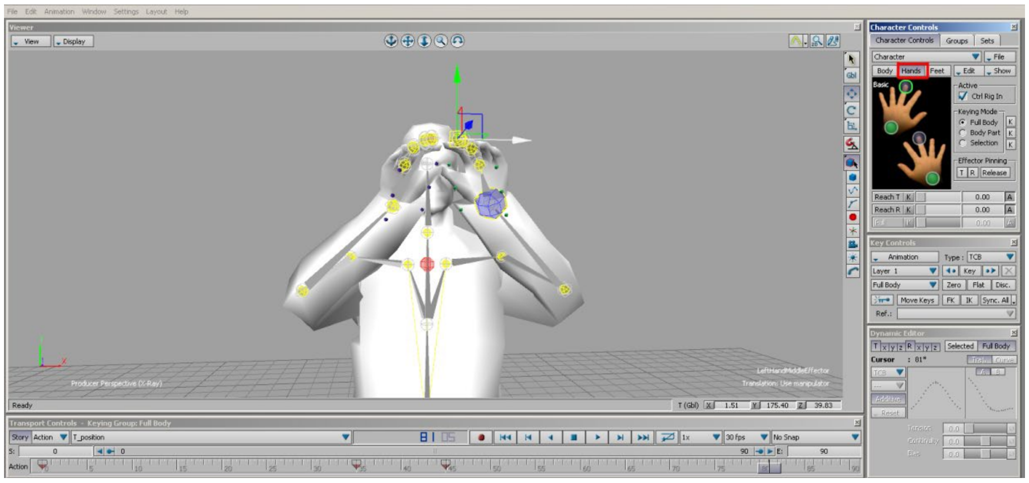
3. In the Character Controls panel, select the two green circles for on the hands, and raise both arms upward and a bit in front of the character at head level.

4. In the Key Controls, press Flat to assign a keyframe. Next you will position both hands in front of the face and form a circle with the fingers.



5. Move the time slider to frame 45 and position both hands in front of the face as shown below.

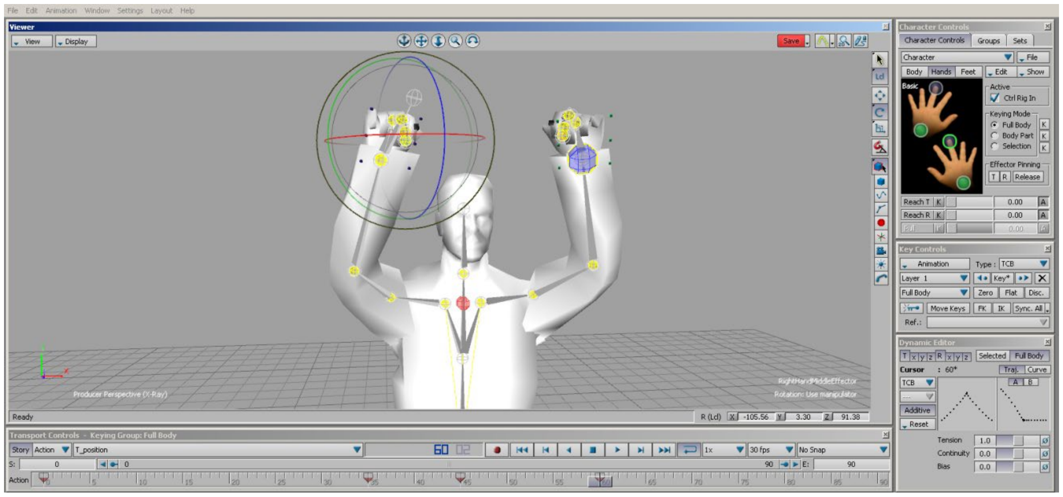
6. In the Character Controls Hand tab, position the fingers properly.



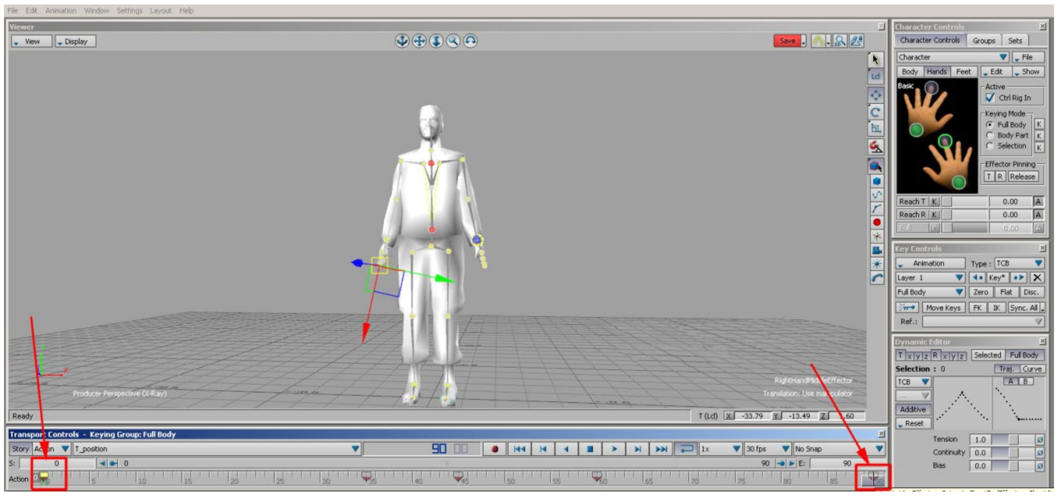
7. When you are satisfied with the hand positions, assign a keyframe.

8. Move the time slider to frame 60 and modify the pose by raising both arms upward in the Character Controls Body tab, and close the hand using Character Controls Hand.

   To finalize the process of keyframing your animation you copy the first keyframe you made to the end of the animation.



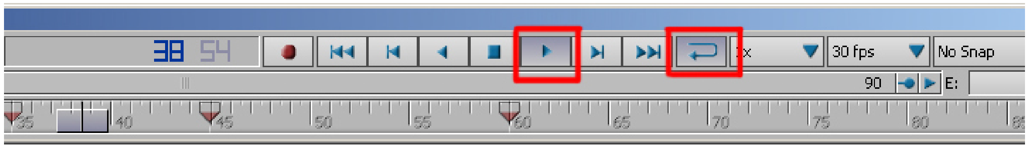
9. Go to time frame 90, select the first keyframe in Transport Controls and press Ctrl-C and Ctrl-V to copy and paste the first frame to the last keyframe.



Select the first keyframe and copy it by pressing Ctrl-C.

At time frame 90, paste the keyframe by pressing Ctrl-V.

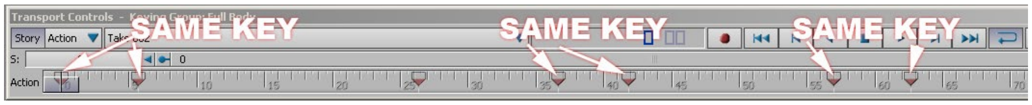
You have now a loop in your animation. You can preview the animation in Motion Builder using the Transport Controls Play button and activate the Loop button.



Suppose that by playing the animation, you notice that a certain position should be held a bit longer. This modification is fairly easy to do.

10. Locate the appropriate keyframe in the Transport Controls time bar, select it, go to a few keyframes that are performed later in time, and paste that pose.

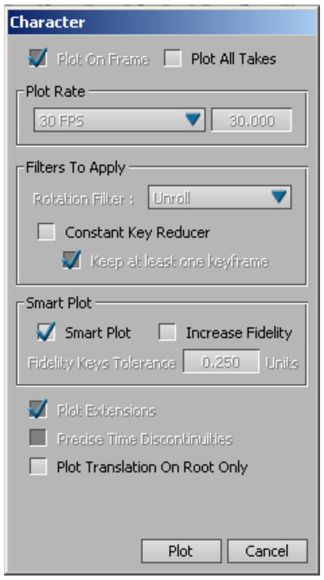
The image below shows the final keyframe layout:



Exporting the animation to BVH format

You can now create the final BVH animation and preview your animation in MVRsimulation's Model Viewer.

1. In the Navigator window, expand the Character branch in the tree panel and double-click on Character. To the right is a menu containing information about the MVRsimulation control rig setup.
2. Select Plot Character and when the Character message box appears, click Skeleton.
3. In the next dialog box that appears, select Plot All Takes \Characters\Runtime\Animations and then click OK.



Character

[✓] Plot On Frame   [ ] Plot All Takes

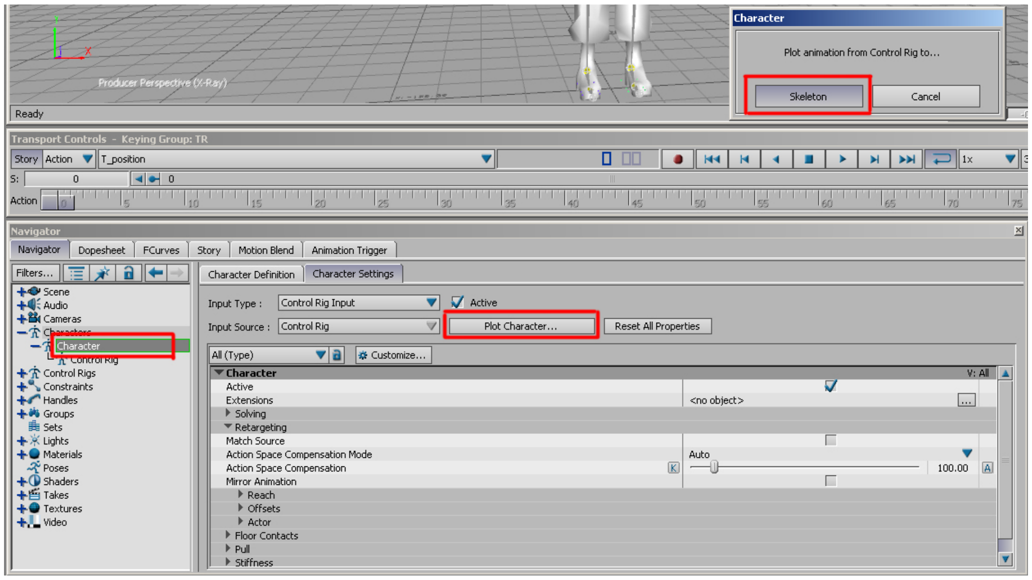
Plot Rate
30 FPS   30.000

Filters To Apply
Rotation Filter : Unroll
[ ] Constant Key Reducer
[✓] Keep at least one keyframe

Smart Plot
[✓] Smart Plot   [ ] Increase Fidelity
Fidelity Keys Tolerance 0.250 Units

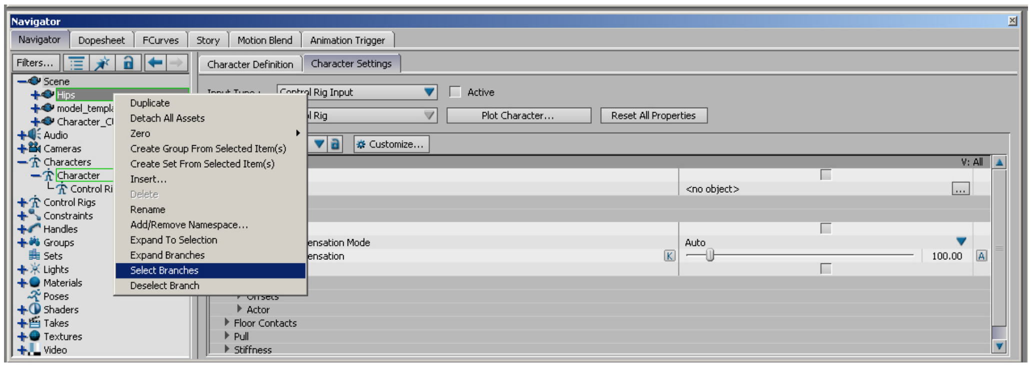
[✓] Plot Extensions
[ ] Precise Time Discontinuities
[ ] Plot Translation On Root Only

This action transfers your animation to the MVRsimulation skeleton in preparation to export the animation in BVH format.



To export the .bvh file:

1. Select the skeleton hierarchy in the Navigator window.
2. Choose Scene > Hips and then right-click to choose Select Branches.



3. Choose File > Export.
4. When the Export Files dialog box appears, select Biovision (.bvh) as the file type, and then click Save.

The process described in these steps was used to create and export the following animations located in the \MVRsimulation\VRSG\Animations subdirectory:

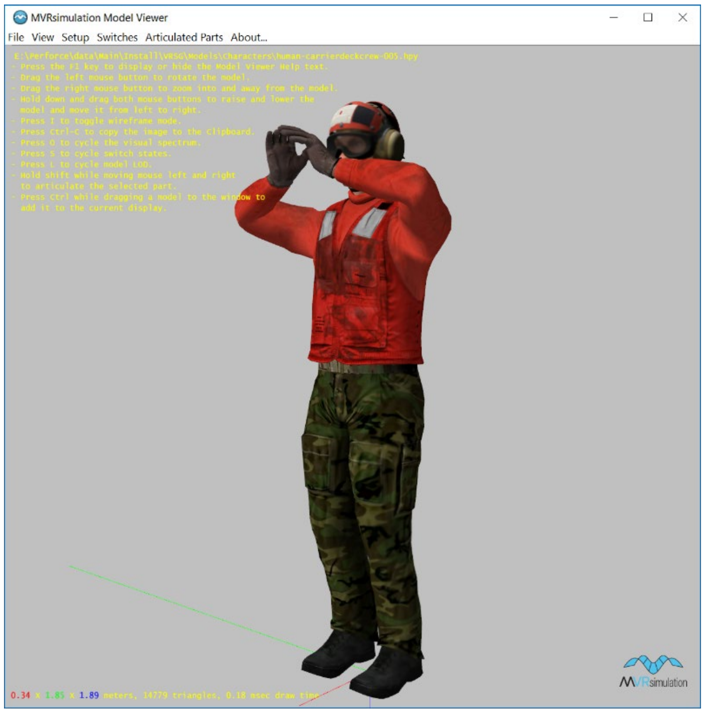
flightdeck-turn_right-signal.bvh	flightdeck-slow_down-signal.bvh
flightdeck-turn_left-signal.bvh	flightdeck-move_foward-signal.bvh
flightdeck-jet_blast-signal.bvh	flightdeck-brake-signal.bvh
flightdeck-pull-signal.bvh	flightdeck-full_power-signal.bvh
flightdeck-move_foward-signal.bvh	

Previewing the animation in the Model Viewer

Next you can preview the animation with a model in MVRsimulation's Model Viewer.

Double-click the human-carrierdeckcrew-005 model located in the 3D character library.

From the Windows Explorer, drag the crew_carrier_brake_signal.bvh BVH file onto the Model Viewer window. Doing so plays the BHV animation with the current model displayed in the Model Viewer.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

```
E:\Perforce\data\Main\Install\VRSG\Models\Characters\human-carrierdeckcrew-005.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press Ctrl-C to copy the image to the Clipboard.
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.

0.34 x 1.85 x 1.89 meters, 14779 triangles, 0.18 msec draw time
```

Using First Person Simulator (FPS) to simulate a dismounted infantry character

When you have human characters in a networked VRSG scene, you can use the VRSG First Person Simulator™ (FPS) to control your characters and view the scene or exercise from their point of view.

FPS creates a simulation of a dismounted infantry character within. With FPS, you can use the recommended Logitech F310 gamepad or a standard joystick to simulate many dismounted infantry activities such as:

- Moving throughout the virtual world at realistic speeds, assuming a variety of postures.
- Entering buildings, climbing stairs.
- Selecting and firing a user-defined set of weapons.
- Accepting damage from a user-configurable set of munitions.
- Laser ranging or designating to support JTAC/CAS missions.

The rest of this chapter describes how to use FPS, including how to configure it and how to use it in FPS-JTAC (or FAC) mode.

Configuring FPS data files

To configure the behavior of the locally simulated character, you edit the file FPS.ini. This ASCII data file, which is used by FPS, controls a character's available weapons, appearance, reincarnation period, and how the character reacts to damage from munitions from remote entities. The file is located in the directory \MVRsimulation\VRSG. The entries and keywords for configuring these weapons and behaviors are described in this section.

Defining weapons

Accessories carried by human characters are attached as weapons (though they can be other kinds of objects too such as a shovel, cell phone, binoculars, and so on). In the file FPS.ini, use the keyword *define_weapon <weapon-name>* to begin the description of a weapon; use the *end_weapon* keyword to end it. You can define as many weapon types as you like for the simulated character. The following is an example of a weapon description from FPS.ini that defines an M16 rifle:

```
define_weapon m16
   num_clips:          5
   rounds_per_clip:    30
   rounds_per_second:  10
   dis_enumeration:    2:8:225:2:1:5:0
   hpx_model:          weapon-M16.hpx
end_weapon
```

The *hpx_model* keyword is used to define which visual model to associate with the FPS character when it is in third-person view. See the chapter "Configuring Models and Events" for information about associating character models and weapon models with enumerations for DIS life form entities.

The *dis_enumeration* field defines the enumeration that will be applied to DIS FirePDUs and DetonationPDUs when rounds are fired from the FPS entity. The *num_clips* and *rounds_per_clip* defines the initial ammo load. The *rounds_per_second* keyword defines the firing rate when in semi-automatic mode.

Defining damage

Use the *damage_entry* keyword in FPS.ini to enter an entry into the simulator's damage table. A damage entry consists of a 7-tuple DIS enumeration defining the munition, distance from the character, and a probability of damage associated with that munition and range combination. For a given munition type, list the entries in order of increasing range. The following entries illustrate the damage entries for an M16 round:

```
damage_entry: 2:8:225:2:1:5:0 0.4 0.80
damage_entry: 2:8:225:2:1:5:0 1.0 0.50
damage_entry: 2:8:225:2:1:5:0 2.0 0.25
```

In the above example, M16 rounds that strike the character within 0.4 meters will have an 80% probability of deactivating, or "killing" the local character. M16 rounds that strike the character within 1 meter will have a 50% chance of killing the character.

If the FPS character receives a DetonationPDU for an enumeration that is not described in FPS.ini, the DetonationPDU will be ignored and will not affect the FPS character.

Setting the reincarnation period

You can set a restoration, or reincarnation, period for a character using the *reincarnate_period* keyword. You specify the period in seconds. For example:

```
reincarnate_period: 10
```

If the character is deactivated during the simulation (that is, "killed"), it will be fully restored, and all weapon stores will be restored, following the reincarnation period.

Setting the character's appearance

Use the *character* keyword to define the appearance of the FPS character while it is in third person mode; for example:

```
character: human-us_soldier-022.hpy
```

This keyword does not affect the display of remote characters. See the chapter "Configuring Models and Events" for information about associating character models and weapon models with enumerations for DIS life form entities.

Configuring FPS for simulating JTAC missions

FPS contains enhancements that support forward area controller (FAC)-type actions performed by Joint Terminal Attack Controllers (JTACs) in simulations between aircraft and ground systems. These collaborative missions are also known as Close Air Support (CAS) or Forward Air Control (FAC) missions. Among the enhancements are the following keywords available in FPS.ini:

```
melios_fov: <fov-in-degrees>
laser_codes: <code1> <code2> ... <codeN>
```

For example:

```
melios_fov: 8
laser_codes: 1 2 3 4 5
```

melios_fov sets the FOV angle when in Designator/Rangefinder (or CAS), mode
laser_codes enables you to specify a list of possible laser codes to choose from.

In Designator/Rangefinder mode, VRSG displays a generic laser range finder reticle, and uses the *melios_fov* field of view to simulate the magnified optics of a laser rangefinder device. Laser ranging or designating is only possible in Designator/Rangefinder mode.

You can customize the mapping of buttons on simulated military devices, such as NVIS Ranger 47 simulated military device, for use for laser ranging and target designation training. You map the buttons to their functions in the FPS.ini file. For example, you can add the following lines to the FPI.ini, which are VRSG's default button assignments for simulated military devices:

```
designate_ button: 1
range_button:      2
coord_button:      3  ! toggles the coordinate system (such as MGRS
                        vs Geodetic)
spectrum_ button:  4  ! cycles through the sensor views(Visual, EO,
                        IR, and NVG).
```

Press F7 on the keyboard for boresight. Press Shift-F7 instead to reverse boresight if the unit has a reverse mounted tracker, as in the NVIS Ranger 47 simulated military device. (A reverse mount will be obvious if the pitch seems to be backward.)

Although adding the above entry of default button assignments is not necessary when you use one device, unless you want to change the button-function mapping, for situations in which you use multiple devices, you can assign buttons from different devices to the same FPS function. Doing so helps when you need to switch between devices (for example, between the Battlespace Simulations' emulated SOFLAM

and NVIS Ranger 47 devices, which are described in the chapter "Using VRSG With Trackers, HMDs, and Simulated Military Devices").

To implement the multi-device button mapping, hook up your simulated military, device, open the device's "joystick" properties applet and make note of which physical buttons on the device correspond to which "joystick" button numbers. Then, in the VRSG FPS.ini file, enter a comma-separated list of button numbers to specify which functions they activate:

```
designate_button: <which buttons designate>

range_button: <which buttons range>

coord_button: <which buttons toggle the coordinate system display
               (UTM vs geodetic)>

spectrum_button: <which button cycles sensor modes (visual, IR, or
                  NVG)>
```

Adding FPS entries to other data files

When you define weapons types for a local character, you should modify the following VRSG data files to ensure appropriate entries exist for the enumerations of the weapons you have defined:

- \Effects\FireMap.ini
- \Effects\DetMap.ini
- \Models\ModelMap.ini

For more information about these .ini files, see the chapter "Configuring Models and Events."

Using FPS

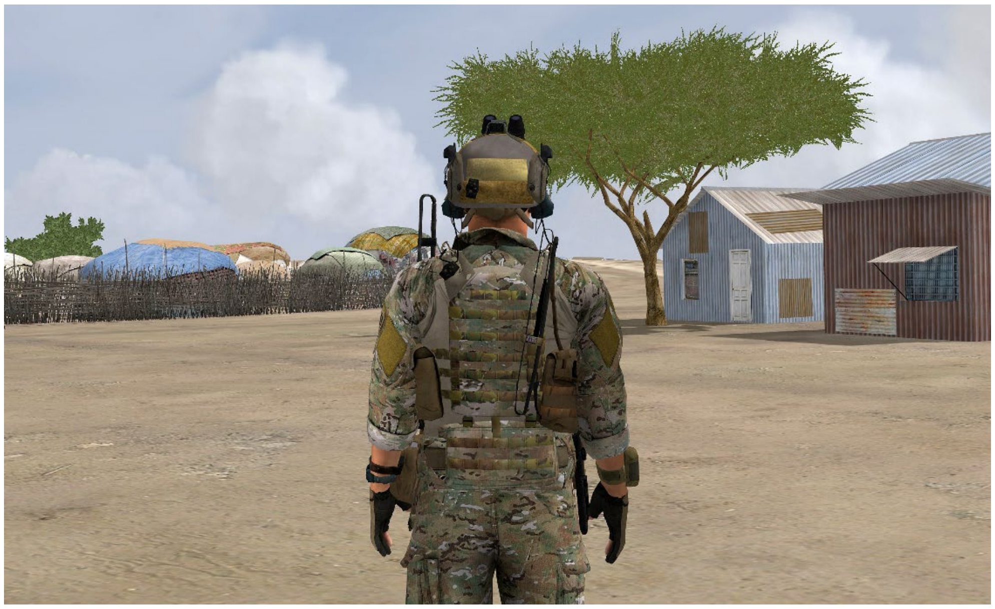
Once you have finished configuring the data files as described, you are ready to use FPS.

FPS initializes itself to the current location of the eyepoint but at ground level. Thus, before activating FPS you can navigate to the intended starting location by selecting stored viewpoints, or by using the 6DOF controller.

To activate FPS within VRSG and create the locally simulated character:

Press the F9 key on the keyboard. The F9 key is a toggle for activating and deactivating FPS mode. To exit from FPS mode and return to normal VRSG operation, press F9 again.

When you activate FPS, the default FPS character, the JTAC soldier model, appears:

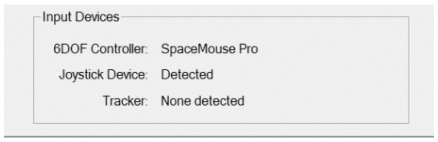


Manipulating the FPS character with a gamepad

When you have the FPS character displayed in the VRSG visualization window, use either the Logitech F310 (recommended) or discontinued Dual Action gamepad, or a joystick to move the character in the virtual world. VRSG works with any 8-button 4-axis input device for which the vendor provides a Windows Joystick driver. For ease of use in manipulating an

FPS character, MVRsimulation strongly recommends the Logitech F310. The device is supported by Windows and VRSG without the need to install any additional software.

Note: If the FPS default infantry character does not appear when you turn on FPS mode, make sure that your gamepad or joystick is plugged in. Also, check that the F-lock key on your keyboard is active. Upon startup, VRSG indicates on the Startup Parameters tab of the Dashboard whether it has detected the gamepad or other joystick device:

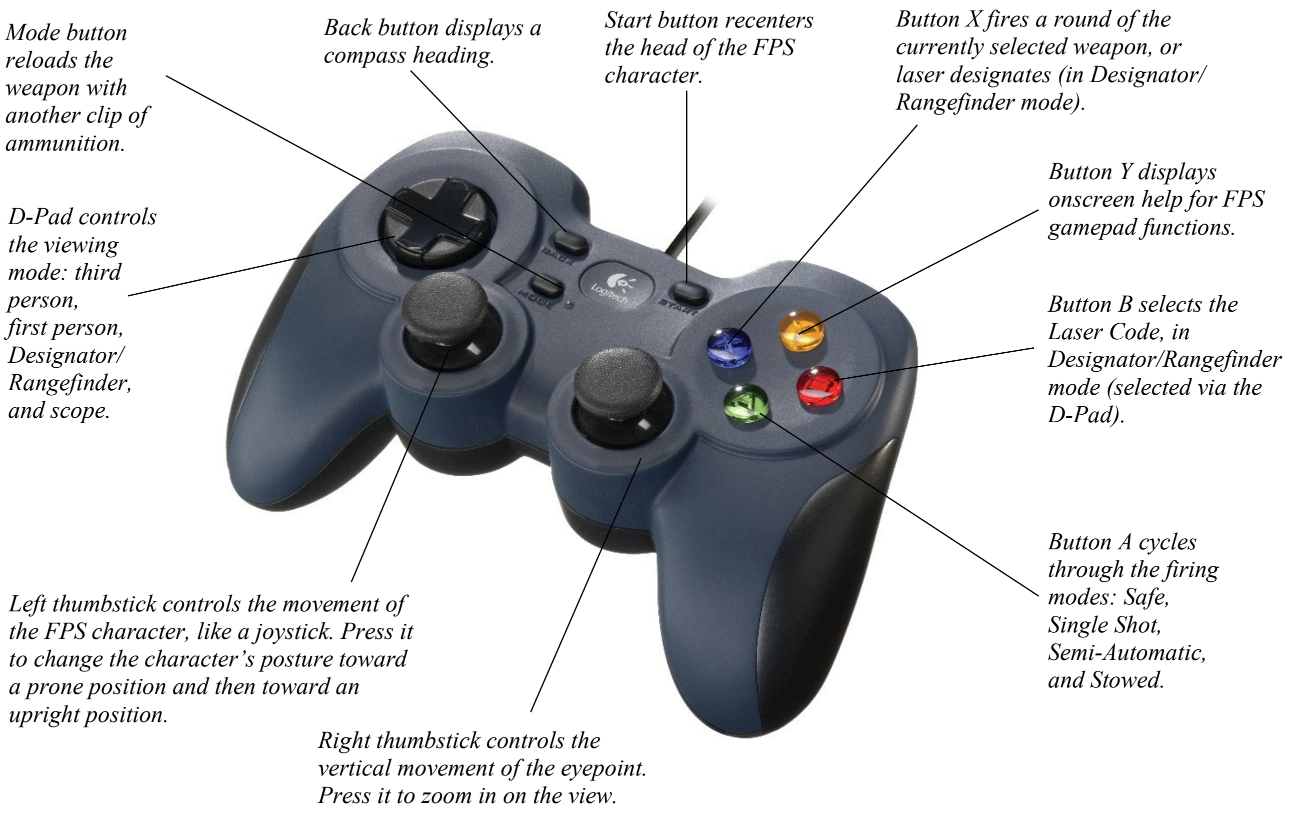


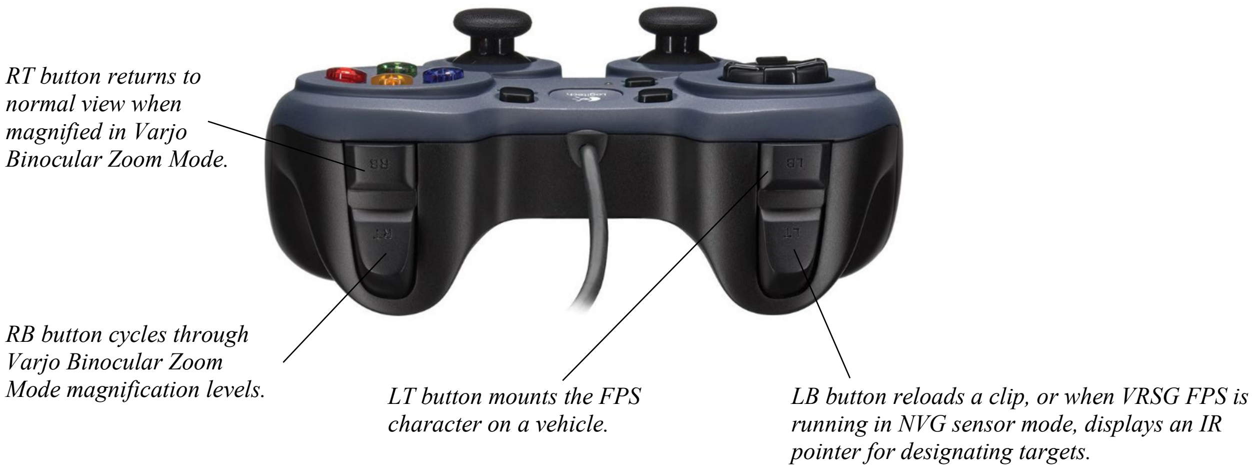
Input Devices

6DOF Controller:	SpaceMouse Pro
Joystick Device:	Detected
Tracker:	None detected

If VRSG has not detected the gamepad or other joystick, check the Device Manager to ensure Windows has detected the device.

The following figures show the mapping of VRSG FPS button functions for the Logitech F310 Gamepad.


Mode button reloads the weapon with another clip of ammunition.
Back button displays a compass heading.
Start button recenters the head of the FPS character.
Button X fires a round of the currently selected weapon, or laser designates (in Designator/ Rangefinder mode).
Button Y displays onscreen help for FPS gamepad functions.
D-Pad controls the viewing mode: third person, first person, Designator/ Rangefinder, and scope.
Button B selects the Laser Code, in Designator/Rangefinder mode (selected via the D-Pad).
Button A cycles through the firing modes: Safe, Single Shot, Semi-Automatic, and Stowed.
Left thumbstick controls the movement of the FPS character, like a joystick. Press it to change the character's posture toward a prone position and then toward an upright position.
Right thumbstick controls the vertical movement of the eyepoint. Press it to zoom in on the view.


RT button returns to normal view when magnified in Varjo Binocular Zoom Mode.
RB button cycles through Varjo Binocular Zoom Mode magnification levels.
LT button mounts the FPS character on a vehicle.
LB button reloads a clip, or when VRSG FPS is running in NVG sensor mode, displays an IR pointer for designating targets.

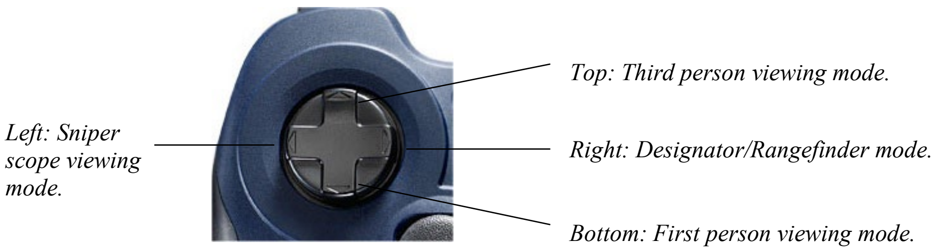
The left thumbstick's x and y axes provide forward and lateral motion to the simulated character. Pushing the thumbstick forward causes the character to move forward at a speed consistent with the current posture of the character. (Bear in mind that the character requires time to accelerate to the intended speed indicated by the thumbstick position.) Pressing the thumbstick in a direction causes the simulated character to change its direction of travel. Pressing the thumbstick down changes the character's posture toward a prone position and then toward an upright position.

Description of VRSG FPS functions

The left thumbstick controls the movement of the FPS character, like a joystick. Press down on the thumbstick to change the character's posture. Initially pressing it increases the FPS character's posture toward a prone position, and then toward an upright position. If the character is in an upright position, a single press transitions the character into a kneeling posture. Likewise if the character is kneeling, a single press transitions the character into a prone posture. Once prone, each subsequent press increases the FPS character's posture toward an upright position.

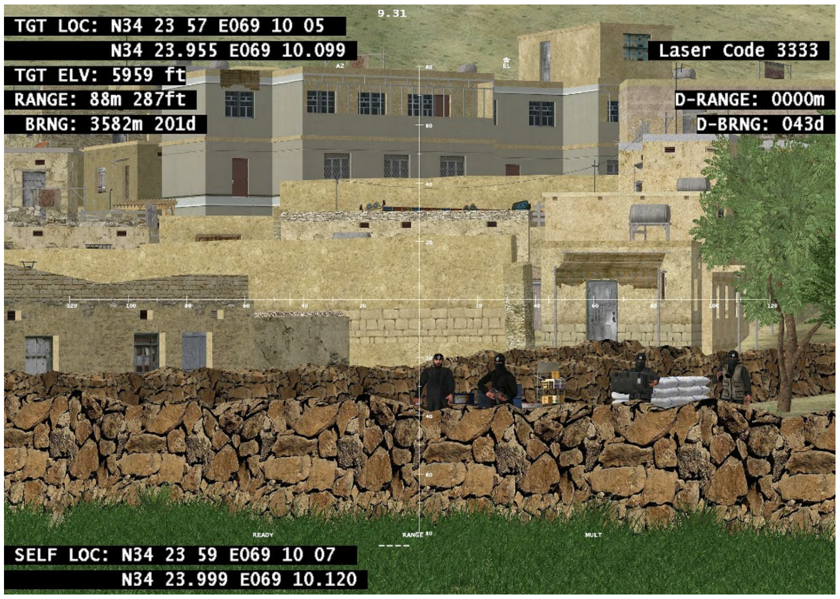
The right thumbstick controls the viewing angle (head angle) relative to the FPS character's body. A left or right motion moves the head in azimuth, while an up or down motion moves the head in elevation. Press Button 10 to return the head to the neutral position. Press down on the right thumbstick to magnify/zoom in on the view.

The D-pad controls the viewing mode. FPS is initialized in a third-person mode where the eye point is behind the simulated character. Pressing the D-pad switches between third-person, first-person, Designator/Rangefinder, and scope viewing modes. On the gamepad, each side of the D-pad controls a view, as shown:


Top: Third person viewing mode.
Left: Sniper scope viewing mode.
Right: Designator/Rangefinder mode.
Bottom: First person viewing mode.

Button X on the F310 gamepad fires a round of the currently selected weapon, or laser designates (in Designator/Rangefinder mode). If fired, the target will display a shot effect from the fired shot.

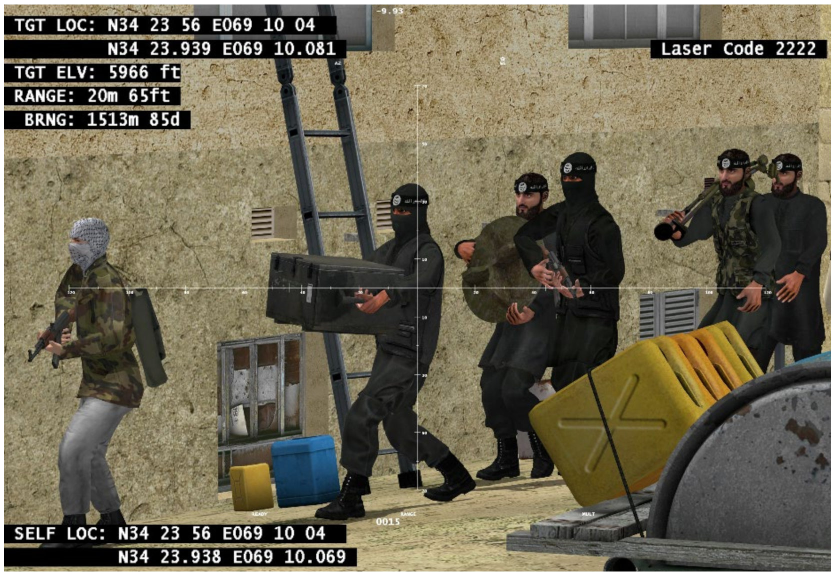
Button A on the F310 gamepad cycles through the set of available firing modes: Safe, Deployed, Single Shot, Semi-Automatic, and Stowed (disarmed). In Safe mode, you are prevented from firing. In Single Shot mode, pressing Button 1 will fire a single round. In Semi-Automatic mode, rounds will be fired at a constant rate while the trigger is pulled. The rate of fire for Semi-Automatic mode is specified in the file FPS.ini, as described earlier. Stowed is a disarmed state that supports a DIS weapon state where the character has a weapon, but is not currently holding the weapon. When you are in Designator/Rangefinder mode (via the D-pad) and range a target, pressing the A button will display a generic reticle and differential range and bearing; the range from the last time you ranged.


```
TGT LOC: N34 23 57 E069 10 05
         N34 23.955 E069 10.099
TGT ELV: 5959 ft
RANGE: 88m 287ft
BRNG: 3582m 201d
Laser Code 3333
D-RANGE: 0000m
D-BRNG: 043d
SELF LOC: N34 23 59 E069 10 07
          N34 23.999 E069 10.120
```

In the upper right, D-RANGE: shows the relative range to last lased position, in meters.

D-BRNG: shows the relative bearing to last lased position, in degrees.

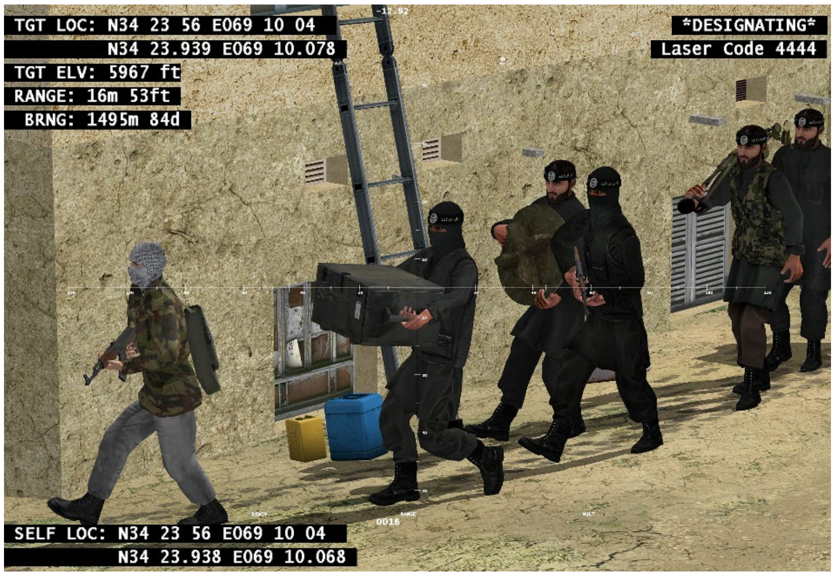
Button B on the F310 gamepad displays a generic reticle selects the Laser Code, when you are in Designator/Rangefinder mode (via the D-pad). Additional overlay information is displayed for 20 seconds, as shown:



```
TGT LOC: N34 23 56 E069 10 04
         N34 23.939 E069 10.081
TGT ELV: 5966 ft
RANGE: 20m 65ft
 BRNG: 1513m 85d
                                        Laser Code 2222
SELF LOC: N34 23 56 E069 10 04
          N34 23.938 E069 10.069
```

Button B on the F310 gamepad cycles though the set of laser codes defined in the FPS.ini file.

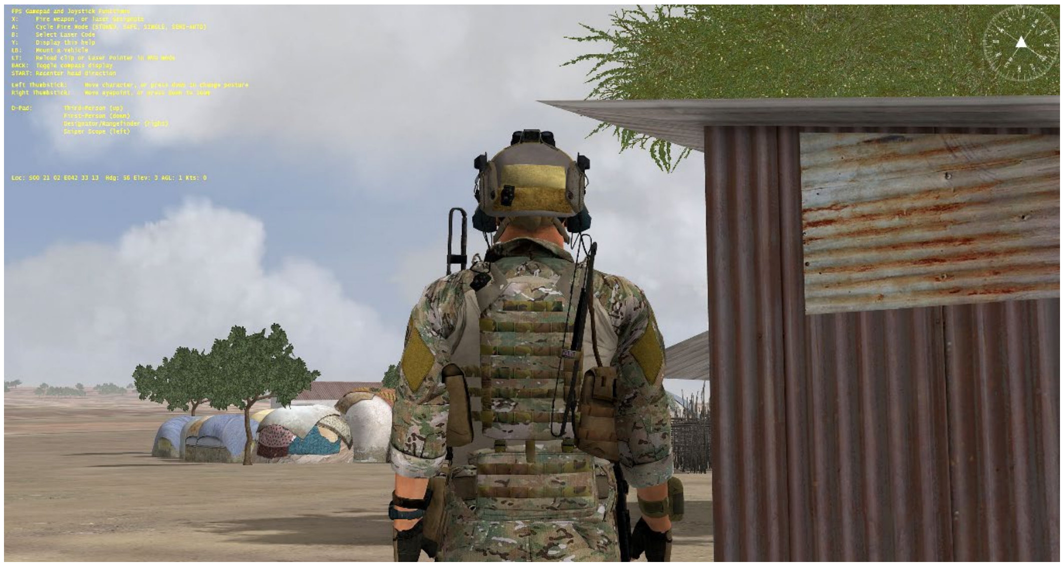
Press Button X on the F310 gamepad to fire the designating laser. It displays the same overlay information as above as well as the 'Designating' indicator, as shown:



```
TGT LOC: N34 23 56 E069 10 04
         N34 23.939 E069 10.078
TGT ELV: 5967 ft
RANGE: 16m 53ft
 BRNG: 1495m 84d
                                        *DESIGNATING*
                                        Laser Code 4444
SELF LOC: N34 23 56 E069 10 04
          N34 23.938 E069 10.068
```

While designating, VRSG continuously outputs DesignatorPDU messages containing the selected laser code. (However, while simply ranging, VRSG will not output a Designator PDU.) If the option Display Designator PDU is selected on the More Preferences tab on the Dashboard, the PDU is displayed as well.

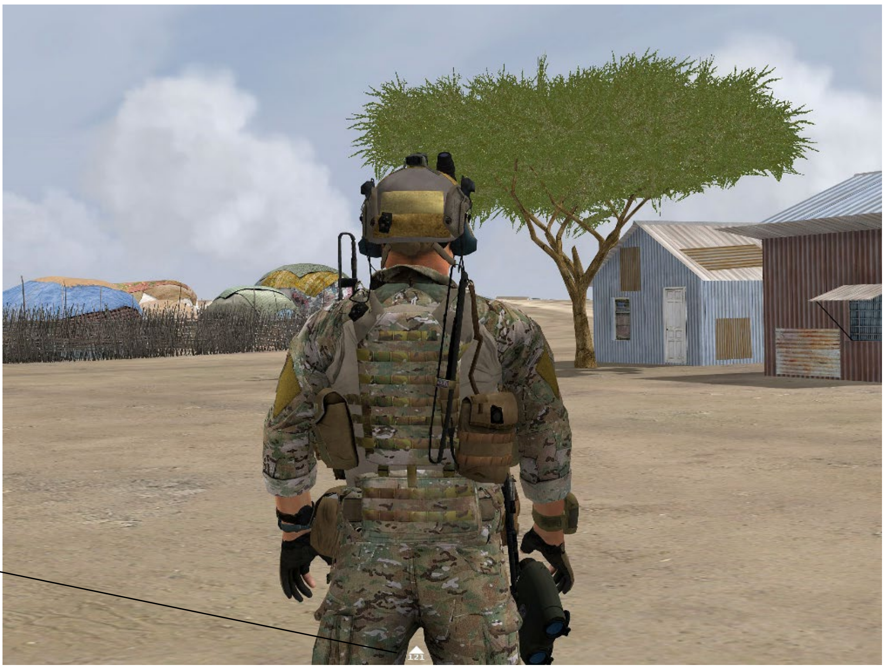
Button Y on the F310 gamepad displays onscreen help for FPS gamepad/joystick functions:



```
FPS Gamepad and Joystick Functions
X:     Fire weapon, or laser designate
A:     Cycle Fire Mode (STOWED, SAFE, SINGLE, SEMI-AUTO)
B:     Select Laser Code
Y:     Display this help
LB:    Mount a vehicle
LT:    Reload clip or Laser Pointer in NVG mode
BACK:  Toggle compass display
START: Recenter head direction

Left Thumbstick:    Move character, or press down to change posture
Right Thumbstick:   Move eyepoint, or press down to zoom

D-Pad:          Third-Person (up)
                First-Person (down)
                Designator/Rangefinder (right)
                Sniper Scope (left)


Loc: S00 21 02 E042 33 13  Hdg: 56 Elev: 3 AGL: 1 Kts: 0
```

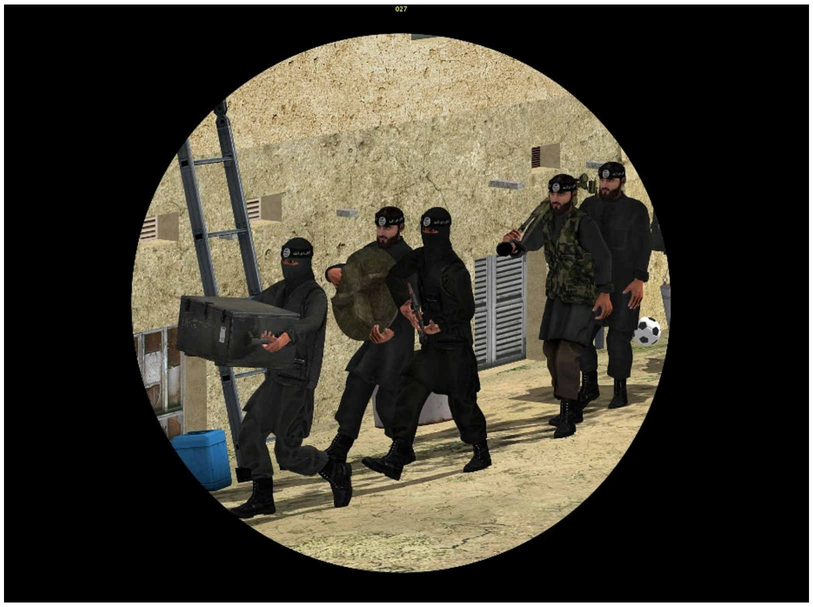
Button LT on the F310 gamepad has two functions. First, it reloads the currently selected gun with another clip of ammunition. Reloads are permitted until the character runs out of clips. The number of clips and the rounds per clip are defined in the file FPS.ini. Secondly, if FPS is running in VRSG's NVG sensor mode, Button LT or 7 displays an IR pointer for roping/designating targets, which you can manipulate with the right thumbstick.

The Back Button on the F310 gamepad toggles a compass heading display centered along the bottom of the display, as shown:



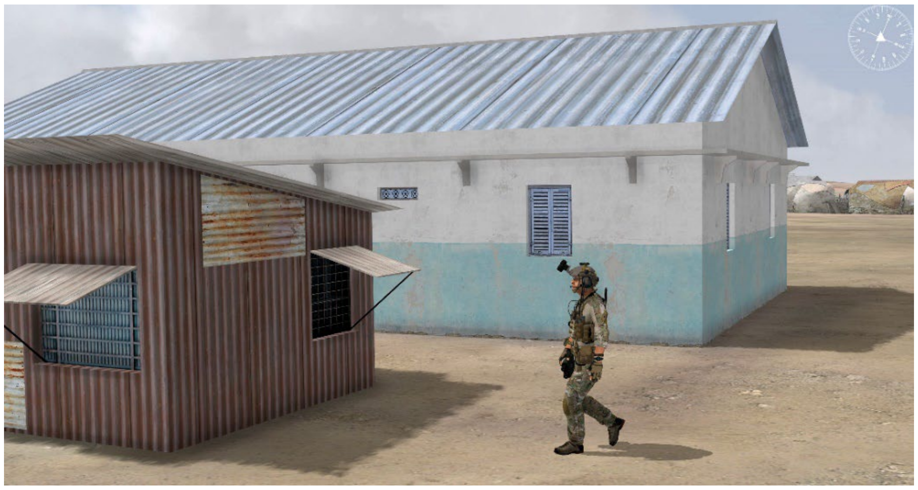
The Start Button on the F310 gamepad recenters the head angle of the FPS character relative to the body.

The following example shows the view in scope mode:



When FPS is used with a head tracker, the second monitor output on an NVIDIA video card can be used to render the scene for a simulated laser rangefinder/designator device. Off-the-shelf devices such as NVIS Ranger 47 Virtual Binoculars can be used to enhance the fidelity of JTAC training. (See the chapter "Using VRSG Trackers, HMDs, and Simulated Military Devices" for more information about the Ranger 47 and other hardware devices with trackers.)

Attach to the FPS character by using the middle mouse button or the attach button on the 6DOF controller. Doing so releases you from the default behind-the-head view and enables you to view the character from any angle.



Detach from the character to resume the default behind-the-head view.

Putting an FPS character on the DIS network

You can easily make your locally simulated character visible on the DIS network. In the Startup Parameters tab of the VRSG Dashboard, click the More Options button to display the Startup Parameters dialog box. In this dialog box, set the network options DIS version, broadcast address, 7-tuple enumeration of the character, and marking.

Enumerations for rounds fired by the FPS character are defined in the FPS.ini file as described earlier in this chapter.

CHAPTER 15

Configuring VRSG for Simulating UAVs

Real-time screen captures from VRSG. The scene shows an MQ-9 Reaper UAS/RPA entity in flight over the high-resolution geospecific modeled port city of Kismayo, Somalia. The inset shows VRSG's simulated UAV camera view.

This chapter describes configuring VRSG for simple UAV / RPA simulations in DIS networked environments. It also describes streaming real-time VRSG H.264 or H.265 feed with UAV telemetry in the KLV metadata of the video stream, to use with visualization on remotely operated devices that can receive full motion video (FMV).

You can configure VRSG to simulate UAVs, ranging from simple configurations to fully integrated applications in these ways:

- Using the Attach Options on the VRSG Dashboard to affix the VRSG view to a UAV entity already simulated on the network. Doing so engages a moderate fidelity camera payload model built into VRSG.
- Using other options on the Dashboard to establish higher fidelity definitions of the camera view functionality.
- Connecting Cloud Cap Piccolo AutoPilot to VRSG via third-party plugin.

This chapter describes these methods, and explains how to create screen captures remotely.

Configuring VRSG for UAV simulation (DIS)

This section describes how you would use the options on the VRSG Dashboard to configure VRSG to mimic a UAV camera view. In this simple configuration, you affix the VRSG view to a UAV entity already simulated on the network, such as attaching to a SAF UAV. In this configuration, the camera payload model is built into VRSG, and the telemetry of the simulated UAV is provided by a DIS entity.

When you start VRSG from the Windows Start menu, the VRSG Dashboard graphical user interface appears. The Dashboard's tabs and dialog boxes are explained in detail in the chapter "Exploring the VRSG System." This section describes only the settings that differ from that chapter or are required for UAV simulation operations. The settings you choose will be based on how you have configured your hardware and where you have loaded your databases.

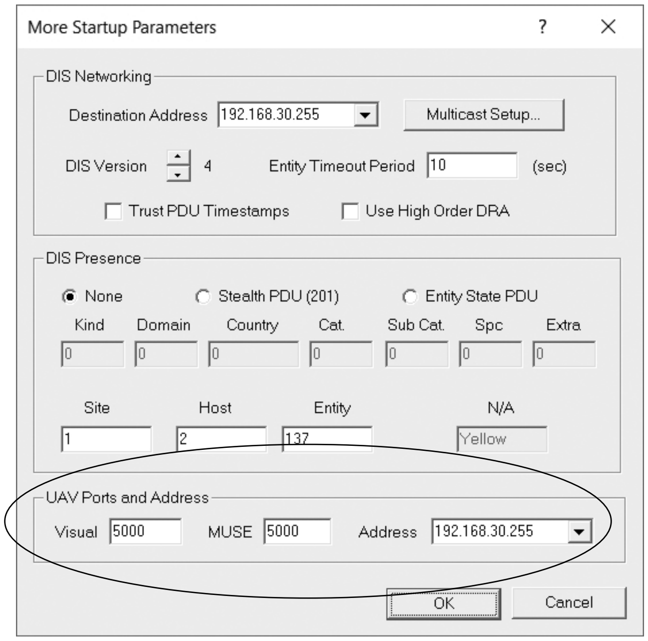
Setting the UAV port and addresses

In some cases the ability to reproduce the UAV view on a remote system is needed. One method to accomplish this is to stream H.264 video to the remote system, as described in a later section in this chapter. If your network does not have sufficient bandwidth to support streaming video, an alternate approach is to instruct VRSG to transmit telemetry to a remote VRSG system, enabling it to reproduce the same scene. The telemetry approach uses a small fraction of the bandwidth of streaming video.

In this type of setup, there is a *master* VRSG computer, which is attached to a DIS UAV entity. This master computer a human operator controlling the sensor with a gamepad or joystick. On the receiving end is a *client* VRSG computer, which is not attached to a DIS entity and does not have a human operator. The client VRSG computer simply replicates the scene of the master, by receiving its telemetry over the network.

To configure both the master and client VRSG in this case:

1. On the Dashboard's Startup Parameter's tab, click the Advanced button to display the Advanced Startup Parameters dialog box.
2. In the UAV Ports and Addresses section, for the master computer, enter the IP address of the client computer. (On the client computer, this field is not important.)
3. The values for Visual port and MUSE port both default to 5000. These values can be changed if needed, as long as the Visual port on the master computer matches the MUSE port on the client computer, and vise-versa. For simplicity, MVRsimulation recommends leaving both values at 5000 for both computers.


More Startup Parameters

DIS Networking
Destination Address: 192.168.30.255
DIS Version: 4
Entity Timeout Period: 10 (sec)
Trust PDU Timestamps
Use High Order DRA

DIS Presence
(● None) Stealth PDU (201) Entity State PDU

Kind	Domain	Country	Cat.	Sub Cat.	Spc	Extra
0	0	0	0	0	0	0

Site	Host	Entity	N/A
1	2	137	Yellow

UAV Ports and Address
Visual: 5000 MUSE: 5000 Address: 192.168.30.255

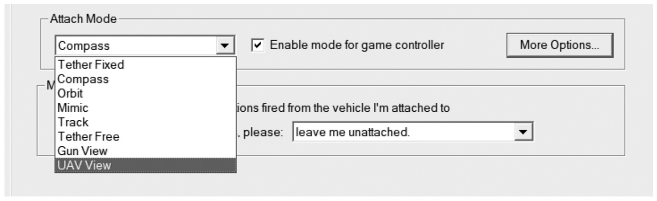
4. Launch VRSG on the master computer and attach to the DIS entity associated with your simulated UAV. On the Dashboard's Attach Options tab, select UAV View for the Attach mode. You can now use your joystick to control the simulated sensor.

Setting entity attachment options

On the Dashboard's Attach Options tab, you specify which entities you want to add or attach to your virtual world display, which mode of attachment you want to use (related to the viewpoint), and which entities to remove or detach from the display.

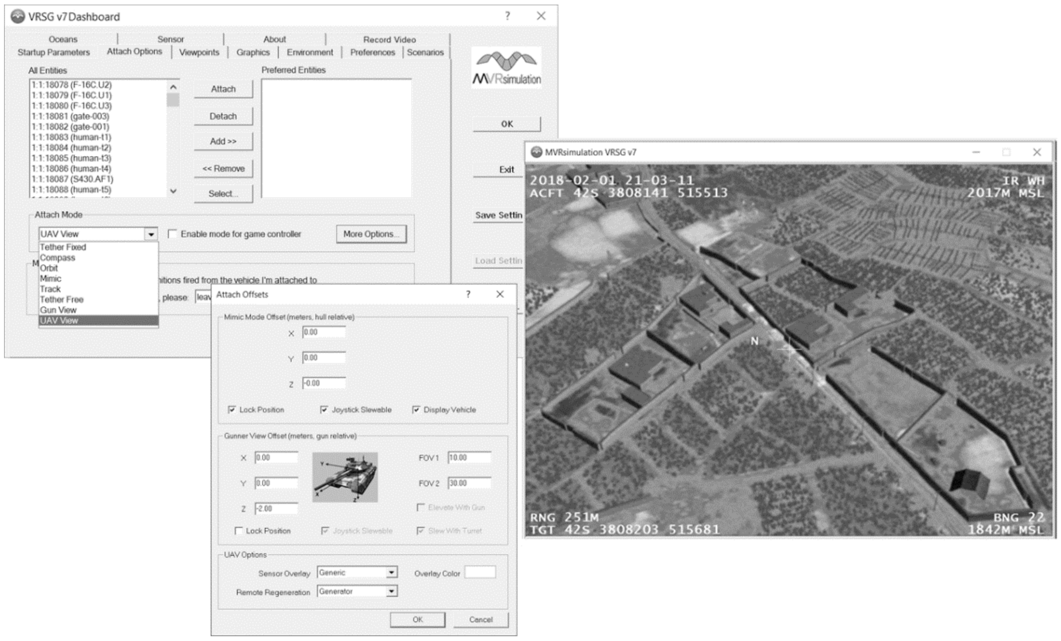
For simple UAV simulations:

1. Select the UAV View attachment mode. Keep the Munitions Attachment Behavior settings at their default values.


Attach Mode
Compass
Enable mode for game controller
...ions fired from the vehicle I'm attached to
..., please: leave me unattached.

   In the case of running both master and client VRSG systems for UAV simulation, select UAV View mode for the master system.

2. Click the Advanced button to display the Attach Offsets dialog box, where you specify a viewpoint that can be associated with a particular entity.



VRSG v7 Dashboard

Oceans | Sensor | About | Record Video
Startup Parameters | Attach Options | Viewpoints | Graphics | Environment | Preferences | Scenarios

All Entities:
1:1:18078 (F-16C.U2)
1:1:18079 (F-16C.U1)
1:1:18080 (F-16C.U3)
1:1:18081 (gate-003)
1:1:18082 (gate-001)
1:1:18083 (human-t1)
1:1:18084 (human-t2)
1:1:18085 (human-t3)
1:1:18086 (human-t4)
1:1:18087 (S430.AF1)
1:1:18088 (human-t5)

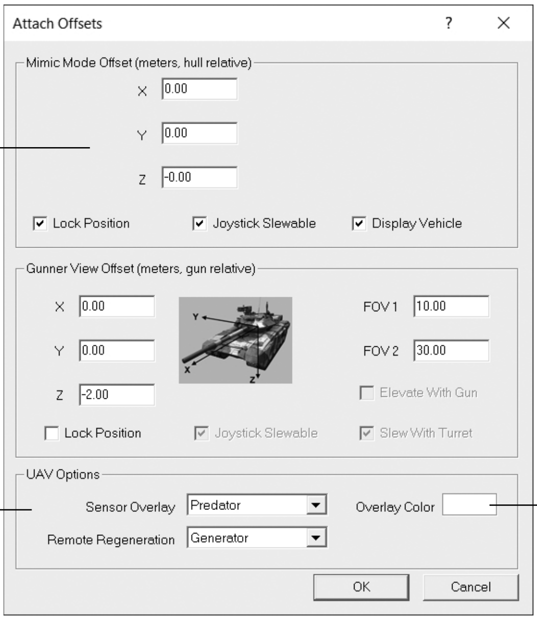
Attach Mode: UAV View
Tether Fixed
Compass
Orbit
Mimic
Track
Tether Free
Gun View
UAV View

MVRsimulation VRSG v7

2018-02-01 21-03-11 IR WH
ACFT 42S 3808141 515513 2017M MSL
RNG 251M BNG 22
TGT 42S 3808203 515681 1842M MSL

Attach Offsets
Mimic Mode Offset (meters, hull relative): X 0.00, Y 0.00, Z -0.00
Gunner View Offset (meters, gun relative): X 0.00, Y 0.00, Z -2.00, FOV 1 10.00, FOV 2 30.00
UAV Options: Sensor Overlay Generic; Remote Regeneration Generator

3. In the UAV Options section of the Attach Offsets dialog box, select the 2D display and color appropriate for your sensor camera view, as shown in the next example. All UAV 2D overlay styles and a Sniper Pod overlay option are available.



Attach Offsets

Mimic Mode Offset (meters, hull relative)
X 0.00
Y 0.00
Z -0.00
[✓] Lock Position [✓] Joystick Slewable [✓] Display Vehicle

Gunner View Offset (meters, gun relative)
X 0.00
Y 0.00
Z -2.00
FOV 1 10.00
FOV 2 30.00
[ ] Elevate With Gun
[ ] Lock Position [✓] Joystick Slewable [✓] Slew With Turret

UAV Options
Sensor Overlay: Predator
Remote Regeneration: Generator
Overlay Color

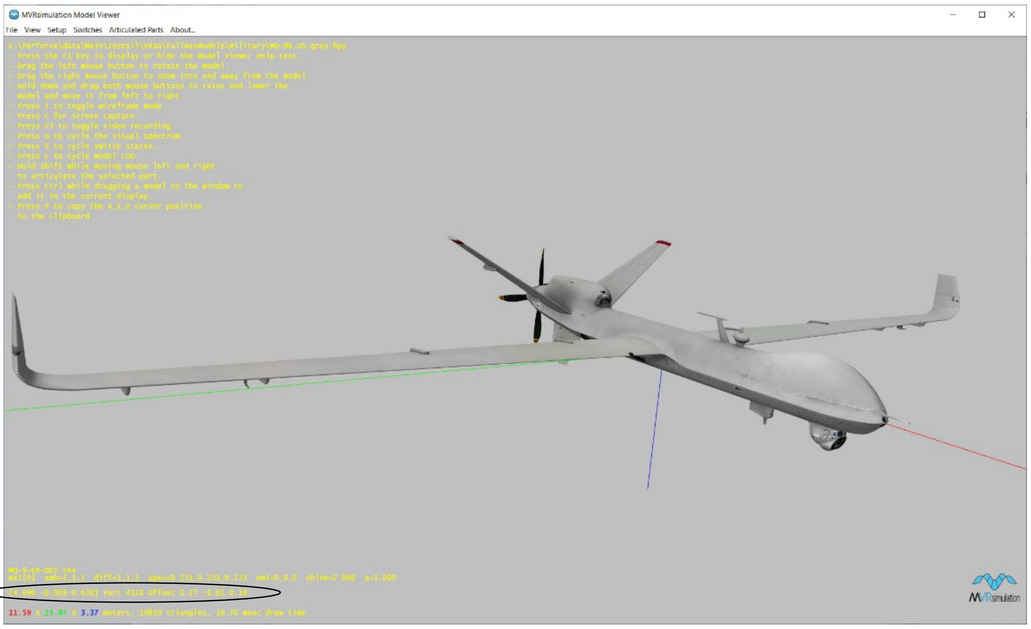
Values to define the location of the sensor on the UAV model.

2D overlay styles for several UAVs, and a Sniper Pod overlay option.

Displays a color palette from which to choose an overlay color.

4. Use the Mimic Offset values to define the location of the sensor in the UAV model's coordinate system. The coordinate system is that used by the DIS Entity model convention: X axis is forward, Y axis is to the right, and the Z axis is down.

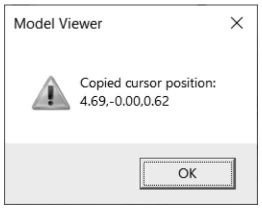
X,Y,Z coordinates of the current cursor postion.


MVRsimulation Model Viewer
File View Setup Switches Articulated Parts About...
MVRsimulation

```
E:\Perforce\data\Main\Install\VRSG\FullResModels\Military\MQ-9B.US.grey.hpy
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording
- Press O to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold Shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
- Press P to copy the x,y,z cursor position
  to the Clipboard.

MQ-9-ER-EXT.tex
mat[0]  amb=1,1,1  diff=1,1,1  spec=0.231,0.231,0.231  emi=0,0,0  shine=2.000  a=1.000
[4.690 -0.009 0.630] Part 4128 offset 0.27 -0.01 0.18
11.59 x 25.97 x 3.37 meters, 19819 triangles, 16.70 msec draw time
```

MVRsimulation's Model Viewer is useful for determining an appropriate offset point for the sensor. Open the UAV model of interest in the Model Viewer by double-clicking the model file (HPY or HPX) in the Windows Explorer. In the Model Viewer, move the cursor to the intended eyepoint offset and note the coordinates shown in the lower-left-corner of the window in the square brackets.

A quick way to capture those cursor position coordinates is to press the "P" key on the keyboard to have Model Viewer copy the coordinates to the Windows Clipboard.



Copied cursor position:
4.69,-0.00,0.62

Entering the correct eyepoint offset ensures that the UAV ownship will be seen from the correct perspective in UAV mode. This will provide representative blockages from aircraft structures such as landing gear, wings, and propellers.

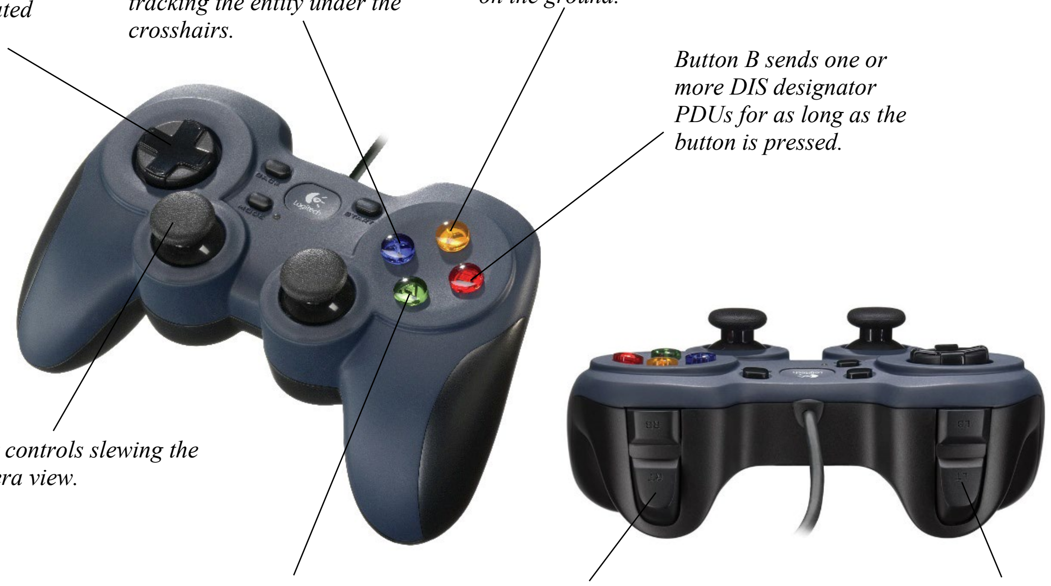
Using a gamepad or joystick to operate the simulated camera view

In the real world, UAV operators use a gamepad or joystick-type device to manipulate the camera view; this is how they see and identify objects on the ground. In VRSG you can simulate manipulating the camera view with a gamepad or joystick. To install and configure a gamepad or joystick to use with VRSG, see the *MVRsimulation Product Installation Guide*.

In simulated UAV camera view mode, VRSG works with any 8-button 4-axis input device that complies with the Human Interface Device (HID) standard. However, for ease of use in operating the UAV camera view, MVRsimulation recommends the Logitech F310 gamepad. This USB device is supported by Windows and VRSG without the need to install any additional software.

Use the UAV View attachment mode to attach to a UAV entity that is on the DIS network or running in a scenario locally on your machine. VRSG obtains the position and orientation of the UAV from DIS, but the camera bearing and depression angle is controlled by the gamepad or joystick attached to the system running VRSG.

The following figure show the mapping of VRSG's simulated UAV camera functions to the Logitech F310 gamepad controls:



The D-PAD controls zooming in on and out from the simulated camera view.

Button X turns on and off tracking the entity under the crosshairs.

Button Y tracks a location on the ground.

Button B sends one or more DIS designator PDUs for as long as the button is pressed.

Left thumbstick controls slewing the simulated camera view.

Button A toggles roll stabilization.

Button RB toggles auto-zooming when tracking an entity or a position on the ground.

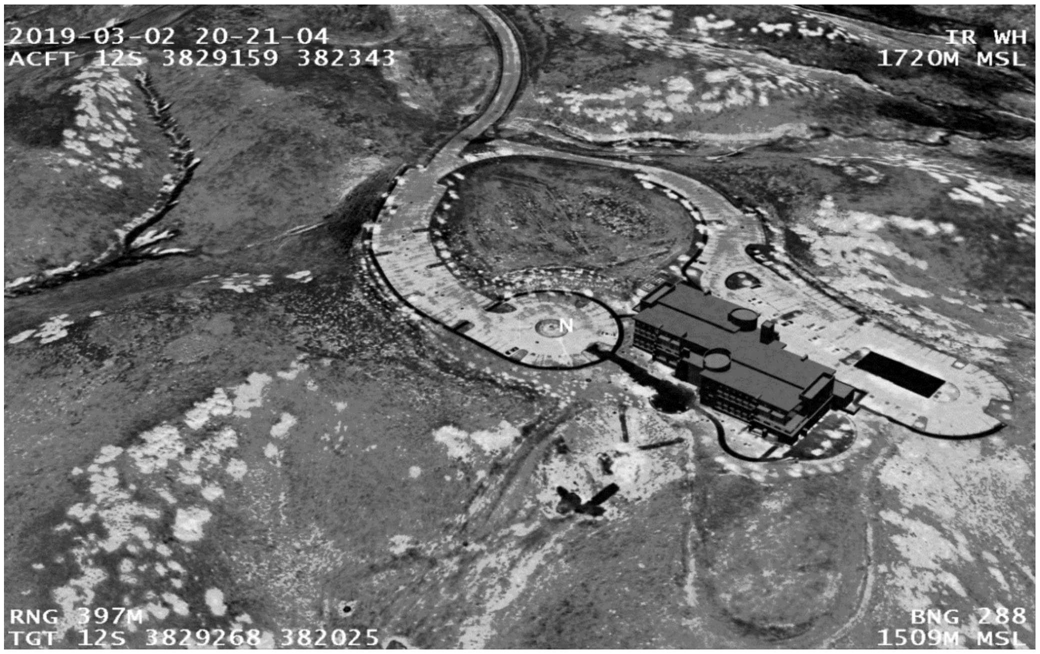
Button LB takes a screen capture.

Use the gamepad controls to:

- Move the left thumbstick left and right to slew the camera view left and right (bearing).
- Move the left thumbstick backward and forward allows you to slew the camera upward or downward (depression angle).
- Press the DPAD forward and back to control zooming in on and out from the simulated camera view.
- Press Button B on the F310 gamepad to send one or more DIS Designator PDUs, which designate what the camera is pointing to. The PDUs are transmitted as long as button is pressed. The contents of a Designator PDU can be customized by an entry in the UAV.ini file, as described in the next section.
- Press Button A on the gamepad to toggle roll stabilization. When engaged, roll stabilization will use the sensor roll angle to compensate for a changing perspective. VRSG will try to maintain a constant ground plane orientation relative to the screen while engaged. You must be tracking an entity or a position on the ground to enable roll stability control with Button 2.
- Press Button X to track on the entity nearest to the center of the field-of-view. The bearing and depression angles will be adjusted automatically to keep the tracked entity in the center of the field of view. Press Button X again to disengage tracking and return bearing and depression to the left thumbstick input control. (The UAV tracking state is now saved as part of a viewpoint. You can restore the tracked entity by restoring a viewpoint.)
- Press Button Y to track a location on the ground. VRSG tracks the point that was in the center of the field-of-view when Button Y was pressed. Press Button Y a second time to disengage tracking and return bearing and depression to left thumbstick input control. (The UAV tracking state is now saved as part of a viewpoint. You can restore the tracked ground position by restoring a viewpoint.)
- Press Button LB to take a screen capture. The screen capture will be saved to an image file, using the settings specified on the Dashboard’s Preferences tab.
- Press Button RB to toggle auto-zooming while you are tracking an entity or a position on the ground. Auto-zoom automatically adjusts the field-of-view so that the tracked object is relatively constant in size. (This is accomplished by modulating the field-of-view as the viewing distance changes.) You must be tracking an entity or a position on the ground to enable auto-zoom with Button RB.

If you use another joystick device for manipulating the simulated camera view, you can configure button mappings for that device.

Installed with VRSG is a file called UAV.ini, which enables you to customize a few UAV settings. The file is located in the VRSG installation directory, \MVRsimulation\VRSG\UAV.ini. In this file, you can configure parameters to customize button mappings for joystick functions. (You can also customize parameters for laser designation and view magnification maximum and minimum values as described in the next section.) Other parameters may be added in future releases of VRSG.



VRSG simulated UAV camera view of geospecific terrain of a desert region.

The terrain was built with 2.5 cm imagery captured by MVRsimulation’s imagery collection UAV. The 2D overlay used in the above screen capture is VRSG’s generic sensor 2D overlay. VRSG is delivered with 2D overlays for many UAVs.

For customizing button mappings for joystick functions, the UAV.ini file contains the following parameters:

```
! default button mappings for joystick device

designate_button:      1
roll_stab_button:      2
track_button:          3
ground_track_button:   4
screen_capture_button: 5
auto_zoom_button:      6
spectrum_button:       9
polarity_button:       10
```

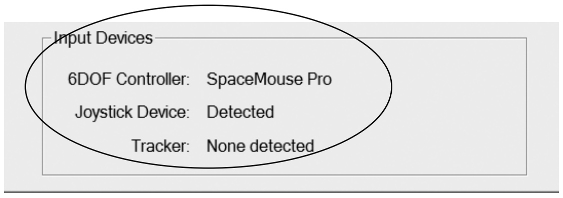
Troubleshooting

If flickering appears in the scene when you are navigating the terrain in the UAV attachment mode, change the Near Clip setting on the Dashboard’s Graphics tab to a higher setting. Having a large near clip plane is beneficial for z-buffer behavior when you are looking at objects at a distance with a narrow field of view, as in UAV camera view simulation.

Uncontrollable drifting in the scene might indicate that the gamepad needs calibration. You can recalibrate the gamepad through the gamepad’s control panel.

If the gamepad or joystick does not respond when you are in UAV attachment mode, make sure that your device is plugged in. Check that the F-lock key on your keyboard is active. Also ensure the Joystick Slewable option is selected on the VRSG Dashboard’s Attach Offsets dialog box.

Upon startup, VRSG indicates on the Startup Parameters tab of the Dashboard whether it has detected the gamepad or other joystick:



Input Devices	
6DOF Controller:	SpaceMouse Pro
Joystick Device:	Detected
Tracker:	None detected

If VRSG has not detected the gamepad or joystick, check the Device Manager in the Control Panel to ensure Windows has detected the device.

Customizing settings for laser designation and view magnification

In the UAV.ini file, located in the VRSG installation directory, \MVRsimulation \VRSG\UAV.ini. In this file, you can configure parameters for laser designation and view magnification maximum and minimum values. (You can also customize button mappings for joystick functions as described earlier.) Other parameters may be added in future releases of VRSG.

For laser designation and view magnification, the UAV.ini file contains the following parameters:

```
laser_power:        40.0
laser_wavelength:   1.06400001049042
laser_codename:     1
laser_code:         123
min_fov:            2.0
max_fov:            45.0
```

For example, you could edit this file to change the min_fov value to a value smaller than 2.0. A horizontal FOV of 2.0 is the default minimum field-of-view (that is, the highest magnification). Changing the value to a smaller FOV would yield a higher degree of magnification.

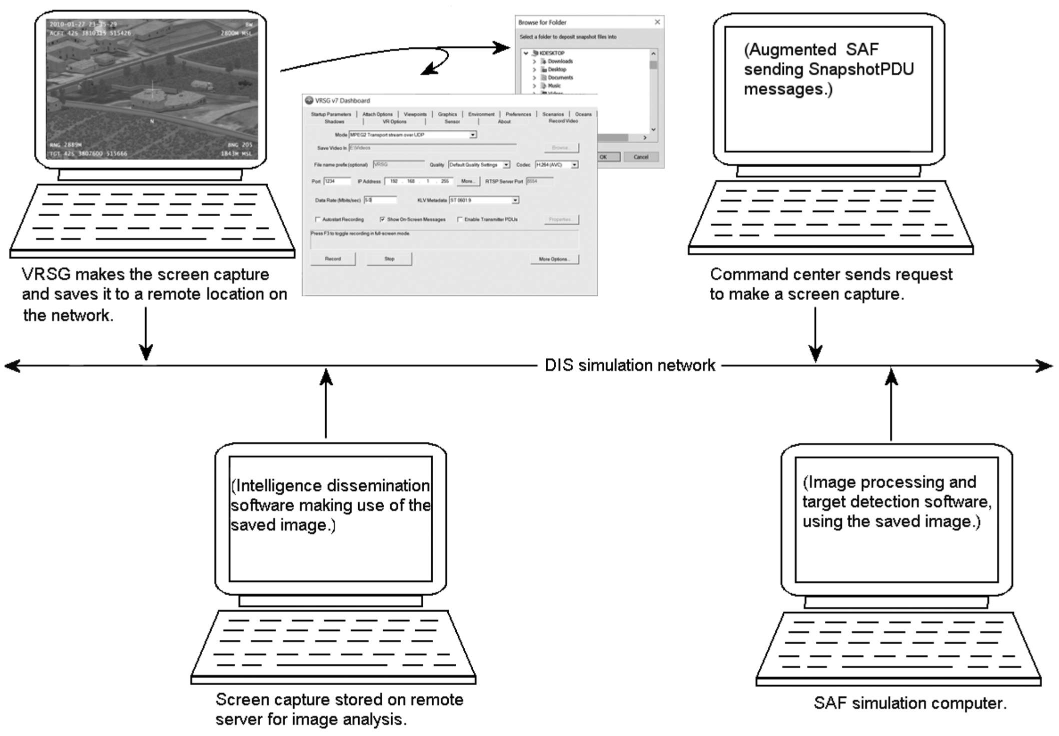
Making remote screen captures using DIS

Many UAV applications send imagery from a vehicle to a remote repository for analysis. Typically, a remote operator will send a message to a UAV and request an image to be sent back to the operator’s location. There are three methods in VRSG to instruct the UAV to take a picture from remote operators on the simulation environment.

In VRSG, you can capture a still image from the virtual world remotely and save it to a file in one of three available image formats. As described in the chapter “Exploring the VRSG System,” you can make such screen captures directly using the VRSG interface.

Similarly, you can send a command from a sensor model that is capable of producing an image, which enables VRSG to capture the current image in its field of view, save the image to a file, and deliver the file for display on another computer. To do so, you write code to send a packet via the UDP interface to initiate VRSG’s virtual world camera and pick up the file on the LAN to deliver the image to an end user on another computer. This process is similar to making screen captures with VRSG's virtual camera directly through VRSG's user interface, but with the ability to remotely command any VRSG channel to produce the image.

VRSG supports SnapshotPDU, which is a DIS PDU that is integrated in certain SAF builds. The diagram below illustrates the remote screen capture process with SnapshotPDU:


(Augmented SAF sending SnapshotPDU messages.)
VRSG makes the screen capture and saves it to a remote location on the network.
Command center sends request to make a screen capture.
DIS simulation network
(Intelligence dissemination software making use of the saved image.)
(Image processing and target detection software, using the saved image.)
Screen capture stored on remote server for image analysis.
SAF simulation computer.

The format for SnapshotPDU is:

```
#define SnapshotPDUKind     194

typedef struct
{
   PDUHeader           header;
   EntityID            id;
   unsigned short      width;
   unsigned short      height;
   unsigned short      _pad0;
   WorldCoordinates    location;
   EntityOrientation   orientation;
   unsigned char       sensor;
   unsigned char       hFov;
   unsigned char       vFov;
   EntityMarking       marking;
} SnapshotPDU;
```

Use the following attributes to configure the SnapshoPDU:

- WorldCoordinates location is a geocentric location.

- EntityOrientation orientation is an orientation of DIS-style Euler angles.
- The sensor field is 0 for visual spectrum, 1 for day-tv, and 2 for FLIR (forward-looking infrared).
- The hFov field is the horizontal field of view, in degrees.
- The vFov field is the vertical field of view, in degrees.
- The marking field contains the name of the intended image file to be produced. The image format is chosen by the extension provided for this filename (that is, .bmp for BMP, .jpg for JPEG, and .ntf for NITF 2.1). The image file will be written to the directory indicated on the Dashboard's Preferences tab.
- The width and height fields define the size of the intended screen capture, in pixels. If these values are zero, VRSG's current framebuffer size will be used.

Screen captures are witten to the directory that is specified in the Preference tab of the VRSG Dashboard, as described later in the section "Setting the capture file format and directory."

Setting the capture file format and directory

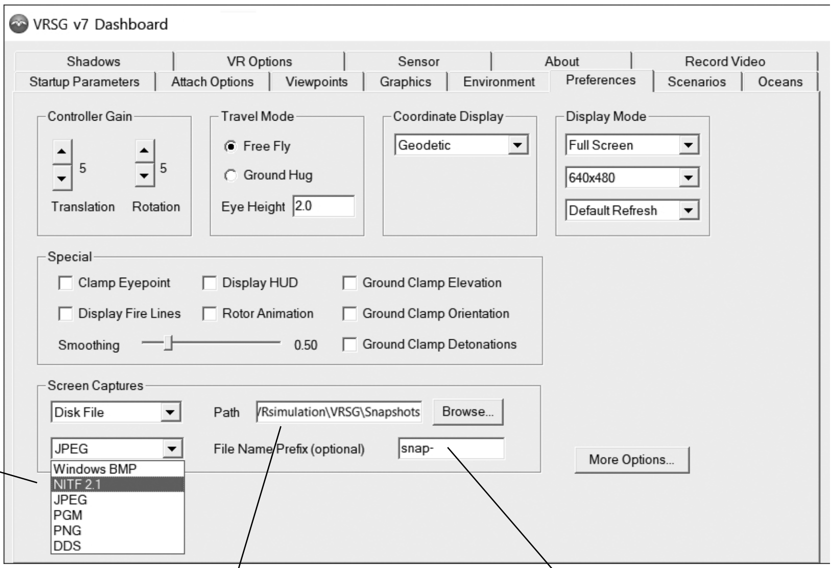
In the VRSG Dashboard interface you set the file format in which you want the screen capture to be saved, an optional filename prefix, and the directory path in which you want screen captures to be saved. These settings are located on the Preferences tab, as shown in the following example.

VRSG can output screen captures in these formats:

- Windows BMP (.bmp)
- JPEG (.jpg)
- NITF 2.1 (.ntf)

BMP and JPEG images are output in full color. NITF images are output in 8-bit per pixel grayscale.

Specify the capture to be saved to a disk file and select the format in which the file should be saved.



Controller Gain: Translation 5, Rotation 5
Travel Mode: Free Fly (selected); Eye Height 2.0
Coordinate Display: Geodetic
Display Mode: Full Screen; 640x480; Default Refresh
Smoothing 0.50
Screen Captures: Disk File; JPEG; Path /Rsimulation\VRSG\Snapshots; File Name Prefix (optional) snap-

Specify the directory in which to save the screen captures. Default is \Snaphots.

Specify a filename prefix for the screen captures. Default is "snap." A value in the EntityMarking field in SnapPDU can override this prefix.

Streaming real-time VRSG video feed to visualize sensor payload imagery

VRSG's HD H.264 video streaming/recording feature includes the option to record UAV Key-Length-Value (KLV) metadata multiplexed into an MPEG2 transport stream. This means that if you are running VRSG in UAV mode, VRSG can provide the UAV telemetry in the KLV metadata of the video stream. VRSG supports a compliant subset of NATO standard STANAG 4609 to include MISB ST 0601.1, 0601.9, or 0601.17 KLV metadata, and MISB security metadata standard 0104.5. VRSG's generated H.264 / .265 video stream with KLV metadata is fully compliant with the latest standards recommended by MISB. The video output's compliance was tested with NGA's Community Motion Imagery Test Tool (CMITT) tool, which validates video and metadata conformance.

To enable the encoding of UAV telemetry, select one of the KLV Metadata options on the Dashboard's H.264 Record tab: ST 0601.1, ST 0601.9, ST 0601.17, or MISB 0104.5. When one of these options is selected during recording, VRSG will generate an MPEG2 transport stream with two embedded streams: one for video and another for the selected KLV metadata.

VRSG outputs MPEG2 transport streams as 1316 byte UDP packets, each consisting of 7 188-byte MPEG2 transport stream packets.

Tactical exploitation systems can use this streaming video feed to visualize sensor payload imagery in real time and extract the UAV metadata. The video stream can be transmitted live over UDP or streamed to a file for later playback.

You can direct the streaming output over UDP to one of these viewers:

- MVRsimulation Video Player – delivered with VRSG and located in \MVRsimulation\Video Player (can decode VRSG's MPEG2 transport stream and embedded UAV metadata).
- GV 3.0 – GOTS viewer available from www.pargovernment.com/gv3.0 (can decode VRSG's MPEG2 transport stream and embedded UAV metadata).
- VLC – shareware available from /www.videolan.org/ (ignores the KLV metadata).

Note that the Windows Media Player is *not* capable of playing the MPEG2 in a transport stream over UDP. Of the three listed viewers, only MVRsimulation's Video Player and GV are capable of decoding and displaying the KLV metadata.

The video encoding and metadata multiplexing run as a VRSG plugin on the same computer.

For more information about VRSG's H.264 streaming and recording capabilities, see the chapter "Exploring the VRSG System."

The H.264 recording plugin supports the latest generation of Nvidia's GPUs which have a hardware H.264 video encoding chip. If VRSG has detected that your video card is insufficient to run the H.264 recording plugin, it will not load the plugin.

KLV data elements

If the KLV MISB 0601.9 standard is selected for H.264 / H.265 streaming, the following keys will be output:

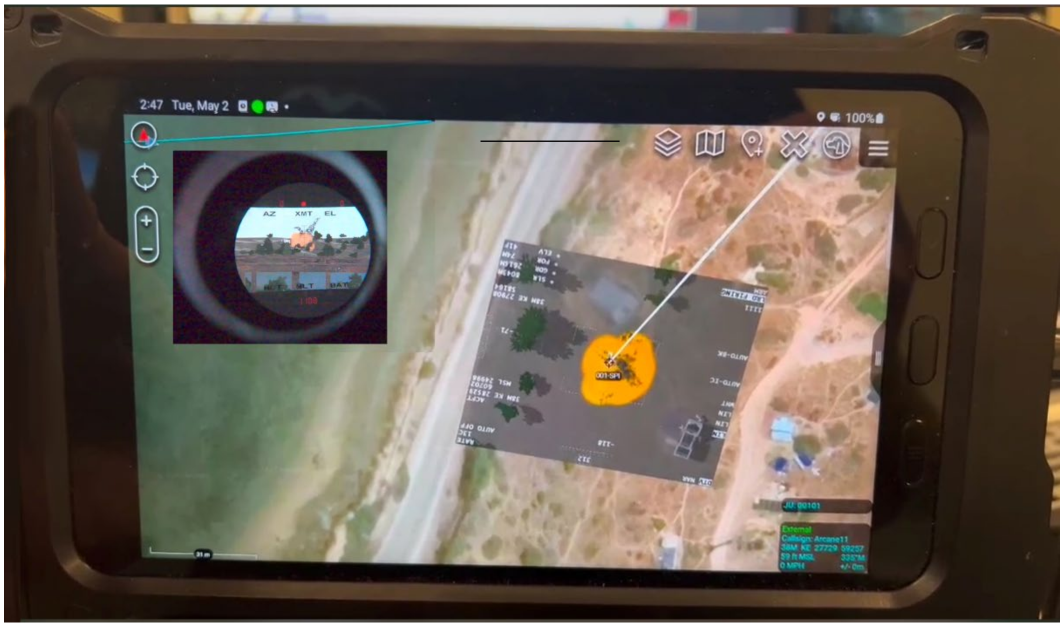
1,2,3,4,5,6,7,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,47,48 ,56,59,65,72

If the older MISB 0601.1 standard is required, these keys are output:
75,82,83,84,85,86,87,88,89,90,91

Using VRSG simulated UAV video with a FMV video receiver

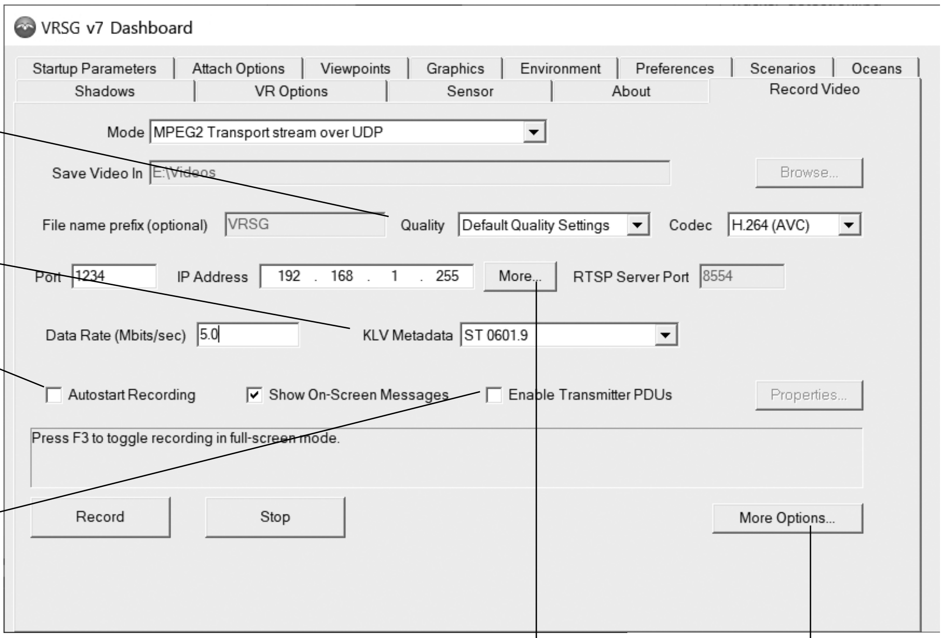
For fielding in joint tactical air controller (JTAC) or forward air controller (FAC) simulation training facilities, VRSG works with any application or device capable of receiving FMV, such as the Android Team Awareness Kit (ATAK) or the ROVER device. When the video receiver is connected to VRSG, VRSG generates the simulated 3D scene and the range and coordinates of a designated target on the device's monitor.

VRSG integrates directly with the Special Warfare Assault Kit (SWAK) to support Digitally Aided Close Air Support (DACAS). VRSG provides the simulated ROVER feed with embedded KLV metadata allowing the ROVER feed to be seamlessly aligned in real-time to the native terrain imagery in SWAK.



Special Warefare Assualt Kit (SWAK) running on a Samsung Active Tab 3 with a VRSG supplied ROVER feed seamlessly overlaid on the embedded terrain. The upper left inset is an image of the corresponding scene as viewed from the emulated SOLFAM.

You can record and stream the simulated UAV camera feed to the video receiver by specifying certain settings on the VRSG Dashboard's Record Video tab.



VRSG v7 Dashboard

Mode: MPEG2 Transport stream over UDP
Save Video In: E:\Videos
File name prefix (optional): VRSG
Quality: Default Quality Settings
Codec: H.264 (AVC)
Port: 1234
IP Address: 192 . 168 . 1 . 255
RTSP Server Port: 8554
Data Rate (Mbits/sec): 5.0
KLV Metadata: ST 0601.9
[ ] Autostart Recording
[✓] Show On-Screen Messages
[ ] Enable Transmitter PDUs

Press F3 to toggle recording in full-screen mode.

Presets handle the tradeoff between video latency and quality.

KLV metadata options.

Starts recording immediately upon VRSG startup.

Select to enable the transmission of DIS Transmitter PDUs which advertise the system as a video source.

Option to add multiple recipient IP

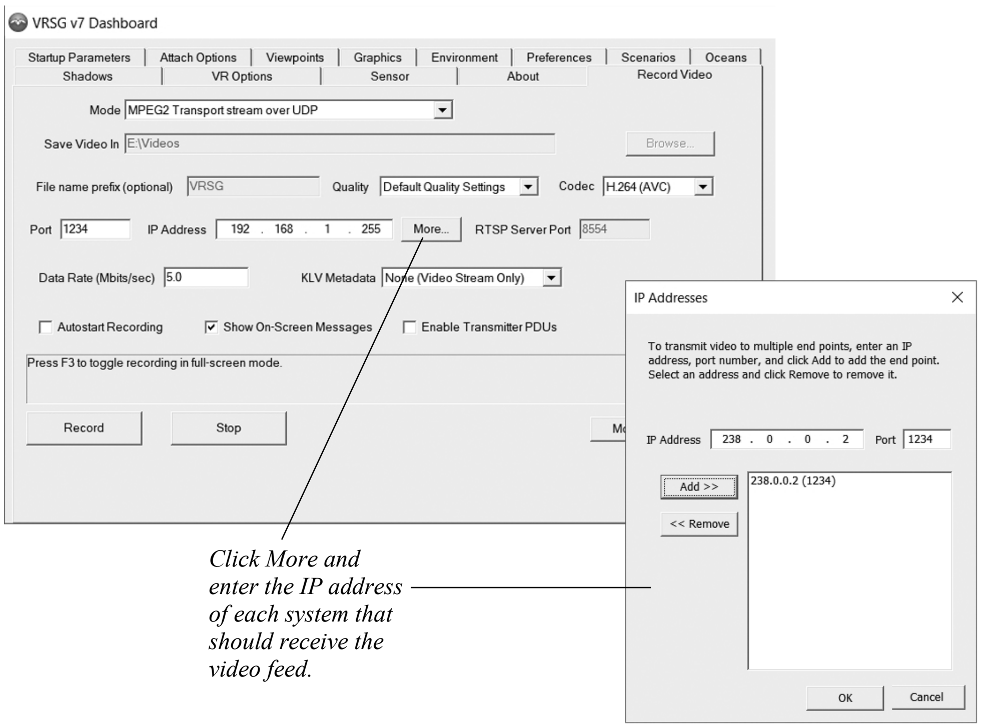
Displays more options for recording simulated sensor view output with KLV metadata.

To record and stream simulated UAV camera feed to a FMV receiver:

1. On the Record Video tab, set the output to Transmit Over Network.
2. Set the IP address to the IP address of the video receiver.
3. Set the port to the port the video receiver is configured to listen to.
4. Click Record to begin transmitting the video.

See the video receiver documentation for information about configuring the device to receive streaming MPEG video.

The local VRSG can be configured to generate H.264 streaming digital video, which would be constrained to the local area network for the purposes of stimulating tactical video receiver devices.



Mode: MPEG2 Transport stream over UDP

Save Video In: E:\Videos

File name prefix (optional): VRSG   Quality: Default Quality Settings   Codec: H.264 (AVC)

Port: 1234   IP Address: 192 . 168 . 1 . 255   RTSP Server Port: 8554

Data Rate (Mbits/sec): 5.0   KLV Metadata: None (Video Stream Only)

☐ Autostart Recording   ☑ Show On-Screen Messages   ☐ Enable Transmitter PDUs

Press F3 to toggle recording in full-screen mode.

To transmit video to multiple end points, enter an IP address, port number, and click Add to add the end point. Select an address and click Remove to remove it.

IP Address: 238 . 0 . 0 . 2   Port: 1234

238.0.0.2 (1234)

Click More and enter the IP address of each system that should receive the video feed.

The video reproduced by the local VRSG regeneration station can be configured to transmit digital video to one or more local video players (such as a device running ATAK, MVRsimulation’s video player, GV, or VLC). Set this up on the Record Video tab as follows:

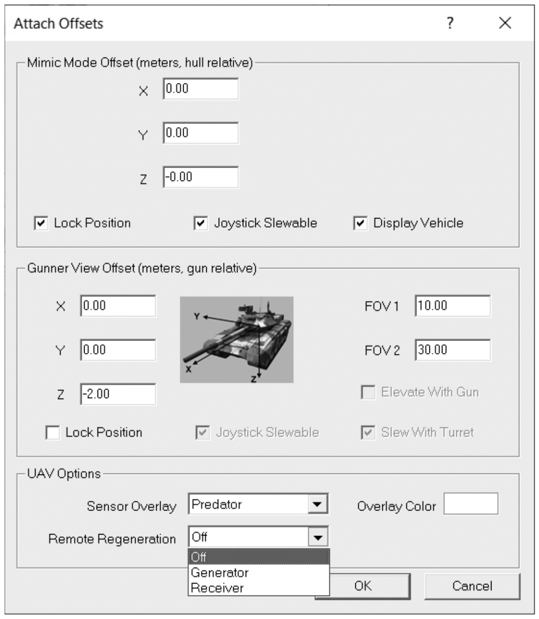
1. On the Record Video tab, set the output to MPEG2 Transport Stream Over UDP.
2. Set the IP address and port of the digital video player. To enter multiple addresses, click the More button next to the IP Addresses field. When the IP addresses dialog box appears, enter the IP address and port of each system that should receive the feed.
3. Click Record to begin transmitting the H.264/.265 video.

See your video receiver documentation for information about configuring the device to receive streaming MPEG video on the local network.

Configuring VRSG as a UAV regeneration station

Some long-haul networked environments do not have the bandwidth to handle the streaming of digital video. In such cases, a site can set up a *UAV regeneration station* which is a local VRSG computer configured to reproduce the UAV video of a remote VRSG computer. As described above, a given VRSG in UAV mode can transmit telemetry packets via DIS packets to a remote VRSG client, where in turn the remote VRSG reproduces the same picture. The bandwidth required to send the telemetry is a small fraction of the bandwidth required for digital video, and is suitable for the modest bandwidth of long-haul networks.

1. On the Attach Offsets dialog box, set the UAV Options on the originating VRSG machine that is generating the video. Be sure to choose the Generator option for Remote Regeneration.


Attach Offsets

Mimic Mode Offset (meters, hull relative)
X 0.00
Y 0.00
Z -0.00
[✓] Lock Position [✓] Joystick Slewable [✓] Display Vehicle

Gunner View Offset (meters, gun relative)
X 0.00
Y 0.00
Z -2.00
FOV 1 10.00
FOV 2 30.00
[ ] Elevate With Gun
[ ] Lock Position [✓] Joystick Slewable [✓] Slew With Turret

UAV Options
Sensor Overlay Predator
Overlay Color
Remote Regeneration Off

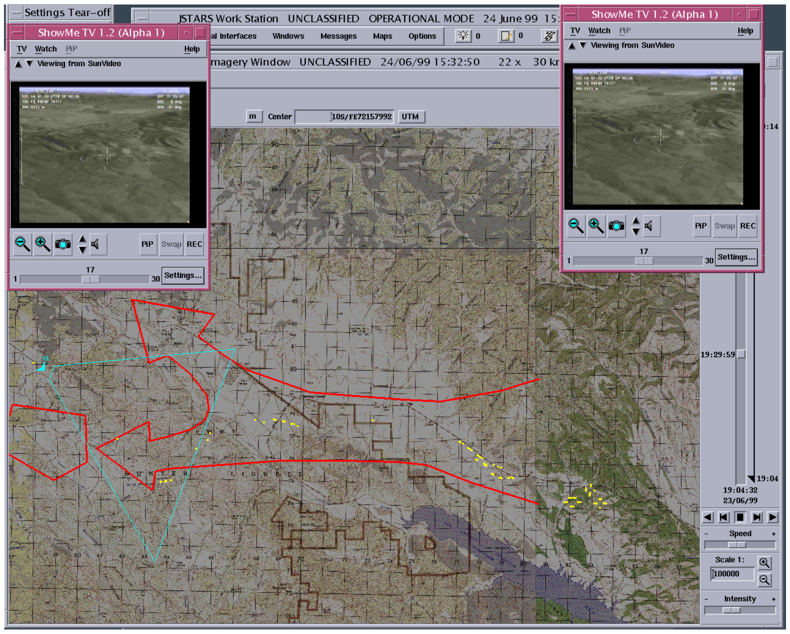
2. As described in the steps above, set the IP address and port of each recipient VRSG client machine.
3. For each recipient VRSG client machine, set on the Attach Offsets dialog box the Receiver option for Remote Generation.

See your video receiver documentation for information about configuring the device to receive the DIS packets from a UAV regeneration station.

Using VRSG UAV video to classify moving targets

You can use VRSG as a simulated, live, virtual video feed from a UAV that is used to classify ground information from a Geographical Situational Display. Airborne or space-borne collection systems that use Ground Moving Target Indication and target identification (GMTI) devices create symbolic representations of moving entities over large geographic areas. VRSG's UAV mode allows it to be attached to any DIS entity, potentially produced by a SAF system.

The following example of a JSTARS workstation (JSWS) using simulated UAV video feeds from two separate VRSG simulations. The blue triangle on the upper-left part of the display shows the field of view of one of the simulated UAVs.



As you view GMTI tracks such as RADAR and hits reports being developed in a geographical situational display simulation, you can also view UAV video data classifying those tracks. The video data must be coincident in location and time to the GMTI tracker. For instance, if the tracker reports a rise in elevation, you see the entity in the video driving up a hill. If you report telemetry, then the entity should be within the field of view (FOV), which should be in the general area of the track in the geographical situational display.

The following is one possible scenario you could use to provide the UAV-simulated video from VRSG to classify moving ground targets:

- In a SAF application such as OneSAF, JointSAF, XCITE, or a commercial system such as BSI's MACE, place entities into the virtual battlespace, task them, and have them move around. The entity position and orientation would be communicated to the MTI and VRSG via DIS.
- Your moving target indicator (MTI) simulation would create a dot on a 2D display for each entity, which is moving (that is, non-zero velocity vector).
- From the UAV payload view, you could slew the camera to view the entities. The coordinates reported by the UAV overlays should correlate with what the MTI is showing for entity positions.
- You could use a SAF application to create a UAV and have it fly a mission plan. Then make the VRSG view coincident to that UAV by selecting the UAV attachment mode on the Dashboard's Attach Options tab. When VRSG is set to the UAV mode, it generates telemetry data in a format compatible with the Joint-STARS (JSTARS) GMTI geographical plan view display to show the UAV position, camera angle, and field-of-view.

Using Cloud Cap Piccolo AutoPilot with VRSG

Georgia Tech Research Institute (GTRI) Aerospace, Transportation and Advanced Systems Laboratory (ATAS) has developed a VRSG plugin to connect a FlightGear open-source flight simulator application to VRSG. The GTRI ATAS lab uses the plugin specifically to connect Cloud Cap Technology Piccolo AutoPilot to VRSG for UAV simulation. Piccolo outputs data for simulation in FlightGear data format.

This plugin allows entities represented by Flight Gear's v0.9.10 data format, to be displayed in VRSG by internally receiving the Flight Gear packets over UDP and translating them to DIS entity state packets which VRSG can natively process. In addition to Piccolo, the plugin will work with any flight simulation application that outputs FlightGear data.

You can download the plugin, named Piccolo.dll, from MVRsimulation's Download Server from the /Software/Plugins/Piccolo/ directory. In that directory is a PDF document with instructions on how to use the plugin.

This plugin has been provided by Georgia Tech Research Institute; it is not officially supported by MVRsimulation. To obtain more information about this plugin, or to report any issues with it, contact Robert Bever at Georgia Tech Research Institute at robert.bever@gtri.gatech.edu.

CHAPTER 16

VRSG Radar Simulation

MVRsimulation's VRSG Radar™ option enables you to build the simulation of a radar system on top of the VRSG rendering engine, using round-earth VRSG terrain tiles, culture, and moving models. VRSG Radar exploits the capabilities of modern commercial graphics accelerators to solve real-time radar simulation problems that historically have required extensive custom hardware and software, at great expense. VRSG Radar is suitable for building radar simulations such as the F16 Digital Radar Land Mapping System (DRLMS), or even synthetic aperture radar (SAR) displays for Unmanned Aerial Vehicles (UAV) or other such equipped platforms.

VRSG Radar real-time synthetic aperture radar (SAR) display of 3D terrain of an Afghanistan village.

A real radar system sends out a pulse of electromagnetic energy and measures the intensities of the energy that is reflected off of objects in the environment and returned to the system. The amount of time it takes for energy to return to the radar antenna determines its range from the system. The intensity of the returning energy at a given instant in time indicates the reflectivity of the material it contacted, and the degree to which the material is facing the emitting system. The system typically sends out pulses as it sweeps from side to side in azimuth, resulting in a two-dimensional collection of data.

VRSG Radar assists in constructing a radar simulation by modeling the processes of the real radar system. Developers can create a radar host system that collects the raw radar return data from VRSG Radar and presents the data in a format suitable for the system being modeled.

VRSG Radar handles the complexity of modeling the beam as it travels through space, impacts objects, and simulates the amount of energy that would be returned to the radar antenna.

Requirements

The hardware requirements for running VRSG Radar are similar to that of VRSG itself, as described in the *MVRsimulation Product Installation Guide*.

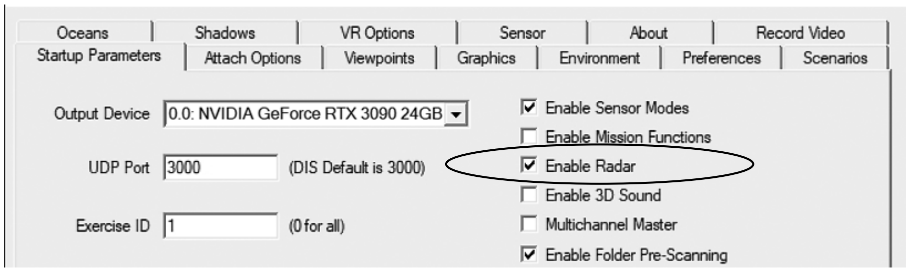
VRSG Radar can run any set of terrain tiles that you can run in VRSG. Although you can run the Radar Host on the same computer as VRSG Radar, for best performance you should run it on a separate computer. The Radar host computer does not need to be a high-end system; it requires only basic 3D capabilities.

VRSG Radar uses the same visual database as the out-the-window channels, providing perfect correlation with the visual display. You might choose to modify certain textures to augment their reflectivity in the radar spectrum, but the databases will remain perfectly geometrically correlated.

Starting VRSG with Radar enabled

The Radar component is installed automatically with VRSG. However to operate the VRSG Radar, international (non-U.S.) customers must first obtain and enter an MVRsimulation-supplied unlock code to enable the option for your VRSG license. VRSG Radar contains some enhanced features not available in the core product. For information about obtaining unlock codes for this component from MVRsimulation, see the *MVRsimulation Product Installation Guide.*

If VRSG Radar has been enabled for your license, an Enable Radar checkbox will appear on VRSG's Startup Parameters tab as shown below:



Output Device: 0.0: NVIDIA GeForce RTX 3090 24GB
UDP Port: 3000 (DIS Default is 3000)
Exercise ID: 1 (0 for all)
[✓] Enable Sensor Modes
[ ] Enable Mission Functions
[✓] Enable Radar
[ ] Enable 3D Sound
[ ] Multichannel Master
[✓] Enable Folder Pre-Scanning

Select the Enable Radar checkbox to activate the Radar feature for a VRSG session.

If you do not intend to use the Radar feature for a VRSG session, you should leave this box unselected to save the creation of resources needed to support Radar.

When the Enable Radar checkbox is selected, VRSG becomes available to a Radar Host to service Radar requests. VRSG will operate as normal in the visualization window, as either a standalone viewer or as a dedicated channel to a simulation host. The Radar requests will be serviced as a background task.

Using the VRSG Radar ICD, the Radar Host can optionally have VRSG's visualization window display a debug view down the radar beam's line-of-sight. This view can help in determining whether the beam is being properly positioned and oriented. Without the debug window enabled, you can use VRSG's visualization window as an out-the-window, EO, or IR scene as normal.

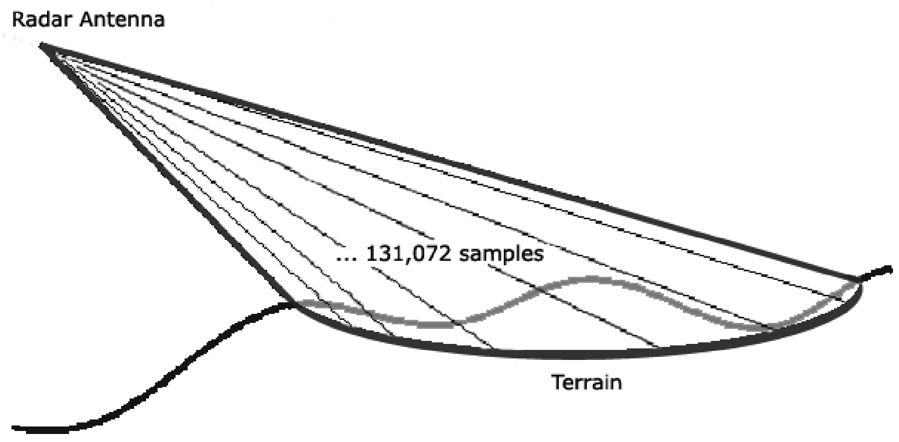
The displayed VRSG Radar image is the view that is looking down the line-of-sight of the beam with a horizontal and vertical field-of-view angle that matches the beam profile and divergence. To digitize the radar cross section (RCS) of the beam, VRSG produces two raster images, one of range and one of intensity. This pair of range and intensity rasters are processed into the *range bins,* which are returned to your application, as described in the next section. These rasters can be very high resolution (e.g. 4k x 4k), resulting in over 16 million ray casts per pulse. The rendering of these rasters and their processing into range bins is performed on the GPU, allowing for high frame rates.

How VRSG Radar works

VRSG Radar works in conjunction with a user-developed Radar Host. The Radar Host models the position and orientation of the radar system, as well as the orientation of the beam relative to the platform hosting the radar system. As the Radar Host sweeps the beam in azimuth, it sends a request to VRSG Radar for the returned raw radar data corresponding for the current radar position, orientation, beam azimuth, beam dimensions, and maximum range. VRSG Radar responds with the set of summed intensities across the quantized range space. The Radar Host in turn processes this raw radar return data for display purposes.

The Radar Host must quantize the beam sweep in azimuth into discrete "looks" it sends to VRSG Radar for processing. Thus the fidelity in azimuth is related to the beam sweep rate, and the speed in which VRSG Radar and the radar host can process each look. VRSG Radar can typically support frame rates up to 300 Hz using modern PC hardware, so a beam sweep rate of 6 degrees per second can yield an azimuth resolution of as fine as 0.02 degrees per look.

For each radar look requested by the radar host, VRSG Radar quantizes the beam's field-of-regard (FOR) into 131,072 samples of range and intensity using the default radar image size of 64 x 2048. VRSG Radar casts 131,072 rays out into the scene distributed across the beam's FOR and intersects these rays with database and moving model geometry, as shown in the following diagram:



At the first point along the ray where terrain database or model geometry is intersected, VRSG Radar calculates the amount of energy that would be returned; it does so by considering the incident angle of the beam with respect to the local surface normal, in conjunction with the reflectivity of the intersected material. All these things are taken into consideration in simulating the amount of energy that would be returned to the radar antenna.

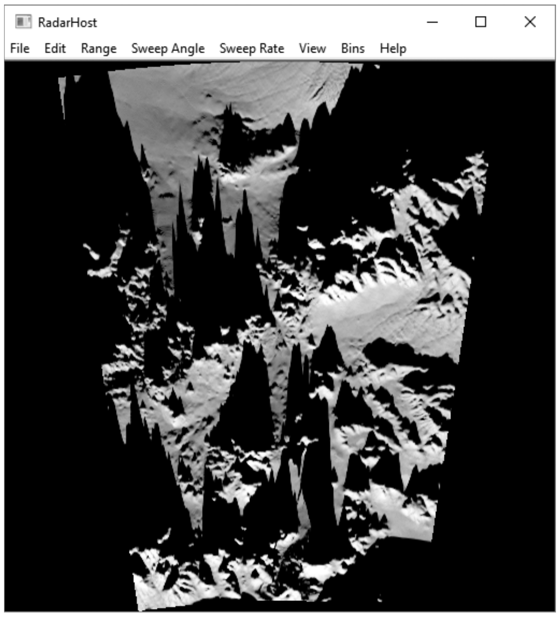
The range interval requested by the radar host is quantized into a set of *range bins* (the set of corresponding range and intensity pairs). The Radar Host requests the number of range bins using the Radar ICD. The number of range bins can vary from 256 to 4096 depending on the application. The returned intensity for a range bin is the sum of all 131,072 samples that fall in that range interval. It is this array of summed intensities that the radar host processes into a displayable form.

Because this approach mimics how a real radar system works, subtle radar characteristics such as shadows and bright spots on front-facing objects are a natural byproduct of the implementation.

Quantizing 131,072 samples 300 times per second is over 40 million rays cast per second. Although this might seem to be an unachievable amount of processing, VRSG harnesses the power of DirectX 11 programmable vertex and pixel shaders to make this possible on a low-cost PC platform. The entire radar equation runs in a pixel shader that interpolates surface normals and material reflectivity at the pixel level.

Material reflectivity

An important factor in how much energy is returned to the radar antenna is the reflectivity of the material struck by the radar beam. As most fixed-wing simulator visual databases are on the order of tens or even hundreds of geocells in size, it is not practical to collect geospecific material attributes for such a large area for the purposes of physics-based reflectivity modeling. Instead, users can exploit the fact that objects that are highly reflective in the radar spectrum are typically also bright in the visual spectrum. Thus using geospecific imagery to derive the material reflectivity is a reasonable and cost-effective approach for training systems. VRSG Radar uses the intensity of the texture map at the point of intersection to determine how much radar energy would be returned to the antenna. The local material reflectivity is then modulated by the incidence angle of the incoming beam with respect to the local surface normal.



For a large geospecific database, the user may be satisfied with the results obtained from using the visual database as-is in the radar simulation. However, as time and budgets permit, the user can modify the visual database textures to make material reflectivity more consistent with the actual radar system being modeled. For example, an asphalt airstrip would likely have a relatively low radar return. The visual database, however, might use a light gray texture map on the airstrip that resulted in inadequate contrast in the radar simulation between the airstrip and the surrounding terrain. This condition could be corrected easily by darkening the airstrip texture for the radar simulation's copy of the database, leaving the visual channels copy of the database unchanged. Such point-wise iterative enhancements could continue until the radar database produced satisfactory results.

All 3D models are modeled with 3D geometry consisting of a set of triangles that has a normal vector associated with each vertex. These vertex normals are used to compute the amount of energy that will be returned, by comparing the angle of the normal with the incoming beam direction. Each face also has a reflectivity value, which by default is its visual spectrum texture. The reflectivity will also modulate the returned energy. Rounded objects will have smoothly interpolated normals, creating greater opportunities for alignment with the beam vector, which in turn generates higher returns. More faceted shapes (such as stealth aircraft), will have less alignment and therefore generate less returns.

Radar reflectivity can be modulated on a per-model basis using the "-radarReturnScale" command in ModelMap.ini or a cultural feature (.clt) file. See the chapter "Configuring Model and Events" for more information. This command is useful to globally enhance or reduce the return intensity for a given model.

VRSG Radar ICD

VRSG Radar and the Radar Host communicate using a simple UDP-based protocol. VRSG Radar reads requests on port 6011 and returns results back to the Radar Host on port 6012.

The details of the message structures are in MVRsimulation's VrsgRadarICD header file, which you can download from the MVRsimulation Download Server's /Software/Interfaces directory. Customers on active maintenance who need an account on MVRsimulation's Download Server can request an account by emailing a request to downloads@mvrsimulation.com.

The simplest Radar host needs to only implement two messages:

- *VrsgRadarFrameRequest* – the host sends this message to VRSG to request a Radar look.
- *VrsgRadarFrameResponse* – VRSG responds with the range bins in one or more of these messages.

Typically, the Radar Host would turn the range bins into a texture map used to texture lines or pie-shaped primitives to update an incremental portion of the real-time Radar display. The Radar Host can also perform various signal processing functions on the returned data.

The current version of VRSG Radar returns up to 4096 range bins for every frame request, depending on how many range bins the host has requested. If the host requests more than 512 range bins, then the host will receive multiple VrsgRadarFrameResponses for every VrsgRadarFrameRequest. The response message contains the frameId, the current message sequence (msgNum), and the total number of messages (msgMax) being returned for the frame. These fields enable the host to synchronize with VRSG Radar.

In addition, the ICD includes these two messages which the host can use to customize the radar behavior:

VrsgRadarPerformanceSettings – customize performance settings such as texture and geometry level-of-detail scales, and configure maximum range for terrain pre-paging.

VrsgRadarSetBufferSize – allows the host to override the default size of the radar image. The default size is 64 x 2048. You can customize the size to specify an aspect ratio more appropriate for your beam profile. A bigger buffer will also enable you to increase resolution, as more samples will be generated to populate the range bins. Larger buffers take longer for VRSG to process, so there is a fidelity/performance trade-off.

Example Radar host application

Upon request, MVRsimulation will provide an example Radar Host application with full source code.

The example application illustrates how to communicate with VRSG Radar and prepare returned Radar data for display.

You can use this example application as a starting point for a new application, or as an example for how to embed the functionality into an existing application.

DRLMS example

The DRLMS exemplar radar host illustrates the simulation of a radar system similar to that used by the F16 digital radar land mapping system.

The full source code provided with the example application uses Microsoft Foundation Classes for the graphical user interface and DirectX 11 for the Radar display. Although Direct3D is used for the graphical display in the example application, the VRSG Radar architecture is graphics-API independent. You can construct a Radar host on any platform using any graphics API.

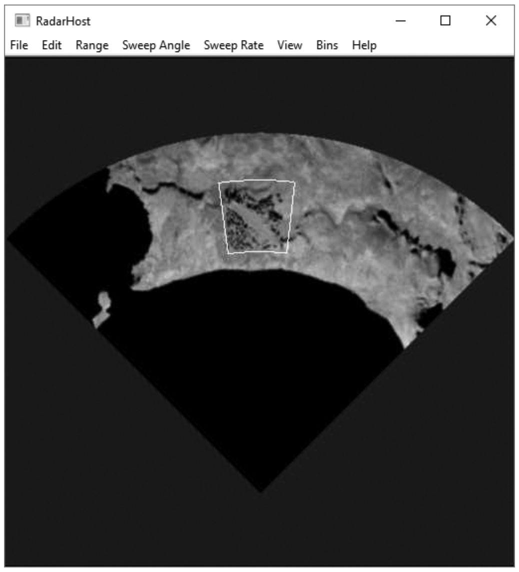
For Radar Host developers, the example application illustrates how to:

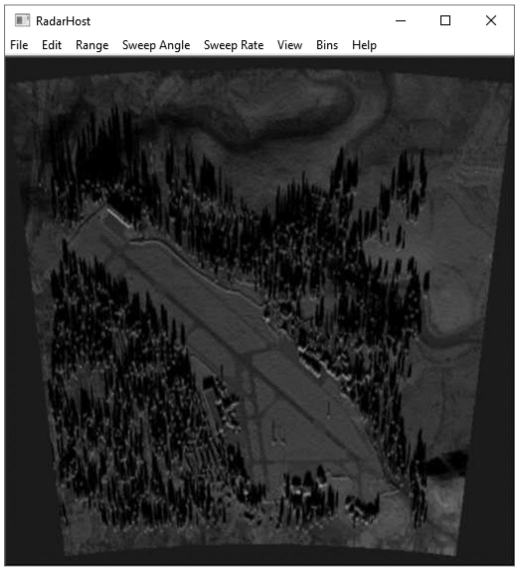
- Set up a communications socket with the Radar host.
- Ortho-rectify the radar return data and display it in a Cartesian window space.
- Display overlays and cursors on top the rendered radar imagery.

The example application features graphical interface controls to specify:

- Radar position and orientation.
- Radar beam geometry and maximum range.
- Sweep rate and sweep angle.
- Zoom mode for higher detail imaging.

The following two images illustrate MVRsimulation's virtual Burlington, Vermont, (BTV) airfield imaged at a 20 nautical mile range, and the corresponding zoom mode.





To activate the zoom mode, click the left mouse button in the lower-left corner of the intended zoom area. Drag the mouse to the intended upper-right corner. The zoom region is outlined by a yellow line as shown (in white) above. Choose View > Zoom Mode to render the zoom area in higher detail.

The zoom mode is rendered with a slower scan rate to increase azimuth resolution, and the range bins are collapsed onto the zoom region for greater range resolution. By default, the Radar host initializes itself at the location N35 W117. You can change the beam origin and other beam characteristics under the File > Setup By Aircraft Location menu. Be sure to have terrain tiles loaded in VRSG to cover the area you are imaging.

CHAPTER 17

Using VRSG With VR Systems, Trackers, and Simulated Military Devices

You can use a 3- or 6DOF tracking system with VRSG to control the position and/or orientation of the eyepoint in the rendered scene. VRSG is integrated with the near-eye display mechanism and tracker so as to minimize latency during rapid movements.

Tracker use with VRSG can range from virtual reality (VR) or mixed reality (MR) head-mounted displays (HMDs) to handheld emulated military devices for target locating and laser designating used by JTACs.

At MVRsimulation's DJFT Observer station looking through the Varjo XR-3 headset at the VRSG scene displayed in the emulated SOFLAM. In this mixed-reality environment, one can interact with both the equipment and the virtual world.

This chapter describes VRSG's support for several commercial VR/MR and other tracking systems, and for simulated military equipment (SME) that utilizes a tracker.

At the time of publication, the latest VR devices VRSG supports are:

- Varjo XR-1, XR-3 and XR-4 mixed-reality systems.
- Vajro VR-2 and Varjo VR-3 virtual reality systems.
- HTC VIVE Pro virtual reality system and the HTC VIVE Tracker.

- Valve Index.
- HP Reverb Pro virtual reality headset.
- Windows Mixed Reality VR headsets.

Other supported commercial tracking devices include:

- Ascension Technology's Flock of Birds tracker.
- Polhemus Scout head tracker.
- Inertial Labs tracking system.
- InterSense tracking products.
- NaturalPoint's TrackIR 5 head-tracking input device.
- SA Photonics SA-62 HMD system and SA Photonics stereo SA-92/S augmented reality display.
- Sensics zSight head tracker.

For fielding in simulation training facilities, VRSG works with simulated military equipment used by forward air controllers (FACs) or JTACs, to illuminate targets for aircraft and artillery strikes. These simulated devices have a tracker embedded in the hardware and are also discussed in this chapter. VRSG provides support for:

- Emulated hardware.
- Notional emulated hardware.

If you want to use another VR or other tracking system with VRSG, contact MVRsimulation to see whether support for your system has been added since this user’s guide was published. MVRsimulation is open to suggestions for adding VRSG support for other VR systems.

Setting up commercial VR systems

VRSG natively supports any VR HMD that is compatible with OpenVR API / SteamVR.

Use of VRSG with a VR system is nearly the same for all systems. Most require two viewports; the Varjo HMDs each require four viewports. Settings include whether to render the scene in a single pass, render the controllers and/or base station, and resolution refinements for foveated rendering.

VRSG support for the Varjo line of HMDs: Varjo VR-2, Varjo VR-3, and the Varjo XR-1 and Varjo XR-3 mixed-reality system is through a VRSG plugin, VarjoHMD.dll located in the directory \MVRsimulation\VRSG\Plugins\HMD\.

MVRsimulation VRSG real-time screenshot taken in an HTC-VIVE Pro headset.

Upon startup, VRSG scans for the VR tracking system and displays a confirmation message, as described in the section "Running VRSG with a tracker or HMD." By default, VRSG assumes the VR system's seated tracking mode.

You can write a VRSG plugin to interact with the controllers for actions like button presses and analog trigger values using the appropriate function in the Plugins.h header file. Use the function:

- `extViveControllerState` for VIVE systems.
- `extOdysseyControllerState` for Windows Mixed Reality systems.

See the chapter "Developing VRSG Plugins" for more information about VRSG's plugin interface.

Configuring a VR system

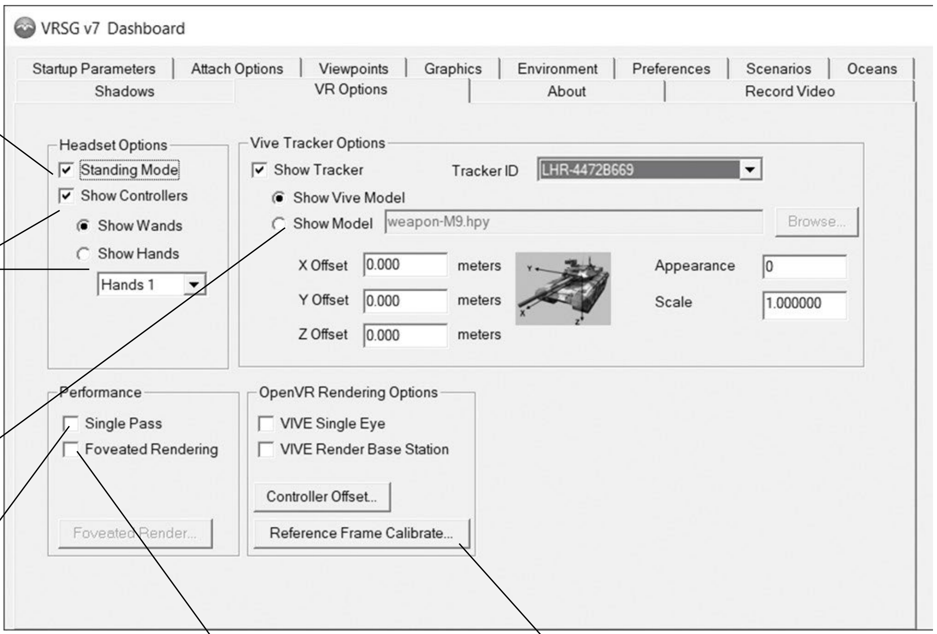
You control various settings for running VRSG with a VR system on the Dashboard's VR Options tab, with the exception of Varjo systems, which you configure on their own tabs, as described later in this section. By default, VRSG assumes the VR system is in seated tracking mode. To use the VR system in standing mode, select the Standing Mode checkbox on the VR Options tab.

Usage of VRSG with a VR system is nearly the same for all systems, with a few differences mentioned in this section.

Select Standing Mode to override VRSG's default seated mode.

Select Show Controllers, and then choose to show the wands in the virtual world as wand/touch models or to depict them as a set of hand models.

Options for showing the VIVE tracker, whether to display the tracker as a weapon model and the offsets for positioning the model.



[✓] Standing Mode
[✓] Show Controllers
(●) Show Wands
( ) Show Hands
Hands 1
[✓] Show Tracker
Tracker ID LHR-4472B669
(●) Show Vive Model
( ) Show Model weapon-M9.hpy
X Offset 0.000 meters
Y Offset 0.000 meters
Z Offset 0.000 meters
Appearance 0
Scale 1.000000
[ ] Single Pass
[ ] Foveated Rendering
[ ] VIVE Single Eye
[ ] VIVE Render Base Station

Renders the scene in a single pass (rather than the default two passes – one for each eyepiece).

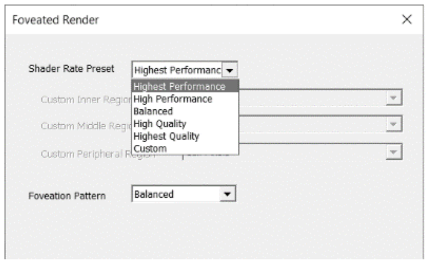
Click for options to refine foveated rendering.

Calculates the difference between the reference frames of the Windows Mixed Reality VR headsets and the Vive tracker for using them together.

To display VR motion controllers in the scene, select Show Controllers, and then choose to show the controllers in the scene either as hand models or controller models as shown in the examples below. VRSG has two sets of hands models; select one from the drop-down menu. The Performance options:

- Direct VRSG to render the scene in a single pass (rather than the default two passes – one for each eye) thus reducing the rendering burden on the system.
- Enable foveated rendering with full control over the resolution of each viewing region via several refinement settings. To minimize the rendering burden you could set the highest level of detail to render only in the region in the dead center of the viewing region.

Foveated Rendering improves performance by reducing image quality along the peripheral areas of the screen. Click the Foveated Render button to display the available options and their possible settings:



Foveated Render

Shader Rate Preset: Highest Performanc

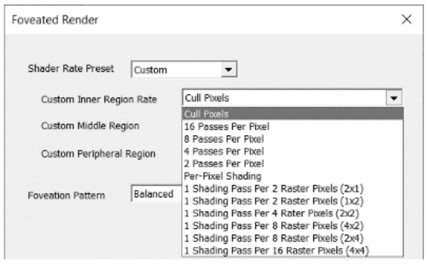
Highest Performance
High Performance
Balanced
High Quality
Highest Quality
Custom

Custom Inner Regio

Custom Middle Regi

Custom Peripheral Region

Foveation Pattern: Balanced



Foveated Render

Shader Rate Preset: Custom

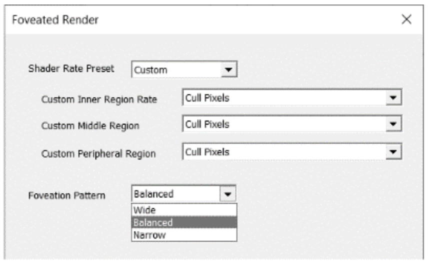
Custom Inner Region Rate: Cull Pixels

Cull Pixels
16 Passes Per Pixel
8 Passes Per Pixel
4 Passes Per Pixel
2 Passes Per Pixel
Per-Pixel Shading
1 Shading Pass Per 2 Raster Pixels (2x1)
1 Shading Pass Per 2 Raster Pixels (1x2)
1 Shading Pass Per 4 Rater Pixels (2x2)
1 Shading Pass Per 8 Raster Pixels (4x2)
1 Shading Pass Per 8 Raster Pixels (2x4)
1 Shading Pass Per 16 Raster Pixels (4x4)

Custom Middle Region

Custom Peripheral Region

Foveation Pattern: Balanced



Foveated Render

Shader Rate Preset: Custom

Custom Inner Region Rate: Cull Pixels

Custom Middle Region: Cull Pixels

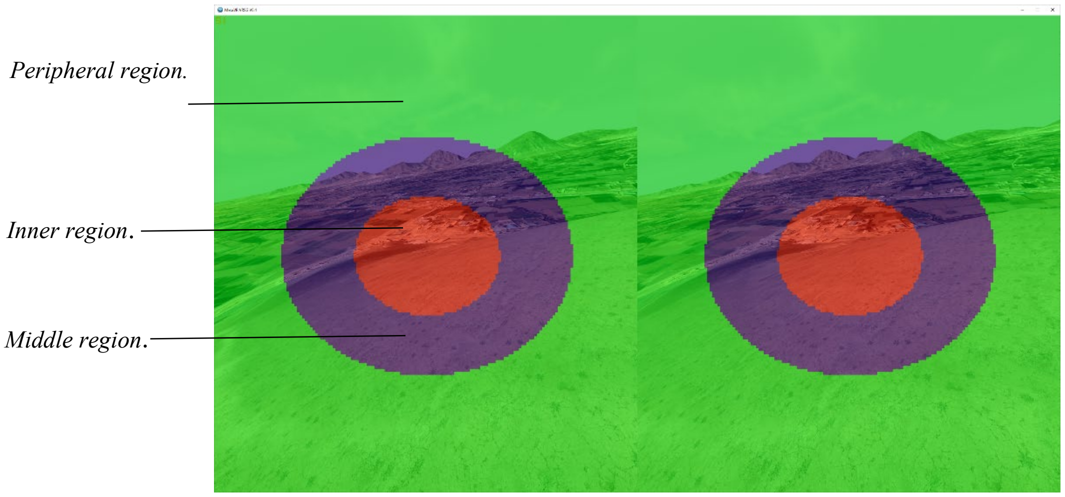
Custom Peripheral Region: Cull Pixels

Foveation Pattern: Balanced

Wide
Balanced
Narrow

Shader Rate Preset defines the standard image quality options available when using Foveated Rendering. The Highest Quality setting renders the image pixels at their highest detail, while the Highest Performance renders the image pixels at their lowest quality, thus improving VRSG performance. If you choose the Custom setting, you can define image quality for each of the three regions of the image.

Foveation Pattern sets the pixel resolution of the three viewing regions as shown:



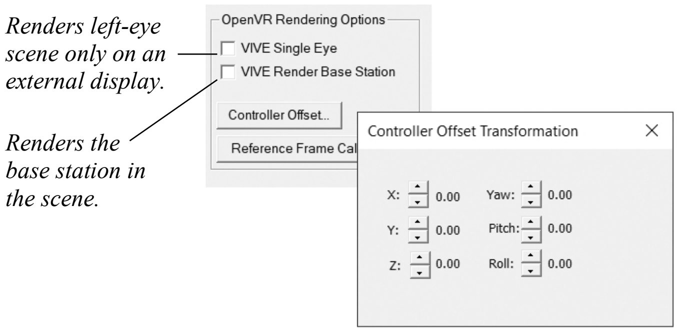
For example, to optimize performance, you could set the Inner Region (shown in red in the example above) to render at the highest image quality and the Peripheral and Middle Regions as rendering at a lower image quality. This results in highly detailed pixels rendered in the center of the viewer's gaze and lower detailed pixels on the peripheral of the viewer's gaze.

Foveation Pattern controls the diameter/pixel area of the three regions. The Balanced setting is shown in the image above. The Narrow setting shrinks the Inner and Middle regions. The Wide setting expands the Inner and Middle regions.

The OpenVR Rendering options:

- Instruct VRSG to render the left eye scene only ("Single Eye") on an external display, rather than rendering the scene for both eyes on the display in a split-screen manner). This option is useful in cases where the OTW view in the headset is also being rendered on an external display.

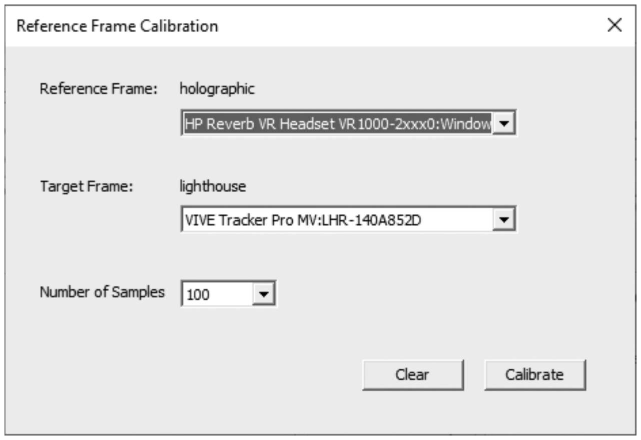
- Render the base station in the scene; useful for orienting yourself in relation to physical space or for setting the VIVE reference frame for a Windows Mixed Reality headset used with a VIVE tracker.



Click Controller Offset to set the x, y, z values for transforming a Windows Mixed Reality device used with a VIVE tracker to the VIVE reference frame.

If you use a VIVE tracker with the HP Reverb or another Windows Mixed Reality headset:

1. Click the Reference Frame Calibrate option to calculate and synch the reference frames of the headset and the VIVE tracker for using them together.


Reference Frame Calibration

Reference Frame: holographic
HP Reverb VR Headset VR1000-2xxx0:Window

Target Frame: lighthouse
VIVE Tracker Pro MV:LHR-140A852D

Number of Samples 100

2. In the Reference Frame field, which lists the headset(s) VRSG has detected, choose the headset to calibrate.
3. In the Target Frame field, choose the tracker.
4. For Number of Samples, choose the number of orientation and position data points of each device to sample for the calibration. The default number of 100 data points is usually sufficient.
5. Click Calibrate. While the calibration calculations are in progress, the message "Calibration" will be displayed as an overlay on the VRSG scene displayed in your headset, until the calibration is finished.

Configuring Varjo systems

VRSG provides support for all the Varjo VR and XR mixed-reality systems in the form of a DLL file. The file VarjoHMD.dll, installed with VRSG, resides in the subdirectory \MVRsimulation\VRSG\Plugins\HMD.

You can use Varjo XR-1 or XR-3 mixed-reality system with either a pass-through polygon mask or a green screen. A pass-through polygon is used within the ownship model to mask the portion of the scene that will be viewed as real-world pixels – instrument panels or other physical objects (as opposed to the rendered pixels of the virtual scene). For example, you could use a pass-through polygon to mask out a portion of virtual controls in a cockpit simulator, so the trainee can see and touch the physical controls. This masking geometry must match the shape of the physical surface that you want viewable within the HMD.

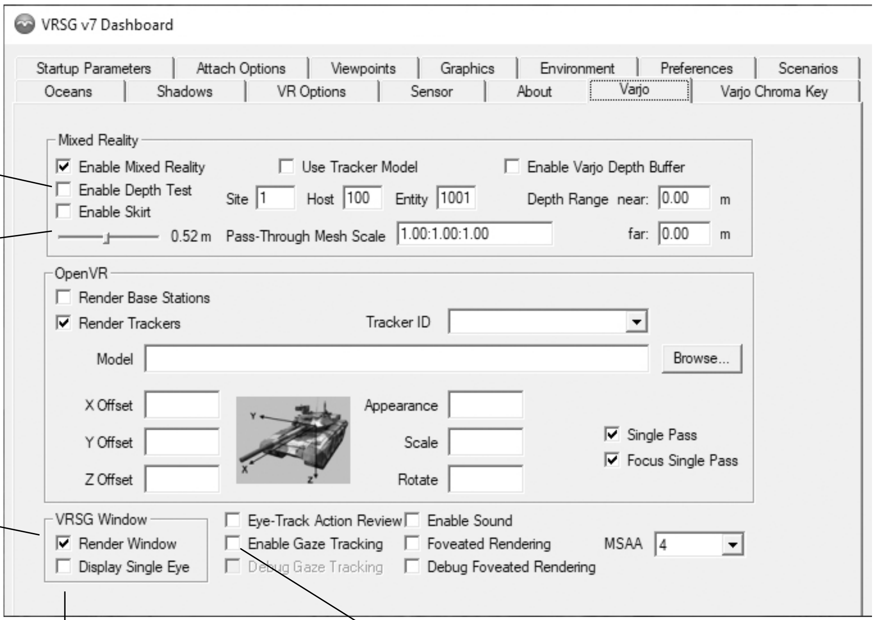
To use a Varjo device with VRSG, *copy* the VarjoHMD.dll up one directory level so that it is located directly in the \Plugins directory, as in: \MVRsimulation\VRSG\Plugins\VarjoHMD.dll. (*Note: Do not* move the .dll, simply copy it.)

When you start up VRSG with the VarjoHMD.dll in place, the Dashboard will contain two Varjo-specific tabs. Use the settings on these tabs instead of those on the VR Options tab. Some options on the Varjo tab are similar to the ones available on the VR Options tab.

Depth sorts the pass-through geometry mesh against the rest of the geometry in the virtual scene.

Increase or decrease the scale of the pass-through skirt polygon.

Renders the scene in an external display in addition to within the Varjo headset.



Mixed Reality
- [✓] Enable Mixed Reality
- [ ] Use Tracker Model
- [ ] Enable Varjo Depth Buffer
- [ ] Enable Depth Test
- [ ] Enable Skirt
- Site 1 Host 100 Entity 1001
- Depth Range near: 0.00 m far: 0.00 m
- 0.52 m Pass-Through Mesh Scale 1.00:1.00:1.00

OpenVR
- [ ] Render Base Stations
- [✓] Render Trackers
- Tracker ID
- Model
- X Offset, Y Offset, Z Offset
- Appearance, Scale, Rotate
- [✓] Single Pass
- [✓] Focus Single Pass

VRSG Window
- [✓] Render Window
- [ ] Display Single Eye
- [ ] Eye-Track Action Review
- [ ] Enable Gaze Tracking
- [ ] Debug Gaze Tracking
- [ ] Enable Sound
- [ ] Foveated Rendering
- [ ] Debug Foveated Rendering
- MSAA 4

Instructs VRSG to render only the left eye scene on an external display, rather than for both eyes in a split-screen manner.

Turns on the Varjo headset's ability to track the pupils of the wearer's eyes.

Mixed reality settings

In the Mixed Reality section (which pertains to Varjo XR-1 and XR-3 devices):

- Click the Enable Mixed Reality checkbox to turn on the mixed-reality functionality, which includes rendering the pass-through geometry within the virtual scene.
- Select Depth Test to have the pass-through geometry depth sorted against the rest of the geometry in the virtual scene. Otherwise, the pass-through geometry is always rendered on top of all geometry within the virtual scene.
- Enable Skirt activates a circular pass-through mesh below the Varjo eyepoint to see the physical world. The slider below Enable Skirt controls the diameter in meters of the circular pass-through mesh.
- Enter a Pass-Through Mesh Scale to increase or decrease the scale of the pass-through polygon mesh. The scale value is an x,y,z percentage scale. This field is useful adjusting the mask such that it matches the edge of the intended physical surface without any gaps.

  Note: See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats" for information about how to add a custom attribute to your own model called "PassThroughMesh" in your modeling tool. MVRsimulation's conversion utility, Fbx2Hpx, marks the geometry node appropriately when exporting your model to MVRsimulation's HPX format.
- When you are attached to a DIS entity, select Use Tracker Model to render that model assigned to the Vive Tracker instead of the DIS enumerated model identified in the ModelMap.ini file. This option is needed as the Vive Tracker and VR headset are both tracked within the same relative physical space. This option causes the model and headset to be relative to each other in the virtual world. Site:Host:Entity identifies the DIS ID of the entity that the camera is attached to.
- Click the Enable Depth Buffer to enable Varjo's depth range functionality. This allows for real-world objects (for example a user's hands) to be seen in front of the virtual scene. The Depth Range near/far values define the range in meters that real-world objects can be seen.

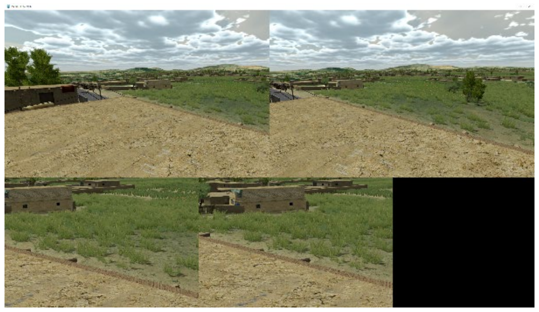
OpenVR and external VRSG window settings

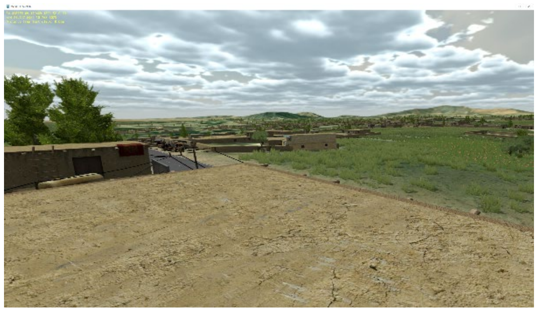
In the OpenVR section:

- Choose whether to show the base station.
- Choose whether to show the tracker(s), the ID of the tracker to show, and whether to display the tracker model. Also set and the offsets for positioning the tracker model, and its appearance and scale.

In the VRSG Window section:

- Render Window (default setting) displays the VRSG visualization window on an external display in addition to within the Varjo headset. If performance becomes an issue, unselect this checkbox to not have the scene rendered in the external VRSG window to improve performance. When this checkbox is unselected, the external VRSG window is displayed as black.
- Display Single Eye instructs VRSG to render only the left eye scene on an external display, rather than rendering the scene for both eyes in a split-screen manner. This option is useful in cases where the OTW view in the headset is also being rendered on an external display.


The four VRSG viewports rendered inside a VR HMD shown in a split-screen manner on an external display.

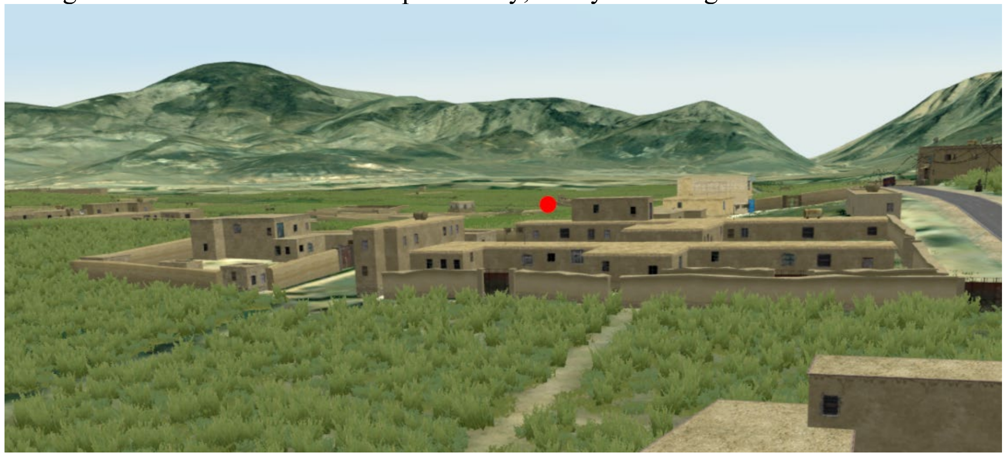

When Display Single Eye is checked, the external display will only show the left eye view.

Eye tracking

VRSG includes several options for using the Varjo headset's eye-tracking data collection capability.

- Eye-Track Action Review inserts into the network DIS stream the Varjo headset's head position and (optionally) pupil-tracking data. Enable Gaze Tracking must also be turned on (and calibrated) for capturing pupil-tracking data.
- Enable Gaze Tracking turns on the Varjo headset's ability to track the pupils of the wearer's eyes. Selecting this checkbox starts Varjo's setup prompts to calibrate the exact pupil location for the session. Selecting this checkbox sends the DIS packets needed for other instances of VRSG to render the eye-tracking, as described in the section "Visualizing the eye tracking data in VRSG."
- Debug Gaze Tracking renders a red dot at the location of the pupil gaze, drawing from the eye-tracking data, as shown below. The display allows the headset wearer to immediately determine if the eye tracking data corelates correctly to the actual location of

their gaze. If the dot does not line up correctly, the eye-tracking needs to be recalibrated.

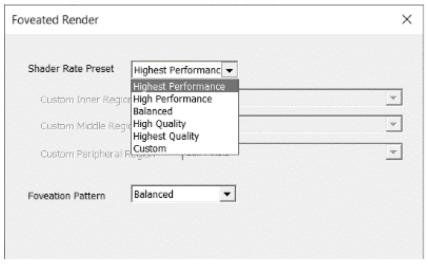


Enabling sound

Check the Enable Sound option to use the HMD’s position in geocentric space for all sound calculations.

Foveated rendering

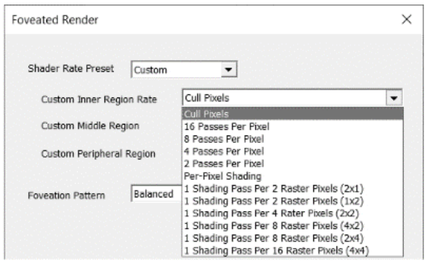
Foveated Rendering improves performance by reducing image quality along the peripheral areas of the screen. Click the Foveated Render button to display the available options and their possible settings:



Foveated Render

Shader Rate Preset: Highest Performanc

Foveation Pattern: Balanced



Foveated Render

Shader Rate Preset: Custom

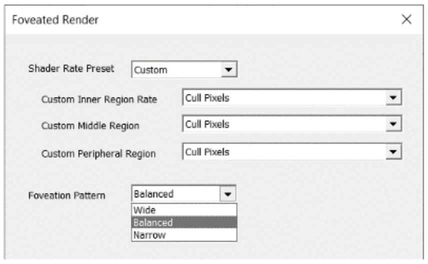
Custom Inner Region Rate: Cull Pixels

Custom Middle Region

Custom Peripheral Region

Foveation Pattern: Balanced

Cull Pixels
16 Passes Per Pixel
8 Passes Per Pixel
4 Passes Per Pixel
2 Passes Per Pixel
Per-Pixel Shading
1 Shading Pass Per 2 Raster Pixels (2x1)
1 Shading Pass Per 2 Raster Pixels (1x2)
1 Shading Pass Per 4 Rater Pixels (2x2)
1 Shading Pass Per 8 Raster Pixels (4x2)
1 Shading Pass Per 8 Raster Pixels (2x4)
1 Shading Pass Per 16 Raster Pixels (4x4)



Foveated Render

Shader Rate Preset: Custom

Custom Inner Region Rate: Cull Pixels

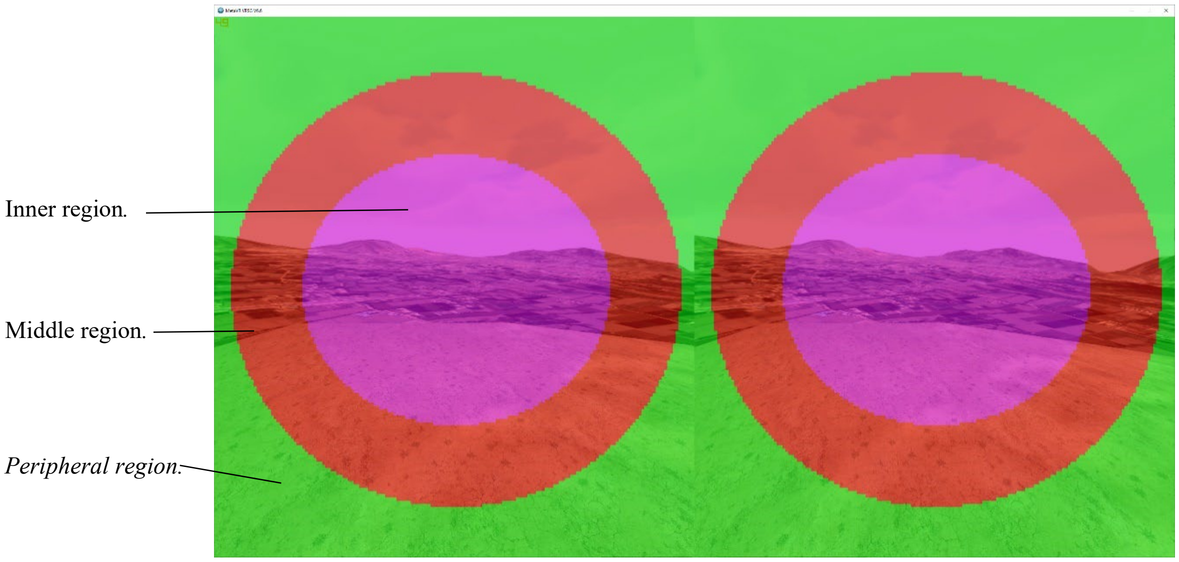
Custom Middle Region: Cull Pixels

Custom Peripheral Region: Cull Pixels

Foveation Pattern: Balanced

Shader Rate Preset defines the standard image quality options available when using Foveated Rendering. The Highest Quality setting renders the image pixels at their highest detail, while the Highest Performance renders the image pixels at their lowest quality, thus improving VRSG performance. If you choose the Custom setting, you can define image quality for each of the three regions of the image.

Foveation Pattern sets the pixel resolution of the three viewing regions. For example, to optimize performance, you could set the Inner Region (shown in pink in the example below) to render at the highest image quality and the Peripheral and Middle Regions as rendering at a lower image quality. This results in highly detailed pixels rendered in the center of the viewer’s gaze and lower detailed pixels on the periphery of the viewer’s gaze.



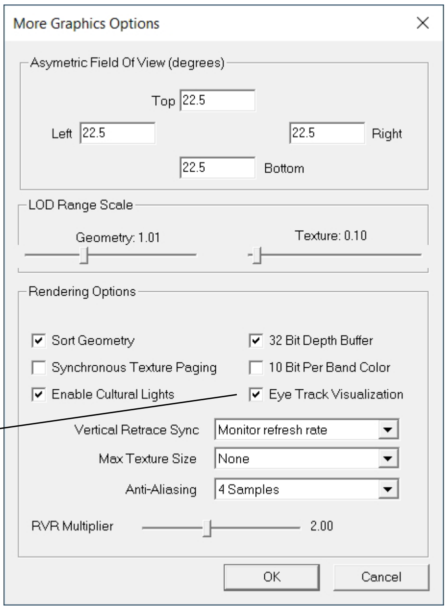
The Foveation Pattern controls the diameter/pixel area of the three regions. The Narrow setting shrinks the Inner and Middle regions. The Wide setting expands the Inner and Middle regions.

Debug Foveated Rendering displays the colorized regions of resolution density as shown above.

Visualizing the eye tracking data in VRSG

In real time, an instructor can watch VRSG visualizing the Varjo headset wearer's head position and eye-tracking data during a training mission on a separate VRSG channel in stealth mode. To turn on this visualization, select the Eye Tracking Visualization checkbox on the More Graphics Options dialog box (accessed from the Dashboard's Graphics tab).

Turns on visualization of the headset wearer's head position and eye gaze (in real time or during PDU log playback).


More Graphics Options

Asymetric Field Of View (degrees)

Top 22.5
Left 22.5
Right 22.5
Bottom 22.5

LOD Range Scale

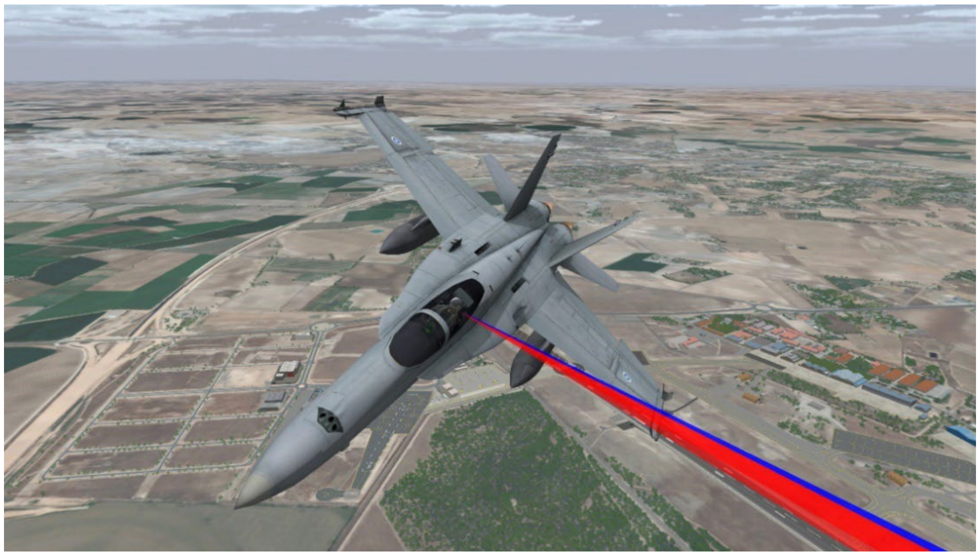
Geometry: 1.01
Texture: 0.10

Rendering Options

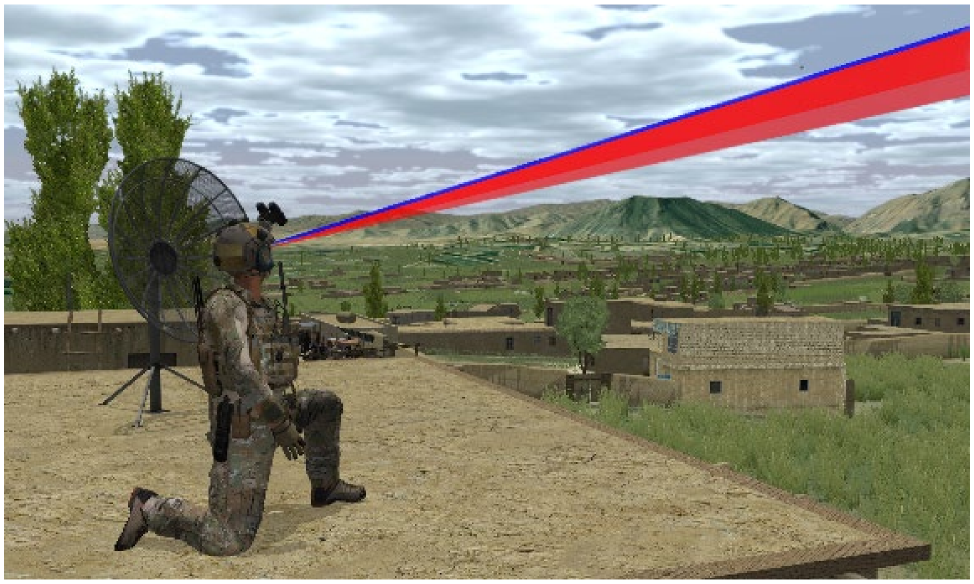
- [✓] Sort Geometry
- [✓] 32 Bit Depth Buffer
- [ ] Synchronous Texture Paging
- [ ] 10 Bit Per Band Color
- [✓] Enable Cultural Lights
- [✓] Eye Track Visualization

Vertical Retrace Sync: Monitor refresh rate
Max Texture Size: None
Anti-Aliasing: 4 Samples
RVR Multiplier: 2.00

The visualization depicts the head position via a head model and the tracked pupil direction via red and blue cones. The Varjo wearer's head position and eye-tracking data can also be recorded to a PDU log and visualized in the same way for after action review (AAR) purposes. Tools for recording a PDU log include MVRsimulation's Playback tool, BSI's Discord (for users who run VRSG with MACE), and Plexsys's Voice, Audio, and Data for After Action Review (VADAAR, for users who want to record more than just DIS entities).



VRSG real-time stealth view of a simulated air mission, rendering the eye-tracking data (head position and pupil gaze) from the pilot trainee wearing a Varjo headset.

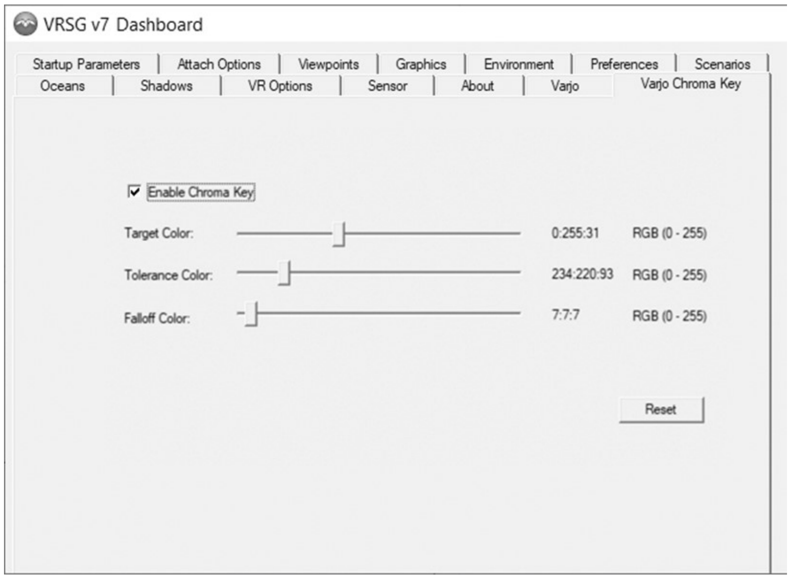


VRSG real-time stealth view of a simulated close air support mission, rendering the eye-tracking data from a JTAC trainee wearing Varjo headset.

Use a MSAA (multi-sampling anti-aliasing) option to smooth out the edges of polygons for rendering the VRSG scene in the Varjo headset. Choose a multi-sampling value of 1x, 2x, or 4x.

Green screen color settings

The Dashboard's Varjo Chroma Key tab contains settings for defining the color matching needed when using green screen material.



☑ Enable Chroma Key

Target Color:	0:255:31	RGB (0 - 255)
Tolerance Color:	234:220:93	RGB (0 - 255)
Falloff Color:	7:7:7	RGB (0 - 255)

1. Click Enable Chroma Key.
2. Either accept the default settings or use the sliders to specify different color values for Target, Tolerance, or Falloff Color to better match the color of your particular green screen material.
3. Click the Reset button to reset to Varjo default green screen values.

Binocular zoom mode

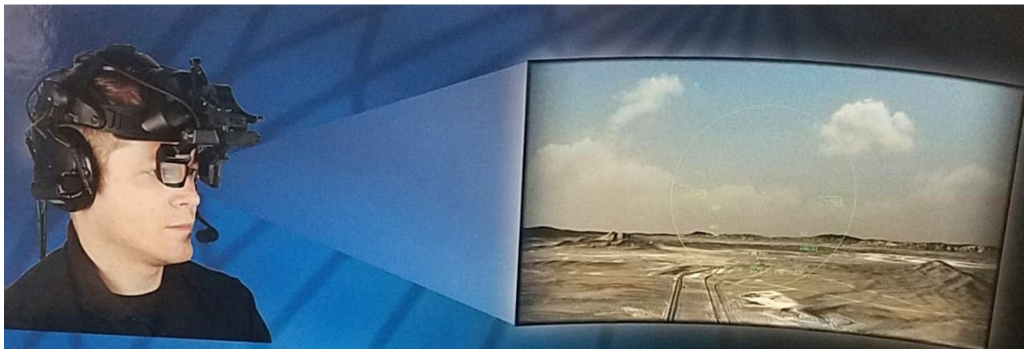
The VarjoHMD.dll plugin also adds a binocular zoom mode for the Varjo HMDs. Binocular Zoom mode requires the use of a Logitech F310 gamepad. For more information about using the Logitech F310 gamepad in FPS mode see the chapter "Using 3D Characters in VRSG."

The Right Top (RT) and Right Bottom (RB) buttons control the zoom feature. Binocular Zoom mode contains three levels of magnification: 25, 15 and 5 degrees field of view. To step through the different magnification levels, press the RB button. The RB button will continue to cycle through the different magnification levels. To return to normal mode, press the RT button.

A notional M22 reticle is visible in the center of the entity's viewpoint. Use the right joystick to control the position of the reticle while in Binocular Zoom mode. When you return to normal view, the reticle will not be visible.

Configuring the SA Photonics SA-62 and SA-92S HMDs

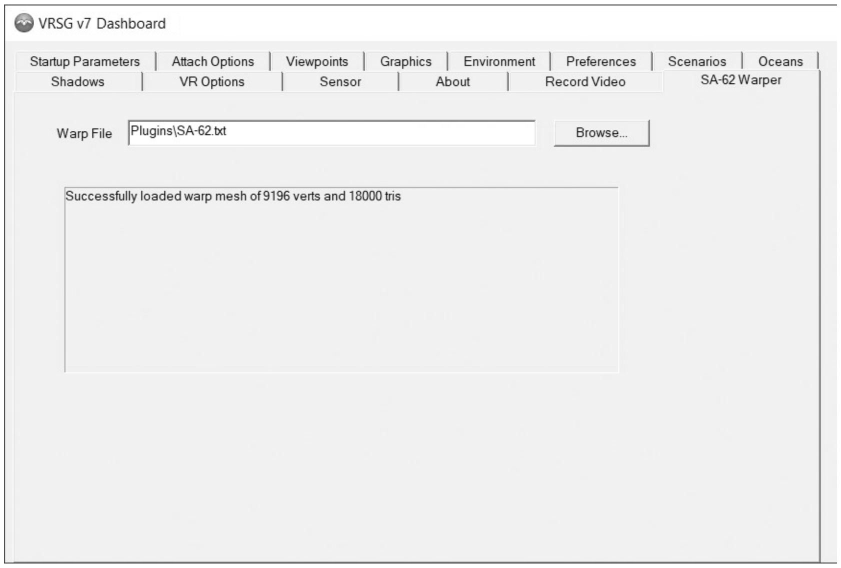
VRSG provides support for the SA Photonics SA-62 VR and SA-92S AR systems in the form of a DLL file.


ZedaSoft's EyeBox, with an SA-Photonics SA-92S HMD with see-through 1920x1200 resolution per eye optics, a Polhemus head-tracker, and VRSG channel (shown on the right). Photo courtesy of ZedaSoft.

You can obtain the file SA-62.dll or SA-92.dll, on MVRsimulation's Downloads Server under /Software/Plugins/Displays/.

To use the plugin, copy or move the SA-62.dll or SA-92.dll so that the file is located directly in the \Plugins directory, as in: \MVRsimulation\VRSG\Plugins\SA-92.dll. Once the SA-62.dll or SA-92.dll is present directly in the \Plugins directory, VRSG can detect the tracker on startup and will display a tab on the Dashboard for the appropriate device.

On the SA-62 Warper tab, provide a distortion mesh file.

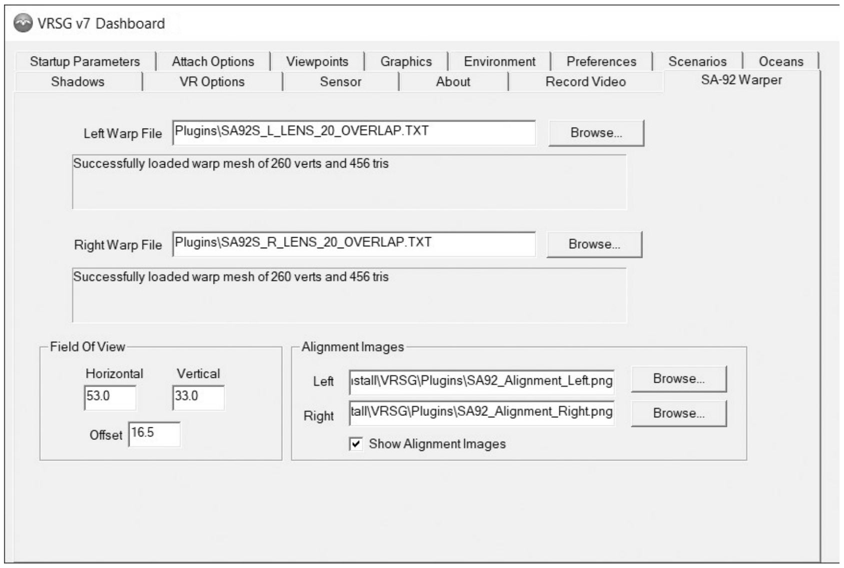


Warp File: Plugins\SA-62.txt

Successfully loaded warp mesh of 9196 verts and 18000 tris

On the SA-92 Warper tab provide:

- A distortion mesh, one for each eye.
- The horizontal and vertical FOV, and the angular offset.
- Optionally, select the Show Alignment Images checkbox and browse for an image for each eye.


VRSG v7 Dashboard

Left Warp File: Plugins\SA92S_L_LENS_20_OVERLAP.TXT

Successfully loaded warp mesh of 260 verts and 456 tris

Right Warp File: Plugins\SA92S_R_LENS_20_OVERLAP.TXT

Successfully loaded warp mesh of 260 verts and 456 tris

Field Of View — Horizontal: 53.0, Vertical: 33.0, Offset: 16.5

Alignment Images — Left: ıstall\VRSG\Plugins\SA92_Alignment_Left.png; Right: tall\VRSG\Plugins\SA92_Alignment_Right.png

[✓] Show Alignment Images

Setting up commercial trackers

MVRsimulation's tracker support for InterSense trackers and the SA Photonics HMDs is in the form of a DLL plugin.

VRSG support for the Flock of Birds, Polhemus Scout, and Sensics zSight head trackers is native; no third party DLLs are required. The HTC VIVE virtual reality system requires OpenVR.dll, which installed with VRSG. Upon startup, VRSG will scan for the tracker as described in the section "Running VRSG with a tracker."

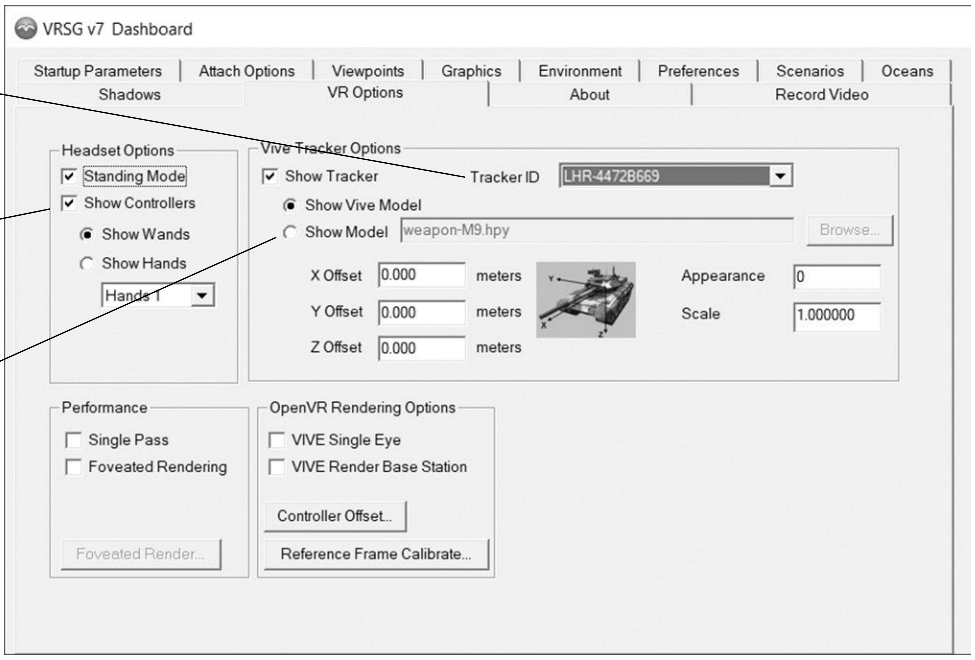
Configuring the HTC VIVE VR system tracker

VRSG supports the HTC VIVE Tracker. Once you start it and VRSG recognizes it, you can optionally display it in the scene and associate it with a model other than the default tracker mesh model (such as a weapon model from MVRsimulation's model libraries) and set the offsets to position the model. You make this association in the VR Options tab of the VRSG Dashboard as show in the following example:

Which tracker the settings apply to (in the case of multiple trackers).

Options for showing the controllers onscreen.

Options for showing the tracker as a model in the VRSG scene.


VRSG v7 Dashboard

Tabs: Startup Parameters | Attach Options | Viewpoints | Graphics | Environment | Preferences | Scenarios | Oceans | Shadows | VR Options | About | Record Video

Headset Options
- [✓] Standing Mode
- [✓] Show Controllers
  - [● Show Wands]
  - ○ Show Hands
  - Hands 1

Vive Tracker Options
- [✓] Show Tracker — Tracker ID: LHR-4472B669
- [● Show Vive Model]
- ○ Show Model: weapon-M9.hpy (Browse...)
- X Offset: 0.000 meters
- Y Offset: 0.000 meters
- Z Offset: 0.000 meters
- Appearance: 0
- Scale: 1.000000

Performance
- [ ] Single Pass
- [ ] Foveated Rendering
- Foveated Render...

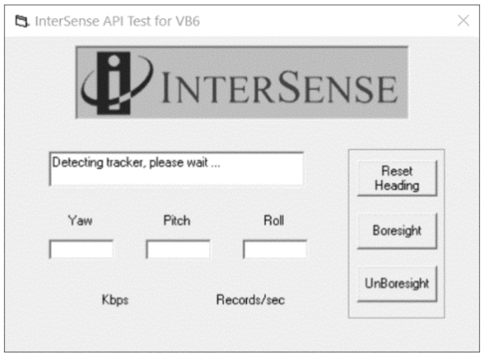
OpenVR Rendering Options
- [ ] VIVE Single Eye
- [ ] VIVE Render Base Station
- Controller Offset...
- Reference Frame Calibrate...

Configuring the InterSense tracker

VRSG provides support for all InterSense trackers in the form of a dll file. The file isense64.dll, installed with VRSG, is all that you need to use VRSG with an InterSense tracker.

To configure VRSG's InterSense tracker support:

1. Connect the tracker to the computer on which you will be running VRSG, and turn on the tracker.
2. In the VRSG installation directory, locate the utility \MVRsimulation\VRSG\Plugins\Tracker\Intersense\vbtrax.exe. Double-click the utility to run it.


InterSense API Test for VB6

Detecting tracker, please wait ...

3. If the vbtrax.exe detects the tracker and it appears to be functional, *copy* the file isense64.dll from the \MVRsimulation\VRSG\Plugins\Tracker\Intersense subdirectory so that it resides directly in the \Plugins directory, as in: \MVRsimulation\VRSG\Plugins\isense64.dll. (Note: *Do not* move the .dll, simply copy

it.) A message "Unknown tracker model" might appear if the device is newer than the vbtrax utility.

4. Start VRSG. VRSG should detect the tracker.

Once the isense64.dll is present directly in the \Plugins directory, VRSG can detect the tracker on startup. Doing so takes a few seconds at startup, therefore if you do not use a tracker with VRSG, do not move the .dll into the \Plugins subdirectory.

The InterSense InertiaCube4 tracker can be fitted within NVIS Virtual Binoculars and other simulated military equipment to offer an integrated tracker device, with programmable buttons, and a high-resolution display. As described later in this chapter, the buttons on the devices can be mapped to laser ranging and designation, resulting in a low-cost generic device for JTAC simulation training.

Note: These InterSense inertial tracking systems define the zero orientation by the orientation of the sensor on power-up; therefore, make sure your sensor is aligned with the logical platform before powering up the tracker.

Inertial systems are subject to drift after long periods of usage. To avoid significant accumulation of drift, you should power cycle the tracker before each VRSG session.

Configuring NaturalPoint's TrackIR 5 tracker

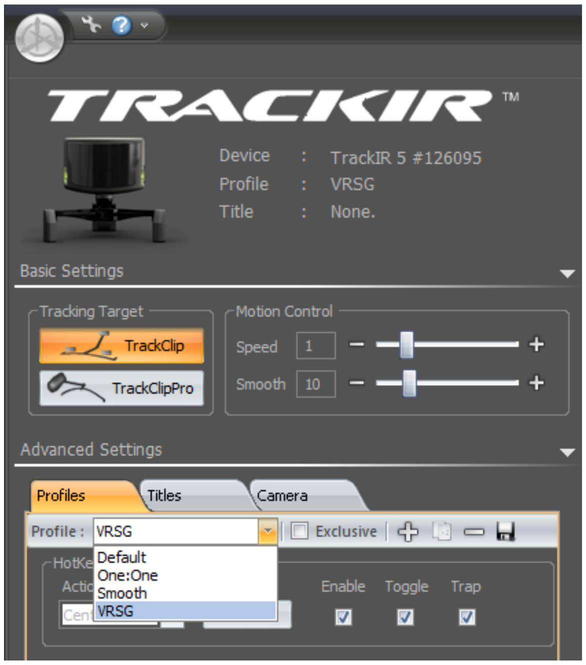
VRSG supports NaturalPoint's TrackIR 5. This 6-DOF head-tracking input device consists of an infrared camera that sits on the top of a computer monitor and a sensor/tracking clip that you attach to the brim of a baseball cap or visor, which you wear during the VRSG session. The tracking clip has three reflective markers that direct infrared light back to the camera. As you move and rotate your head small amounts while facing the monitor, the camera tracks the position of your head via the tracking clip and the VRSG eyepoint shifts accordingly.

The TrackIR 5 software is not distributed with the hardware package (which contains the camera and tracking clip); VRSG support is available in the TrackIR 5 installation; which can be downloaded from the trackir.com website.

TrackIR must be running concurrently for VRSG to auto-detect and utilize the device.

To use TrackIR with VRSG:

1. Place the camera on the top of your computer monitor and affix the clip to the brim of the baseball cap or visor you will wear during VRSG sessions.
2. Start the TrackIR software. The software must be running for VRSG to auto-detect and utilize the device.
3. On the Profiles select the VRSG profile, as shown next:



As result, when you start VRSG, the VRSG profile is selected automatically. The main benefit of this profile is that it enables FPS, which is bound to the F9 key (without this profile, F9 is used for another function by TrackIR).

The tracker device exaggerates head orientation so that you can easily look left/right and up/down with much larger angles than head motion requires. You can configure the angular and positional gains of the exaggeration in the TrackIR software.

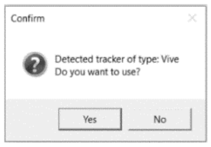
With FPS, the tracker is most suitable in first-person or the sensor modes. The tracker input replaces the right thumbstick for head movement.

Running VRSG with a VR system or tracker

Upon startup, VRSG automatically scans for any VR or tracking system installed and running on your system.

To run VRSG with a VR or other tracking device:

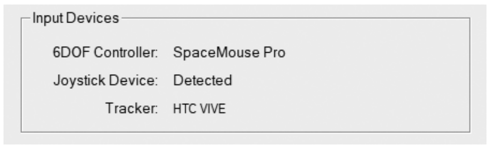
1. Click Start VRSG to display the scene visualization. VRSG automatically detects the VR or other tracking device and, for most, displays a message confirming whether you want to use the tracking device it detected. (This message does not appear upon detection of Varjo devices.)


Confirm

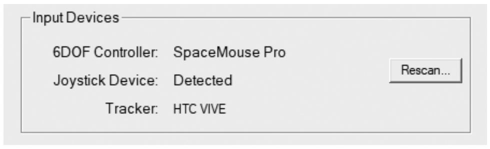
Detected tracker of type: Vive
Do you want to use?

2. Click Yes in the confirmation message box to use the tracker to calculate your orientation in the VRSG view.

When VRSG detects the presence of a VR system or other tracking device and accepts it for use, it displays the tracker status in the Input Devices section on the Dashboard's Startup Parameters tab, as shown.


Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: HTC VIVE

If you plug in a VR system or other the tracking device while VRSG is already running, VRSG detects that a USB device has been plugged in and displays a Rescan button.


Input Devices
6DOF Controller: SpaceMouse Pro
Joystick Device: Detected
Tracker: HTC VIVE

Click Rescan for VRSG to identify the device as the tracker.

When you use a tracker, VRSG adds any positional and orientation inputs to that which is being supplied by the 6-DOF controller. One way to think of this is that the 6-DOF controller sets the position and orientation of the platform relative to the terrain coordinate system, and the tracker inputs set the position and orientation of the viewpoint relative to the platform's coordinate system.

Summary of startup command-line options for controlling use of tracking system

VRSG supports multiple startup command-line options, some of which pertain to use of a tracking system.

`-noTrackerPrompt` suppresses the tracker confirmation prompt and have VRSG always use the tracker.

`-ignoreTracker` prevents VRSG from scanning for a tracking system.

`-reverseTracker` corrects the reverse direction in tracker azimuth and pitch angles in tracking sensors mounted on HMDs or simulated military equipment in a reverse direction.

`-crosshair` instructs VRSG to display the crosshair reticle; useful for running VRSG with simulated military equipment such as binoculars or laser range finder devices.

For more information about starting VRSG at the command-line and all available command-line options, see the chapter "Exploring the VRSG System."

Running VRSG with simulated military equipment

When a simulated laser designator or target locator device is coupled with VRSG, VRSG generates the simulated sensor 3D scene and the range and coordinates of the designated target. When an operator laser designates a target, a DIS PDU is transmitted to indicate the range designation information to other simulations on the network.

VRSG supports several simulated military equipment devices, described in this chapter. If you have a simulated device you want to use with VRSG that is not described here, contact

MVRsimulation Support (support@mvrsimulation.com) to request a sample plugin project that illustrates how to implement a VRSG plugin for a simulated military equipment (SME) device.

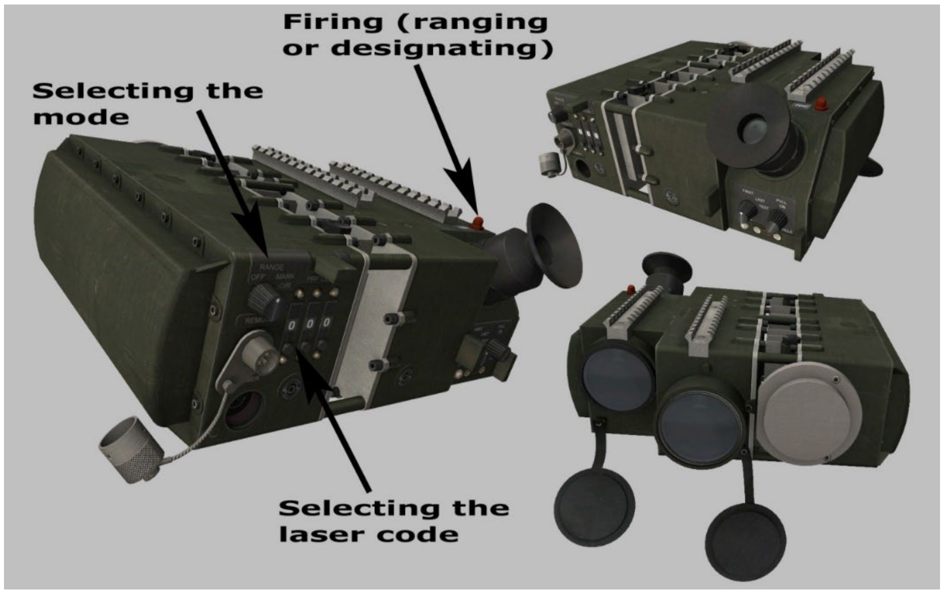
Several types of emulated military equipment interoperate with VRSG including laser target designators such as the Special Operations Forces Laser Acquisition Marker (SOFLAM) and Ranger 47 along with other equipment like the Infrared Zoom Laser Illuminator Designator (IZLID). Emulated hardware interoperates with the virtual environment to provide form-fit-function tactile feedback for certain processes including designating a target, reading coordinates of an entity and setting the JTACs Pulse Repetition Frequency (PRF) laser code.

On the left is the Observer station from of MVRsimulation's Deployable Joint Fires Trainer with an emulated SOFLAM to the right mounted to a free-standing tripod. On the right is a view through the lens of the emulated SOFLAM.

Emulated SOFLAM

This emulated SOFLAM uses the same housing as a field representative PEQ-1B SOFLAM integrated with Battlespace Simulations' (BSI's) MACE Integration Board. As result, the emulated SOFLAM has the same form, fit, and functions as the actual field representative device. All modes and sub-modes of the SOFLAM are simulated. Device state information as well as the ability to set and clear faults are available as a poll/response DIS stream from the device.

The emulated SOFLAM display system uses a single HDMI input to an internal OLED micro display and driver board, delivering a display resolution of 1280 x 1024 pixels (SXGA). The device is equipped with a 3 DOF tracking device. All device state and orientation information is handled over a single driverless USB connection to the host computer, and requires a VRSG plugin (SOFLAM_Plugin.dll, supplied by BSI) to integrate into the training environment.



Hardware setup

To set up the emulated SOFLAM device hardware:

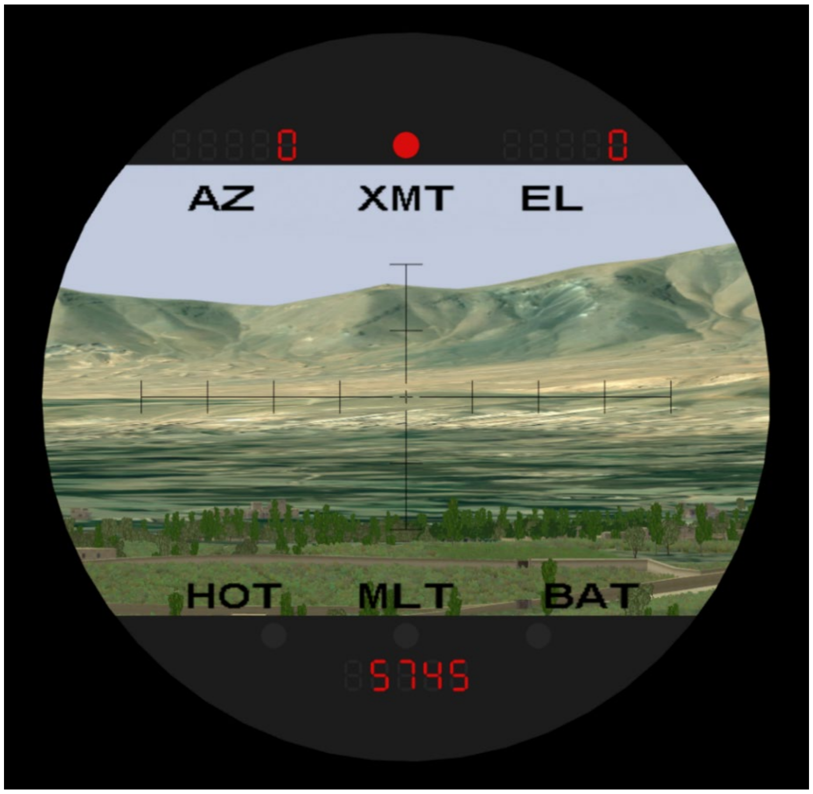
1. Plug the HDMI cable into an available HDMI output on the VRSG computer that will be managing the SOFLAM view.
2. Plug the 5V power adapter for the internal eyepiece display into a power outlet and insert the male end of the power adapter cable into the female canon plug on the SOFLAM.
3. Plug the USB cable from the SOFLAM into an available USB port on the host (VRSG) computer.

Note: Continuously powering the display when the SOFLAM is not in use can result in a shortened display lifespan, and could result in image burn-in. When the device is not in use, unplug the display power cable.

Software setup

Using the BSI SOFLAM with VRSG is straightforward.

Place the SOFLAM VRSG plugin, SOFLAM_Plugin.dll, directly in the VRSG Plugins directory (MVRsimulation\VRSG\Plugins) and start VRSG. When VRSG loads completely, you will see the SOFLAM field-of-view (FOV) in the VRSG viewport, as shown in the next example:



The VRSG display as seen in the emulated SOFLAM.

The eyepoint represents the eyepoint set in VRSG. If it is attached to an entity, the eyepoint will represent the attached entity eyepoint. The dynamic FOV is set in MACE under the platform's ownship equipment FOV setting located in System Settings>Visual>Ownship Equipment X, where X represents the Site:App:Entity associated with the SOFLAM. The SOFLAM typically operates in a FOV of 5 or 8 degrees depending on its original configuration. When the SOFLAM fires, a designator PDU is sent out from VRSG with source and target information, as well as specifics on the laser mode the SOFLAM is set to (see SOFLAM modes).

SOFLAM modes

The SOFLAM can be set to one of three operational modes – Range, Mark, and Override (ORIDE). If the device is set to Range, the designator PDU sent from VRSG represents a laser range finder spot and has a set wavelength of 1.55 µm. If it is set to Mark or ORIDE, the designator PDU sent from VRSG represents a laser target designator spot with a set wavelength of 1.064 µm. The designator PDU also encodes the PRF laser code set on the SOFLAM. Only the last three digits of the PRF laser code are settable on the SOFLAM. The first digit is always 1. For example, to encode a PRF code of 1644, you would set "6-4-4" on the SOFLAM. When the SOFLAM is fired, the XMT warning light is illuminated in the SOFLAM overlay (as shown in the example above), and the range to the target in meters is shown at the bottom of the SOFLAM display.

First and last pulse acceptance modes

The real-world SOFLAM has to deal with situations where the dispersion of the laser spot causes it to "spill over" the edge of a target, resulting in multiple returns. The emulated

SOFLAM can emulate this spillover and force the operator to change laser pulse acceptance mode from "First" to "Last" in order to get a correct range value. This only happens when the instructor sets a multiple returns fault state on the device (MLT). See the section "SOFLAM faults" below for more information.

Test mode

The Test mode simply lights up all seven segment display segments and warning light LEDs in the SOFLAM overlay, just like the test mode on the real device.

Reticle

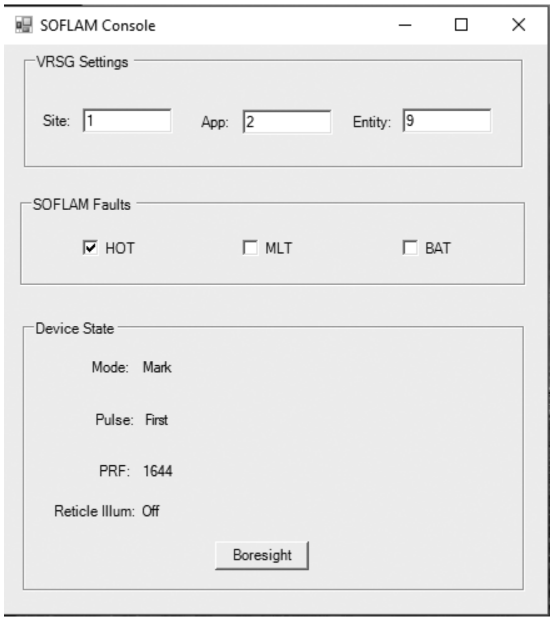
Pull on the reticle rheostat to turn on the "illuminate" aiming reticle in the device. Like the real device, this reticle mode is useful when trying to accurately aim the SOFLAM against a dark, low contrast background.

SOFLAM faults

The BSI SOFLAM supports setting and clearing faults within the SOFLAM device via DIS SetData PDUs. In addition, the BSI SOFLAM will periodically report the device state (set modes, PRF code, and so on) via a DIS data PDU. If the device is used with BSI's MACE, a MACE plugin, as described next, can be used to view SOFLAM device state, set device faults, and boresight the SOFLAM from a remote IOS console.

BSI MACE SOFLAM Console plugin

The BSI MACE SOFLAM Console plugin controls the configuration of the SOFLAM. Use the SOFLAM Console plugin to control which entity the SOFLAM is attached to, the warning lights in the VRSG view, and bore sighting the SOFLAM. You can also view the Device State from the SOFLAM Console.


SOFLAM Console

VRSG Settings
Site: 1 App: 2 Entity: 9

SOFLAM Faults
[✓ HOT] MLT BAT

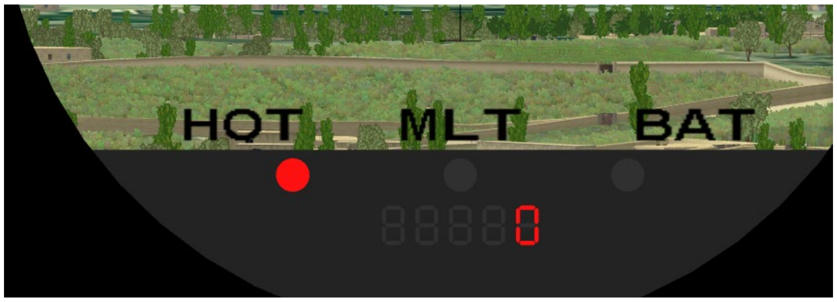
Device State
Mode: Mark
Pulse: First
PRF: 1644
Reticle Illum: Off

VRSG Settings

Set the Site:App:Entity to match the entity that the SOFLAM is attached to.

SOFLAM warning lights

When an instructor sets a device fault using the SOFLAM console plugin, the appropriate fault warning light is illuminated in the SOFLAM overlay. For example, with a "HOT" caution set, the "HOT" warning light is illuminated in the SOFLAM overlay, as shown in the next example:



The implications of the SOFLAM warning lights are summarized as follows:

- HOT – the laser in the SOFLAM is overheating and firing of the designator laser is inhibited. In order to fire the laser, switch the SOFLAM mode to ORIDE.
- MLT – the SOFLAM is receiving multiple returns from multiple targets. In the real world, this would illuminate as the laser is fired and multiple returns are received. In the emulated SOFLAM, this state is set by the instructor at his/her discretion. When it is set, the SOFLAM-generated range is only accurate in "Last" pulse acceptance mode. To use properly, we recommend that the SOFLAM be kept in "First" pulse acceptance mode (typical real-world SOP) and set the MLT fault when the JTAC is firing the laser in a scenario where the "last" pulse is the desired target pulse.
- BAT – low battery warning. This mode is indicates there is limited time remaining to operate the SOFLAM.

Boresighting the SOFLAM

SOFLAM view can be aligned to the center of the eyepoint view by pointing the physical SOFLAM in the direction you want to align to, and then click Boresight on the SOFLAM Console Plugin. The SOFLAM view is now aligned to the attached entity's eyepoint in the direction the SOFLAM is pointing to.

Troubleshooting

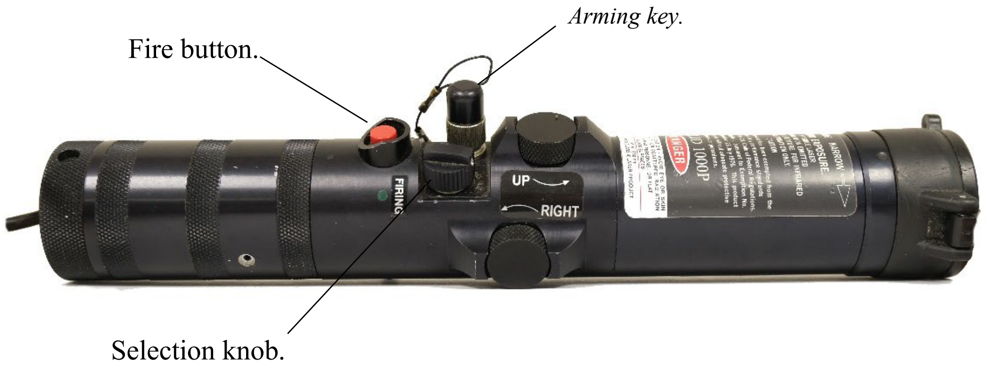
If the SOFLAM is not plugged in, or if the device fails to enumerate when it is plugged into the host computer, VRSG will display a "No SOFLAM attached" warning. When a SOFLAM is plugged in, VRSG will briefly display the notification, "SOFLAM plugged in. Attaching."

If the SOFLAM is plugged in but is failing to track, try simply "hot plugging" the USB cable by removing it from the host computer and plugging it back in. Sometimes the SOFLAM will enumerate properly and VRSG will properly detect and attach to the device, but it will fail to track. This issue is often caused by faulty USB cables or USB ports. Long unpowered cables (greater than 3 meters) can sometimes cause problems, as can low-cost USB extensions.

Emulated IZLID

VRSG's Night Vision Googles (NVG) mode supports laser illumination of targets with an emulated Infrared Zoom Laser Illuminator Designator (IZLID). The emulated IZLID uses a

3DOF sensor to control the view of the laser beam in the virtual world. The laser beam creates a splash when it interacts with culture within VRSG.

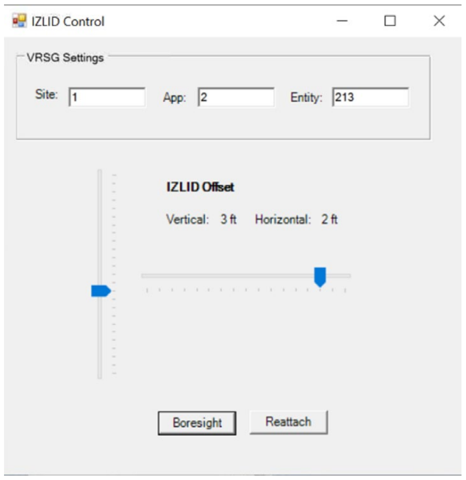


The Arming key must be installed for the IZLID to power on and function. Screw the arming key until tight. The Selection Knob controls the current mode of the IZLID, operation modes include: Off, Low, High, and Pulse. Turn the Selection Knob to the appropriate mode.

To view the laser beam in VRSG turn on NVG mode. Press and hold the fire button to fire the laser and move the IZLID to point the laser at a target.

BSI MACE IZLID Control plugin

The BSI MACE IZLID Console plugin controls the configuration of the emulated IZLID. Use the IZLID Control plug-in to control which entity the IZLID is attached to, the entity offset in the VRSG view, and bore sighting the IZLID.



Site: 1 App: 2 Entity: 213

IZLID Offset

Vertical: 3 ft Horizontal: 2 ft

VRSG Settings

Set the Site:App:Entity to match the entity that the IZLID is attached to.

IZLID Offset

Use the vertical and horizontal sliders to adjust the laser beam origin relative to the attached entity. Hold the IZLID in the same position that you will operate it and adjust the sliders until the origin aligns with the view. This alignment should be checked for each new person who uses the IZLID.

Boresight

Position the IZLID in the direction of the VRSG view and click Boresight to align the forward direction of the beam relative to the VRSG view.

Reattach

Click Reattach if the IZLID has stopped responding or if you updated the Site:App:Entity in the VRSG settings to attach to a new entity.

Troubleshooting

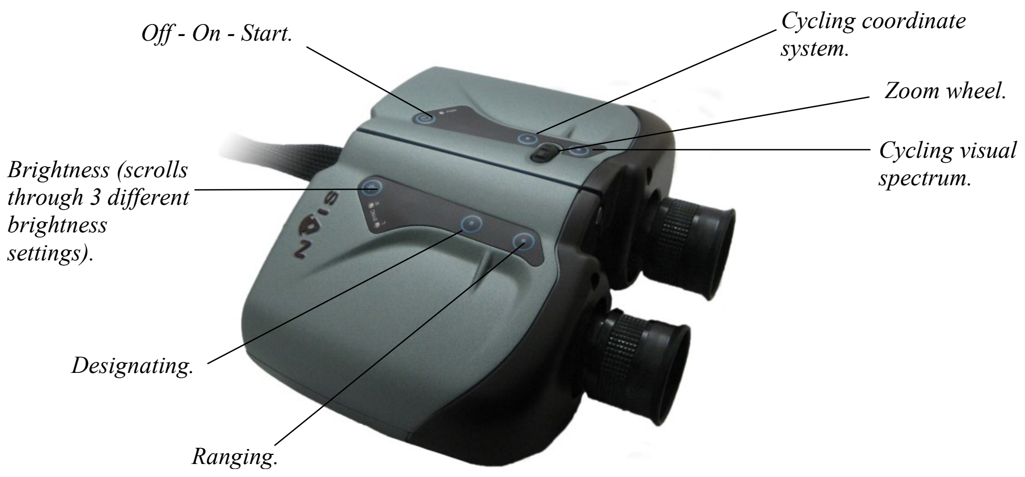
If the IZLID is not plugged in, or if the device fails to enumerate when it is plugged into the host computer, VRSG will display a "No IZLID attached" warning in the VRSG SOFLAM view. When an IZLID is plugged in, VRSG will briefly display the notification, "IZLID plugged in. Attaching." No notifications will be displayed once the IZLID is properly attached.

If the IZLID is plugged in but is failing to track, try simply "hot plugging" the USB cable by removing it from the host computer and plugging it back in. Sometimes the SOFLAM will enumerate properly and VRSG will properly detect and attach to the device, but it will fail to track. This issue is often caused by faulty USB cables or USB ports. Long unpowered cables (greater than 3 meters) can sometimes cause problems, as can low-cost USB extensions.

NVIS Ranger 47 notional emulated hardware

One notional emulated hardware device that interoperates with VRSG is the NVIS Ranger 47 virtual binoculars. This device is a hand-held display designed for handling a variety of training and simulation tasks. The Ranger 47 features dual SXGA OLED microdisplays with focus-adjustable eyepieces displaying a 47-degree diagonal field-of-view (36H x 28V). The device also includes a central hinge for interpupillary distance (IPD) adjustments. Stereopsis is supported via two independent video inputs.

In addition to power and brightness buttons, the Ranger 47 provides four programmable USB joystick compatible buttons, plus a z-axis scroll wheel.


Off - On - Start.
Cycling coordinate system.
Zoom wheel.
Cycling visual spectrum.
Brightness (scrolls through 3 different brightness settings).
Designating.
Ranging.

A small external mounting plate on the bottom of the NVIS Ranger 47 accommodates standard motion trackers, such as the Intersense InertiaCube4. Note that the InertiaCube4 is mounted upside down on the Ranger 47.

As shown in the image above, the buttons on this device is programmed to map to VRSG FO/FAC/JTAC functions such as ranging, designating, cycling the visual spectrum (Day TV, IR hot, IR black, and NVG), and cycling the coordinate system displayed for ranging and designating (Lat/Lon and MGRS).

The Ranger 47 plug-and-play device simply requires a USB connection plus a DVI connector for video. The device offers several programmable buttons for ranging and designating functions. VRSG uses the Microsoft Human Interface Device (HID) protocol via DirectInput to instrument the embedded tracker and map the buttons to the device's primary ranging and designating functions.

Before you start running VRSG with the Ranger 47, note that for older versions of NVIS Ranger 47, you must start the computer with the device turned off; otherwise the device will not work. (This is a known firmware issue with device). In addition, ensure the VRSG shortcut used to start VRSG does not contain a `–ignoreJoystick` or `–ignoreTracker` flag. The Ranger47 shows up as two devices on the VRSG Dashboard’s Startup Parameters tab, as an InertiaCube4 (IC4) tracker and a joystick.

1. Start VRSG, and on the Dashboard’s Startup Parameters tab ensure that VRSG detects the IC4 tracker and joystick. If VRSG does not detect the tracker, check that the isense.dll is in the proper place in the VRSG\Plugins folder.
2. If the joystick is not found, check USB devices in Window, to make sure it is among the list of controllers.
3. On the Dashboard’s Startup Parameters tab, select the Enable Sensor Mode option, before launching VRSG visualization, so that the Ranger 47 can cycle through the visual spectrum.
4. Press the Ranger 47’s Power button to turn on the display, essentially awakening it from sleep mode. (To turn it off, press that button again and hold.)
5. When VRSG detects the Ranger 47, it displays a message that asks whether you want to use the device. Click Yes.
6. Because the InertiaCube4 tracker is mounted upside down on the Ranger-47, you cannot boresight the normal way (as scene is displayed upside down), so instead of pressing the F7 key on the keyboard to boresight, press Shift F7 to reverse boresight.
7. Ensure the device’s buttons are operational; if they are not, open the control panel for the Ranger 47 to troubleshoot. If you cannot determine the problem, then this might be a Ranger 47 model that must be powered off when the computer is turned on.
8. Ensure the device zooms in and out properly with the zoom wheel. If the zoom wheel is not working, exit from VRSG and add the following entry to the FPS.ini file located in \MVRsimulation\VRSG:

   ```
   enable_zoom_wheel: 1
   ```

   (More about the FPS.ini file is described in the chapter “Using 3D Characters in VRSG.”)

9. Press X on the keyboard to activate VRSG's generic reticle display inside the Ranger 47. The tick marks are labeled with their angular offsets from the center in mils units.

When you range a target, the display will tell you the range from the last time you ranged by displaying the differential range and bearing.

D-RANGE: shows the relative range to last lased position, in meters.

D-BRNG: shows the relative bearing to last lased position, in degrees.

While pressing the Designate button, you can set the 4-digit PRF code for laser guidance on the numeric keypad. The PRF code will display for a few seconds. The next section describes the button mapping to VRSG ranging/designating functions.

Customizing ranging/designating function mapping to buttons on simulated military equipment

You can customize the mapping of buttons on simulated military devices for used for laser ranging and target designation training to their functions in the FPS.ini file. This ASCII file is located in \MVRsimulation\VRSG.

For example, you can add the following lines of text to the FPI.ini, which contain VRSG's default button assignments for simulated military devices:

```
designate_ button: 1
range_button:      2
coord_button:      3  ! toggles the coordinate system (such as
                        MGRS, or Geodetic)
spectrum_ button:  4  ! cycles through the sensor views (Visual, EO,
                        IR, and NVG)
```

Press F7 on the keyboard for boresight. Press Shift-F7 instead to reverse boresight if the unit has a reverse mounted tracker, as in the NVIS Ranger 47 simulated military device. (A reverse mount will be obvious if the pitch seems to be backward.)

Although adding the above entry of default button assignments is not necessary when you use one device (unless you want to change the button-function mapping), for situations in which you use multiple devices, you can assign buttons from different devices to the same FPS function. Doing so helps when you need to switch between devices (for example, switching between the Battlespace Simulations' SOFLAM and NVIS Ranger 47 devices).

To implement the multi-device button mapping, hook up your simulated military device, open the device's "joystick" properties applet and make note of which physical buttons on the device correspond to which "joystick" button numbers. Then, in the VRSG FPS.ini file, enter a comma-separated list of button numbers to specify which functions they activate:

```
designate_button: <which buttons designate>

range_button: <which buttons range>

coord_button: <which buttons toggle the coordinate system display
              (UTM vs geodetic)>

spectrum_button: <which button cycles sensor modes (visual, IR, or
                 NVG)>
```

For example, for the SOFLAM emulated device, you would enter the following values:

```
range_button: 9
range_button2: 2
```

```
designate_button: 9
designate_button2: 3
```

These values direct the unit to:

- Range lase when button 2 is pressed (knob on Range position) *and* button 9 (Fire) is pressed.
- Designate when button 3 is pressed (knob on Mark position) *and* button 9 (Fire) is pressed.

For more information about the FPS.ini file, see the chapter “Using 3D Characters in VRSG.”

CHAPTER 18

Distortion Correction and Edge Blending with VRSG

Computer generated imagery is typically generated by projecting 3D objects onto a planar focal plane. Some simulators use display surfaces that are not flat planar surfaces, such as the curved surfaces of hemispherical domes or cylindrical displays. When computer-generated imagery is projected into a non-planar display surface, or the display surface perpendicular is not aligned with the direction of projection, a distortion of the projected image results. A process known as *distortion correction* is used to solve this problem.

A separate problem arises when multiple projectors are used on a single common display surface. Areas of the display surface that are illuminated by more than 1 projector will be excessively bright. The process of correcting for these overlap areas so that an image of common intensity is achieved is referred to as *edge blending*. A solution that handles both distortion correction and edge blending is often referred to as a *warp/blend system*.

Some projectors have built-in distortion correction and edge-blending capabilities; others do not. VRSG is delivered with an MVRsimulation plugin for making simple manual adjustments to correct projected imagery on a curved surface. Another MVRsimulation plugin provides horizontal distortion correction for curved monitors. VRSG also provides an MVRsimulation plugin for projectors that require a pixel shift for emulated 4K resolution.

Also delivered are plugins for using VRSG with the third-party warp/blend systems of Scalable Display Technologies, VIOSO Projection Software and Dome Projection software on a curved surface with a projection system.

All these plugins are installed in \MVRsimulation\VRSG\Plugins\Displays.

The plugin solutions described in this chapter are software based, where the video output from VRSG is modified before the display device to accomplish the warp/blend. This method is in contrast to projectors or other display devices that handle the warp/blend process downstream from VRSG's video output. Such displays are beyond the scope of this chapter.

For information about trade-offs in image projection FOVs and performance in multi-channel systems, see the section “Projection considerations for multi-channel configurations” in the chapter “Running VRSG With Synchronized Channels.”

Distributing plugins in a shared environment

If your site has a VRSG directory structure shared across multiple computers, but not all computers need a particular plugin (plugins distributed with VRSG like the ones described in this chapter or plugins your site-specific plugins), you can make unique \Plugins directories by renaming the \MVRsimulation\VRSG\Plugins\ directory, to \Plugins_*hostname* where *_hostname* is the target computer for the plugin. VRSG will check for any \Plugins_*hostname* directories before it checks the \Plugins directory. This way, in the shared directory environment, different VRSG channels can load different plugins, or no plugins at all.

Using MVRsimulation's Warp plugin

You can accomplish simple warp/blend tasks using MVRsimulation's Warp plugin. To activate the Warp plugin for VRSG, copy or move the file \MVRsimulation\VRSG\Plugins\Displays\Warp3.dll up a directory level so that it resides in \MVRsimulation \VRSG\Plugins. The next time you start VRSG, the Warp tab will be displayed on the VRSG Dashboard as shown on the next page. MVRsimulation recommends performing the warp/blend process with VRSG in desktop cover display mode, so that you can access the VRSG Dashboard without needing to toggle back and forth between fullscreen and windowed modes.

Geometry correction

The warping process involves creating a set of control points and moving those points individually until the desired geometry correction is achieved. You can increase the control point density as needed to fine tune specific areas. To begin the warping process, click the Enable Warping and the Show Control Points checkboxes on the Warp tab.

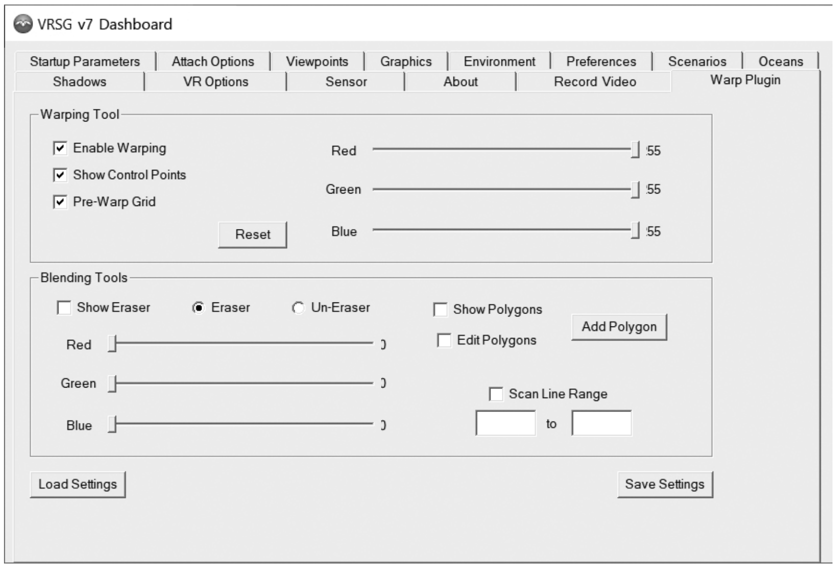
The control points will be displayed as semi-transparent purple boxes, initially one at each of the 4 corners of the display. When you click the left mouse button over a control point, the control point will be highlighted. You can now move the control point to adjust the warping of the image.

As an alternative to using the mouse to move control points, you can press the arrow keys to move the cursor. Use the arrow keys to adjust the cursor so that it is over a control point. Press F6 to toggle the control point to a highlighted state. While highlighting the control point, use the arrow keys to adjust the position of the control point.

You can insert additional control points as needed to refine the warping process. Fewer control points require less effort, but more control points offer a higher fidelity warping result. To insert a new control point into the scene, use the Shift-P keyboard command. Doing so inserts a new control point under the current cursor position.

Remove control points with the Shift-U keyboard command. Control points are removed in the reverse order in which they were added.

Pressing the Reset button on the Warp tab removes all control points beyond the initial 4, and will reset the initial 4 control points to their initial positions.



Warping Tool

- [✓] Enable Warping
- [✓] Show Control Points
- [✓] Pre-Warp Grid

Red	55
Green	55
Blue	55

Blending Tools

- [ ] Show Eraser
- (●) Eraser
- ( ) Un-Eraser
- [ ] Show Polygons
- [ ] Edit Polygons
- [ ] Scan Line Range ____ to ____

Red, Green, Blue

To save your results, press the Save Settings button. It is a good practice to save your settings frequently, so that you have a sound restored point to return to in the event of error.

An optional pre-warp grid may be displayed which shows the effects of the warp. This pre-warp grid consists of vertical and horizontal lines drawn in screen-space before the warping takes place.

You can use the following keyboard shortcuts instead of accessing the controls on the Warp Plugin tab during the warp process:

- Shift-P – inserts a new control point under the current cursor position.
- Shift-U – removes the last inserted control point.
- F6 – toggles selection of the control point under the cursor.
- Arrow keys – move the cursor a single pixel in any direction. If a waypoint is selected, the waypoint will be moved with the cursor.
- G – increases density of the pre-warp grid.
- Shift-G – decreases density of the pre-warp grid, ultimately turning it off.

The Warping Tool section of the Warp Plugin tab contains the Red, Green, and Blue sliders, which are global scene intensity attenuators. These sliders can be useful for changing the intensity of the entire scene to compensate for projector differences or the perceived gain of the display surface.

Edge blending / masking

You use the eraser tool to graphically mask out sections of the image. Think of the eraser tool as a “paintbrush” application that creates an attenuation raster to modulate the input image from VRSG. You have control over each pixel in the raster and how much red, green, and blue that pixel is allowed to show through.

You use the mouse wheel to control the size of the eraser. The eraser size can be as small as 1x1 pixels, or as large as needed. The uneraser tool restores pixels under its footprint. While moving the eraser with the mouse, press the left mouse button to apply the eraser's current footprint to the image. The eraser cursor is rendered darker to indicate the mouse is depressed and the eraser is active. For more precise cursor movement, you can use the arrow keys on the keyboard to move the cursor a single pixel at a time. The F6 button can be used to activate the cursor, similar to pressing the left mouse button.

By default, the eraser replaces pixels in the image with pure black (0,0,0). Drag the red, green, and blue sliders to a non-zero position to cause the eraser to attenuate intensity. The position of these sliders sets the percentage of red, green, and blue that is allowed to pass through the eraser. This semi-transparent erasing is useful to create blend regions where two projectors overlap.

You can use the following keyboard shortcuts instead of the controls on the Warp Plugin tab to control eraser behavior:

- Arrow keys – move the eraser a single pixel in any direction.
- Numpad up – increases the size of the eraser.
- Numpad down – decreases the size of the eraser.
- Numpad right – increases the opacity of the eraser (increments R, G, and B).
- Numpad down – decreases the opacity of the eraser (decrements R, G, and B).
- F5 – toggles between eraser and uneraser modes.
- F6 – toggles active cursor (same as pressing the left-mouse button).
- Mouse wheel – changes the size of the eraser cursor.

Mask polygons

In addition to the raster-based eraser tool, you can create and edit 2D mask polygons. To create a polygon, click the Show Polygons checkbox then press the Add Polygon button. VRSG will prompt you to enter at least 3 vertices for your polygon. Currently the following limitations apply to mask polygons:

- The vertices must be entered in counter-clockwise order.
- The polygon must be convex.

You can add vertices to the polygon by moving the mouse cursor to the intended location and pressing the left mouse button. For more accurate placement, press the arrow keys to move the cursor. You can also press F6 to insert a vertex at the current cursor location.

After you have entered the vertices of your polygon, click End Polygon to complete the polygon. (You can also press the Esc key to complete the polygon.)

Click the Show Polygons checkbox to activate the polygons and put their masking behavior into effect. You can enable the editing of polygons after they have been entered by selecting the Edit Polygons checkbox. In polygon edit mode, you can adjust the vertices of polygons, modify their masking intensity, or delete polygons.

In edit polygon mode, the polygons are displayed with control points at the vertices and purple lines along each edge. By moving the mouse over a vertex control point, you can drag the vertex to a new position. With the vertex highlighted, you can also use the arrow keys to move the vertex in any direction a single-pixel at a time.

To modify the masking intensity of a polygon, select the polygon by left-clicking inside the polygon's boundary. The selected polygon is indicated by white borders. Once a polygon has been selected, you can adjust its masking intensity by dragging the red, green, and blue sliders on the Warp Plugin tab.

To delete a polygon, first select the polygon to be deleted by clicking the left-mouse button inside the polygon's boundary. Once you have selected the polygon, press the D key on the keyboard to delete the polygon.

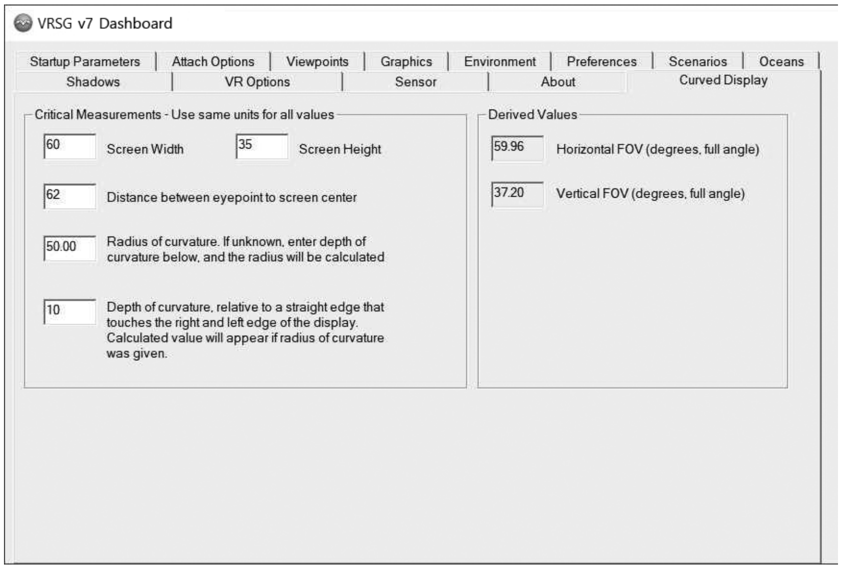
You can use the following keyboard shortcuts instead of the controls on the Warp Plugin tab to control masking polygons:

- Arrow keys – move the eraser a single pixel in any direction.
- F6 – inserts a vertex into the mask polygon at the current cursor location.
- Esc – completes the polygon when you have no more vertices to add.
- D – deletes the selected polygon.

Using MVRsimulation’s Curved Display plugin for monitors

The Curved Display plugin provides horizontal distortion correction for a single curved monitor. Because the display surface is not flat, the signal sent to the monitor must account for how the image projects onto the non-flat surface.

To activate the plugin, copy or move the file \MVRsimulation\VRSG\Plugins\Displays\ CurvedDisplay.dll up a directory level so that it resides in \MVRsimulation\VRSG\Plugins. The next time you start VRSG, the Curved Display tab will be displayed on the VRSG Dashboard as shown:



Critical Measurements - Use same units for all values

60 Screen Width

35 Screen Height

62 Distance between eyepoint to screen center

50.00 Radius of curvature. If unknown, enter depth of curvature below, and the radius will be calculated

10 Depth of curvature, relative to a straight edge that touches the right and left edge of the display. Calculated value will appear if radius of curvature was given.

Derived Values

59.96 Horizontal FOV (degrees, full angle)

37.20 Vertical FOV (degrees, full angle)

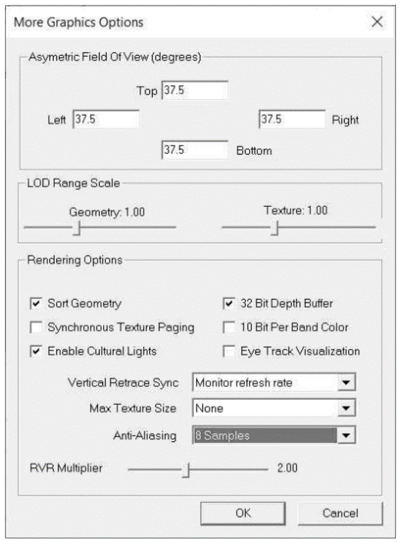
The derived FOV values on the right are calculated from values you enter on the left. You can enter measurements in any unit of measure, as long as they are of the same units. Enter either the radius of curvature or the depth of the curvature; the other measurement value will be calculated.

Using MVRsimulation's Pixel Shift plugin for emulated 4K resolution

VRSG includes a new Pixel Shift plugin, which supports 4K projectors known as *Faux-K* projectors that require a pixel shift for emulated 4K resolution. Faux-K projectors generally require two video inputs, each at 2560x1600 resolution and offset diagonally by a half pixel. The projector electronics combine these two images into a single 4k video output; use the Nvidia Surround feature to connect these two displays into a single 5120x1600 desktop.

To activate the plugin, copy or move the file \MVRsimulation\VRSG\Plugins\Displays\PixelShift.dll up a directory level so that it resides in \MVRsimulation\VRSG\Plugins. Unlike other VRSG plugins, Pixel Shift does not add a tab to the Dashboard; no user inputs are needed. However, the plugin requires an anti-aliasing level of 8.

Change the antialiasing level in the More Graphics Options dialog box, which you access from the Dashboard's Graphics tab.



More Graphics Options

Asymetric Field Of View (degrees)

Top 37.5

Left 37.5

37.5 Right

37.5 Bottom

LOD Range Scale

Geometry: 1.00

Texture: 1.00

Rendering Options

[✓] Sort Geometry

[✓] 32 Bit Depth Buffer

[ ] Synchronous Texture Paging

[ ] 10 Bit Per Band Color

[✓] Enable Cultural Lights

[ ] Eye Track Visualization

Vertical Retrace Sync: Monitor refresh rate

Max Texture Size: None

Anti-Aliasing: 8 Samples

RVR Multiplier 2.00

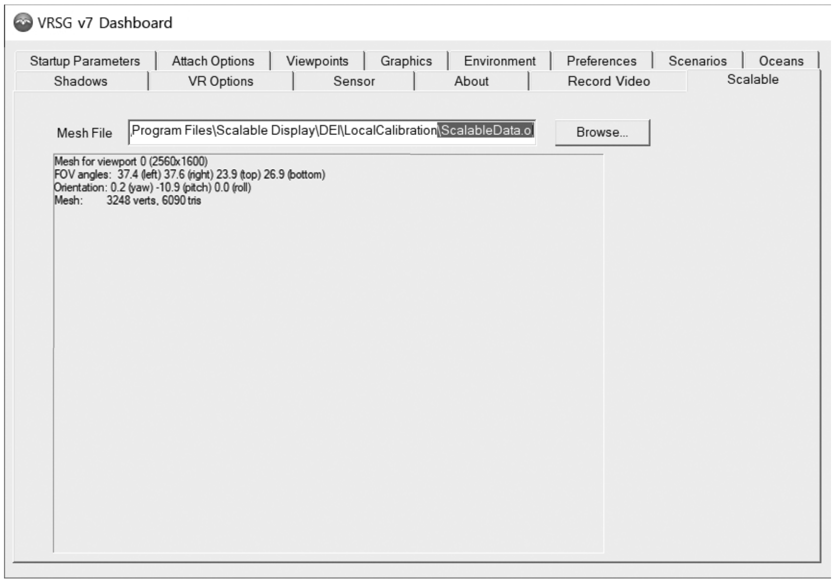
With the Pixel Shift plugin in place, VRSG will render your scene twice. The left half of the framebuffer will contain a 2560x1600 image of the requested scene. The right half of the framebuffer will contain a similar scene, but offset diagonally by a half pixel. Your pixel shift projector will combine these two images into a single 4K scene for final display.

Configuring the Scalable Display Technologies plugin

MVRsimulation provides a plugin for VRSG for use with the Scalable Display Technologies warp/blend system. See the Scalable Display documentation for instructions on calibrating your screens with the Scalable Display software. Once you have verified your calibration using Scalable Display's test patterns, you are ready to test the calibration with VRSG.

The Scalable plugin supports multiple projectors per computer when it is used in conjunction with NVidia's Surround or Mosaic modes. The plugin also supports changes in calibration files produced by Scalable V8.

To activate the Scalable Display plugin to use with VRSG, copy or move the file \MVRsimulation\VRSG\Plugins\Displays\Scalable.dll to \MVRsimulation\VRSG\Plugins. When you restart VRSG, the Scalable tab will be displayed on the VRSG Dashboard as shown below:



Mesh File: ,Program Files\Scalable Display\DEI\LocalCalibration\ScalableData.ol

```
Mesh for viewport 0 (2560x1600)
FOV angles:  37.4 (left) 37.6 (right) 23.9 (top) 26.9 (bottom)
Orientation: 0.2 (yaw) -10.9 (pitch) 0.0 (roll)
Mesh:        3248 verts, 6090 tris
```

When you perform a Scalable Display calibration, a data file is produced on each computer as a result of the calibration process. This data file is located in the following path:

C:\Program Files\Scalable Display\DEI\LastCalibration\ScalableData.ol

If your Scalable Display calibration did not produce a file named ScalableData.ol, then you selected the wrong output format for the calibration. See the Scalable Display documentation for instructions on producing a mesh file for use in simulators. Mesh files can be changed dynamically via a SetDataPDU interface.

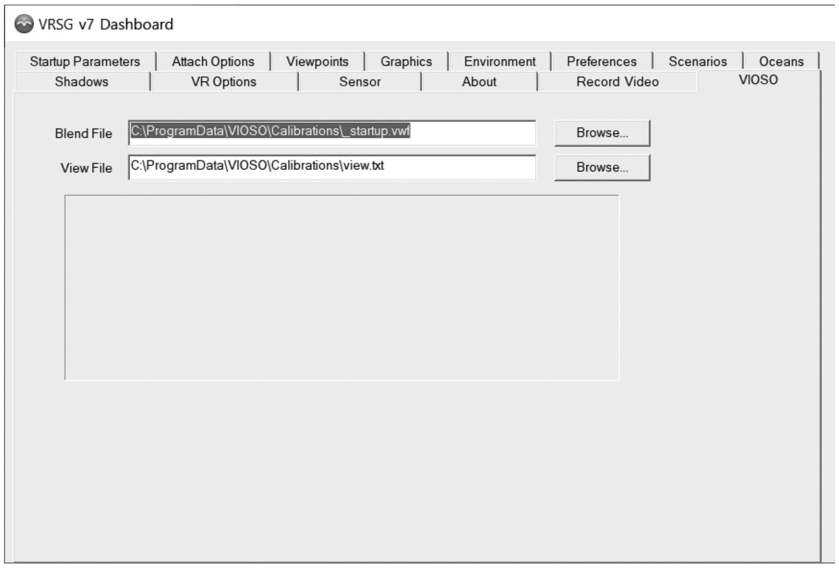
VRSG's Scalable Display plugin has this mesh file name as its default. Under normal circumstances, you should not need to change this value.

If the mesh file was successfully deposited on the computer, you will see information about the calibration displayed on the tab, as shown above. If the mesh file did not successfully load, an error message will be displayed instead.

If each channel has successfully loaded the mesh file, you are ready to start VRSG to verify the results of the calibration.

Configuring the VIOSO Projection software plugin

VRSG includes a plugin to support VIOSO Projection software's warp/blend system. To install this plugin, copy VIOSO.dll from VRSG\Plugins\Displays up one level into \VRSG\Plugins. When you restart VRSG, the VIOSO tab will appear on the Dashboard as shown:



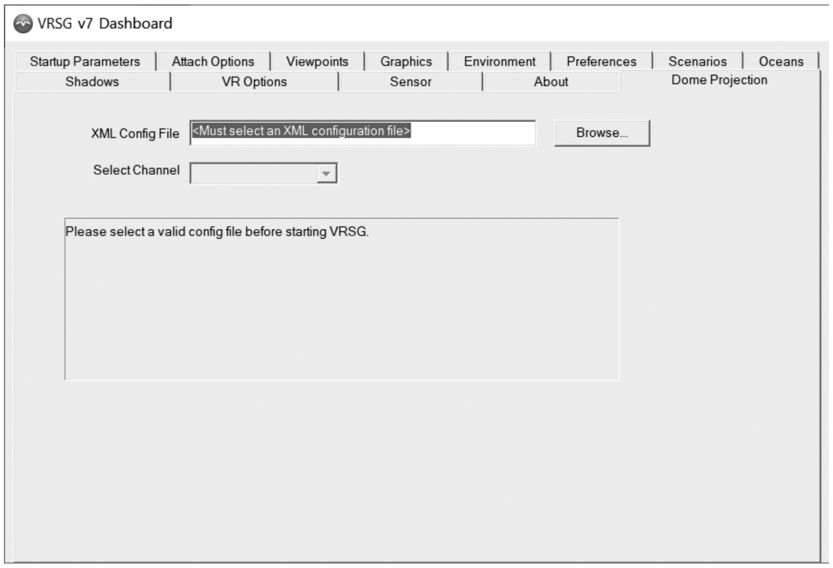
Blend File: C:\ProgramData\VIOSO\Calibrations_startup.vwf

View File: C:\ProgramData\VIOSO\Calibrations\view.txt

When you calibrate your display with VIOSO, two files are added to the directory C:\ProgramData\VIOSO\Calibrations. The file with the extension of '.vwf' contains the warp/blend raster image data, and the view.txt file contains data about the orientation and field-of-view of the channel. The VRSG VIOSO plugin needs both files to accomplish the warp/blend. Click the Browse button to select the appropriate files if the defaults are not correct for your system. If no errors appear in the text box below the filenames, you are ready to start VRSG and test the warp/blend.

Configuring the Dome Projection plugin

VRSG includes a plugin to support Dome Projection's warp/blend system. To install this plugin, copy DomeProjection.dll from VRSG\Plugins\Displays up one level into \VRSG\Plugins. When you restart VRSG, the Dome Projection tab will appear on the Dashboard as shown:



XML Config File <Must select an XML configuration file>

Select Channel

Please select a valid config file before starting VRSG.

When you calibrate your display with Dome Projection, several output files are produced. One of the files produced has the extension .xml. This xml file is the one you should enter into the XML Config File field of the Dome Projection tab.

Click the Browse button to select the appropriate .xml file. After selecting the xml file, the Select Channel list will be activated, containing a list of channels provided by the calibration. Select the channel from this list that corresponds to the VRSG instance.

Several other files are produced as a result of the calibration. These files have the extensions .csv and .dds. These files must be also be present for the plugin to function. The xml file you selected references these external files.

APPENDIX A

VRSG Keyboard Functions

This appendix lists the keyboard functions available in the VRSG virtual world.

Displaying information

Action	Key
Toggle the display of the VRSG Dashboard.	Esc
Display terrain and eyepoint elevation (AGL, MSL, and HAE) at the current position of the eyepoint.	E
Display a military grid reference system (MGRS) grid as a 2D overlay.	G
Display the frame rate statistics of the current scene.	H
Perform laser range to polygon, similar to left-click. Records the coordinates to the waypoints.dat file.	L
Display the name of the model and texture face under crosshairs.	Shift L
Display current scene statistics: the number of triangles, the number of active models, and the frame rate.	Q
Toggle the display of the compass rose.	R
Display system memory and texture memory statistics.	T
Cycle the coordinate system displayed in the UAV/Sensor Overlay: UTM, MGRS, and LAT/LONG.	U
Display the keyboard Help.	F1
Displays the ID of the tile under the cursor.	Shift Click
Display the IDs of entities as a 2D overlay.	F12
Display the IDs of static cultural features as a 2D overlay.	Ctrl F12

Controlling the display, and other actions

Action	Key
Take a screen capture of the current view, using the output type and format settings on the Dashboard Preferences tab.	C
Detach from a network entity or cultural feature to which you are attached.	D
Toggle the ability to move/drag the static feature you are attached to.	Shift D
Toggle detaching/attaching from/to viewpoints in a scenario PDU log.	Ctrl D
Group and ungroup an aggregation of models.	Ctrl G
Toggle displaying the scene in wire-frame mode.	I
Save an attached feature's location and orientation to a cultural feature file.	J
Cycle the dynamic entity scale factor between 1X, 2X, 3X, and 4X.	M
Attach to the entity or cultural feature that is located under the cursor.	Middle mouse button
Reload ModelMap.ini (after editing the file during a VRSG session).	Ctrl M
Attach to the next entity in the Entity list.	N
Cycle through the visual spectrum when the Enable Sensor Mode checkbox is selected on the Dashboard's Startup Parameters tab.	O
Pause/resume scenario playback.	Ctrl P
Save a viewpoint. (To name it, use the Dashboard's Viewpoint tab.)	S
Display crosshairs.	X
Delete the entity or cultural feature to which you are attached.	Shift X
Change the wind direction.	W

HMD, mixed-reality and other scenario controls

Action	**Key**
Cycle VRSG Desktop Cover and Sizeable Window modes.	F2
Toggle between Varjo and desktop mode (when the Varjo plugin is installed in the \Plugins folder).	F5
Boresight the forward orientation when using an HMD.	F7
Cycle which Vive tracker is currently active.	F8
Turn on and off First Person Simulator (FPS) mode.	F9
Turn on and off Fixed Wing Simulator mode. If the Varjo plugin is installed in the \Plugins folder, turn on and off use of depth range.	F11
Turn on and off display of 3D oceans.	~ (tilde)
Increase the playback speed of a scenario10x.	+ (plus sign on keypad; Shift + on keyboard)
Decrease the playback speed of a scenario 10x.	- (minus sign on keypad)

Navigation

Action	**Key**
Move the eyepoint left.	Left
Rotate the eyepoint left.	Shift Left
Move the eyepoint right.	Right
Rotate the eyepoint right.	Shift Right
Move eyepoint forward.	Up
Rotate the eyepoint upward.	Shift Up
Move the eyepoint backward.	Down
Rotate the eyepoint downward.	Shift Down

Action	**Key**
Move to the next saved viewpoint (saved previously with "S" or on the Dashboard's Viewpoints tab).	V
Toggle Nudge mode to slow down the gain on the controller.	Y

APPENDIX B

Utilities Delivered With VRSG

VRSG is delivered with several utilities which can manipulate MVRsimulation-specific content, including 3D terrain and model files.

The primary utility is MVRsimulation's Model Viewer (Modelviewer.exe), which is installed in the directory \MVRsimulation\Common\Bin. You can use Model Viewer to preview content in many MVRsimulation-specific file formats, including models, VRSG terrain tiles, textures, particle effects, billboard effects, and clouds. For a full description of Model Viewer's capabilities, see the chapter "Previewing Models, Effects, and Terrain."

In addition to Model Viewer, a number of other utilities—both MVRsimulation-created and third-party tools—are installed in the directory \MVRsimulation\Common\Util. (Some utilities have their own subdirectories.) You can use these utilities to create or edit various types of files involved in the construction and visualization of 3D terrain and models for use in VRSG. The purpose of each utility is described in the table below.

Filename	**Utility description**	**For more info…**
7z.dll 7z.exe	7-zip is a free and open source file archiver. MVRsimulation product installation files and 3D content libraries are delivered using the .7z compressed archive file format. MVRsimulation's installation programs use 7z.dll and 7z.exe to extract this data during product installation.	See www.7-zip.org.
Fbx2Hpx.exe	Converts models from Autodesk's FBX format to MVRsimulation's HPX model format.	See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."
haspdinst.exe	Sentinel hasp dongle device driver installed automatically during MVRsimulation product installation. (Under normal circumstances, you should not have to access the dongle device driver software.)	See the *MVRsimulation Product Installation Guide*.

Filename	Utility description	For more info…
IRSetup.exe	Accepts input for a sensor profile: sensor's spectral response within its waveband of interest, material classifier, and environmental characteristics that influence the appearance of an IR scene.	See the chapter "Working with Sensor View Modes and the Physics-Based IR."
MVRThumb.dll	System extension dll that enables Windows Explorer to display thumbnail images for MVRsimulation-specific file formats, including HPY, MDS, and TEX.	N/A
oflt2hpx.exe	Converts models from OpenFlight format (FLT) to MVRsimulation's HPX model format.	See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."
PlayBack.exe	Logs and plays back DIS network exercises. During a simulation, use the Playback utility to record all DIS network traffic (PDUs) to a log file. Then use Playback to replay the simulation by re-broadcasting the recorded network traffic from the log file.	See the chapter "Exploring the VRSG System."
terrex-oflt2mds.exe	Converts terrain databases from OpenFlight format (FLT) to MVRsimulation's VRSG terrain tile format (MDS).	See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."
TileMgr.exe	Identifies the names of one or more VRSG terrain tiles (MDS files) which cover a given coordinate or geographic area.	See the chapter "Previewing Models, Effects, and Terrain."

APPENDIX C

VRSG Registry Variables

This appendix lists the registry variables available in VRSG that you can add or edit for fine-tuning your virtual world. These variables are listed for your convenience; they control settings that cannot be controlled through the Dashboard user interface. They are stored in the registry in HKEY_CURRENT_USER\Software\MVRsimulation Inc.\MVRsimulation Vrsg\Parameters. This appendix assumes that you are comfortable with modifying the Windows registry, know how to back up any valuable data beforehand, and are prepared to restore the registry in the event that an error occurs in your registry and your computer does not function properly.

Registry variable	**Purpose**	**Values**
BumperNumberRange (DWORD)	Range at which when exceeded, VRSG will no longer display bumper numbers on vehicles.	A range in meters
cigiRecvPort (DWORD)	Overrides the UDP port on which VRSG receives CIGI messages.	Default is UDP port 8021
cigiSendPort (DWORD)	Overrides the UDP port on which VRSG sends CIGI messages.	Default is UDP port 8022
CsAddr (STRING)	Address to which VRSG transmits MultiChannel messages to host.	A valid IP address in ASCII decimal dot notation
CsRecvPort (DWORD)	Port on which VRSG reads MultiChannel messages. Overrides default of 5069.	A valid port number
CsSendPort (DWORD)	Port on which VRSG transmits multichannel messages. Overrides default	A valid port number
EnableF12 (DWORD)	If zero, VRSG will not respond to the F12 key.	1 = VRSG responds to F12 key; 0 = VRSG ignores F12 key
Entity (DWORD)	Set the entity component of VRSG's ID. This value overrides that which is stored in the settings file.	0..65536

Registry variable	**Purpose**	**Values**
ForceId (DWORD)	Value to place in the forceID field of an EntityStatePDU.	Valid DIS forceID code
Host (DWORD)	Set the host component of VRSG's ID. This value overrides that which is stored in the settings file.	0..65536
IgnoreEntity (DWORD)	Specifies the entity component of an entity VRSG should ignore EntityStatePDUs from.	0 .. 65536
IgnoreHost (DWORD)	Specifies the host component of an entity VRSG should ignore EntityStatePDUs from.	0 .. 65536
IgnoreSite (DWORD)	Specifies site component of entity VRSG should ignore EntityStatePDUs from.	0 .. 65536
IP_MULTICAST_TTL (DWORD)	Sets the multicast time-to-live parameter of multicast sockets.	Unconstrained
LaserCode (DWORD)	Value to place in the laserCode field of a DesignatorPDU.	Unconstrained
LaserCodeName (DWORD)	Value to place in the laserCodeName field of a DesignatorPDU.	Unconstrained
LaserPower (DWORD)	Value to place in the laserPower field of a DesignatorPDU.	Unconstrained
LaserWavelength (STRING)	Value to place in the wavelength field of a DesignatorPDU.	ASCII floating point number
logPort (DWORD)	Instructs VRSG to record CIGI messages from host to channel to a log file port_8021_*N*.pdulog, where *N* is an integer that is incremented for every run.	Recording port 8021 (decimal)
MaxHorizon (DWORD)	Sets the maximum value the Far Clip slider will allow.	Units are in meters
MaxNearClip (DWORD)	Sets the maximum value the Near Clip slider will allow.	Units are in decimeters
MaxSramTextures (DWORD)	Indicates the maximum number of textures VRSG will cache in system memory.	Example: 1000

Registry variable	**Purpose**	**Values**
MaxUavFov (STRING)	Sets the maximum UAV field-of-view for VRSG's internal UAV camera payload model.	ASCII floating point number, e.g. 45.0
MaxVramTextures (DWORD)	Indicated the maximum number of swappable textures VRSG will store in video memory.	Example: 500
MinNearClip (DWORD)	Sets the minimum value the Near Clip slider will allow.	Units are in decimeters
MinUavFov (STRING)	Sets the minimum UAV field-of-view for VRSG's internal UAV camera payload model.	ASCII floating point number, such as 2.0
MulticastAddress [0..N] (STRING)	Indicates a valid multicast address VRSG should subscribe to.	ASCII decimal dot notation
OverlayTextHeight (DWORD)	Sets height of overlay text. Use to increase size of overlay text.	Default =15
OverlayTextWidth (DWORD)	Sets width of overlay text. Use to increase size of overlay text.	Default =10
particlePixelScale Percent (DWORD)	Scales scale the min-pixel-size of particle effects	Percentage; such as 150 for a 150% size increase
relDiDevice (DWORD)	Enables 3DConnexion SpaceMouse Pro wireless mouse to work with VRSG	1= enables the mouse
ReverseImage (DWORD)	Horizontally reverses the viewport of the channel.	0 = display image reversed 1 = display image normal
ShadowExtraOverlays (DWORD)	Instructs VRSG to generate overlays normally supplied by TUAV ground station when displaying SHADOW/TUAV HUD.	0 = no GCS overlays 1 = add GCS overlays
Site (DWORD)	Sets the site component of VRSG's ID. This value overrides that which is stored in the settings file.	0..65536
skipPowerConfigCPU (DWORD)	Instructs VRSG to not change Windows power plan settings to High Performance to make sure the display never sleeps.	1 = prevents switching to sleep mode

Registry variable	**Purpose**	**Values**
skipPowerConfigGPU (DWORD)	Instructs VRSG to not change Windows power plan settings to High Performance to make sure the display never sleeps.	1 = prevents switch to sleep mode
SOL_RCVBUF (DWORD)	Sets the receive buffer size of UDP sockets.	10240 ... 1 MB
starSizeScalePercent (DWORD)	Scales the size of the stars in the night sky.	Percentage; such as 50 for a 50% size decrease
WindowHeight (DWORD)	Sets height of the VRSG visualization window. Overrides the default window height dimension to specify exact dimensions for H.264 video output.	Default = 480
WindowWidth (DWORD)	Sets width of the VRSG visualization window. Overrides the default window width dimension to specify exact dimensions for H.264 video output.	Default = 640

APPENDIX D

Read API for VRSG Terrain Tiles

MVRsimulation provides a free Read Application Programmer's Interface (API) for VRSG round-earth formatted terrain tiles to support host-side mission functions. The API library is available for both Windows and Linux 64-bit systems.

- Windows 64-bit applications should link to MdsAPI.lib; MdsAPI.dll is required for the runtime.
- Linux applications should link to libMdsRead_x64.a. The Linux library was built with gcc version 2.96 20000731 (Red Hat Linux 7.3 2.96-110).
- MVRsimulation customers on active maintenance can download the latest MVRsimulation's Read API for VRSG terrain tiles from appropriate Windows or Linux subdirectories in the /Software/MdsAPI section of MVRsimulation's Download Server. Customers on active maintenance who need an account on MVRsimulation's Download Server can request an account by sending a request to downloads@MVRsimulation.com.

Using the API library with multi-threaded applications

For the MdsAPI library to work with multi-threaded applications, ensure the setup calls are executed in a single thread, as follows:

1. Before any multi-threaded access, call mdsInitialize(), mdsAddDirectoryToSearchPath() to initialize the API.
2. After initialization, use as many threads as needed to call mdsGetElevation().

Note: The first time a terrain tile is loaded, it might take some time for the API to fetch the tile from disk and unpack the data. The thread that invokes loading the tile will block other threads from accessing the API while the shared data structures are being updated.

APPENDIX E

VRSG SetDataPDU Interfaces

VRSG offers an alternate interface to control various settings using DIS SetDataPDUs. This provides an alternate interface to using the VRSG Dashboard user interface directly. VRSG listens to the DIS port for SetDataPDUs even before it is launched into visualization mode, so you have an opportunity to change VRSG's settings before it is started.

VRSG will process SetDataPDUs that are directly addressed to it (that is, the receiver field matches VRSG's ID), or SetDataPDUs that are using a broadcast ID. VRSG considers the following receiver IDs to be broadcast:

	Site	**Application**	**Entity**
Broadcast address	0	0	0
Alternate broadcast address	65535	65535	65535

VRSG processes a set of fixed-length datums and variable-length datums. The fixed-length datums are interpreted as 32-bit integers, unless stated otherwise in the table below. Some fixed-length datums are used to convey 32-bit IEEE floating point numbers.

Variable length datums are multiples of 64-bits in length and are defined by various C structures described herein. As per the DIS protocol, all datums, both fixed and variable are expected to be in network (big-endian) byte order.

MVRsimulation's SetDataAttributes.h header file contains all the SetDataPDU interfaces. You can download this header file to include in your code from MVRsimulation's Download Server, in the /Software/Interfaces directory. Customers on active maintenance who need an account on MVRsimulation's Download Server can request an account by sending a request to downloads@mvrsimulation.com.

```
// MVRsimulation, Inc. SetDataPDU Interfaces for MVRsimulation VRSG
// Copyright (C) 1997 - 2023 MVRsimulation Inc. All rights reserved.
// URL: https://www.mvrsimulation.com Email: support@mvrsimulation.com
// Unless mvrsimulation agrees in writing, this API can be used only in connection
// with systems using MVRsimulation's Virtual Reality Scene Generator (VRSG).
// The use and availability of the API is at all times subject to the provisions
// governing "Additional Materials" under the MVRsimulation, Inc. license agreement
// found at www.mvrsimulation.com/howtobuy/license_agreement_policy.html.
```

APPENDIX F

CIGI Support

VRSG supports Common Image Generator Interface (CIGI) version 3.3 and 4.0 for communication between an image generator and host device in simulations.

VRSG implements a subset of CIGI to support the most commonly used features MVRsimulation has identified as needed by VRSG users. This appendix describes VRSG's CIGI support. See www.sourceforge.net for information about CIGI itself including samples and documentation.

New for this release of VRSG 7 are the following component controls:

- CIGI_SymbolSurfaceEntityAttached, which allows a symbol surface to be able to be attached to a DIS entity.
- CIGI_ComponentIdDepthOfField to control the depth-of-field sensor effect feature.
- CIGI_ComponentIdOwnshipForViewport, which enables each viewport to have a unique ownship entity.
- CIGI symbol rendering improvements including multi-line text support and improved mitering of line edges.

For detailed descriptions of MVR-specific extensions to CIGI, please refer to our sample CIGI hosts and CIGI headers located on the MVRsimulation Download Server in the /Software/Interfaces directory. Customers on active maintenance who need an account on MVRsimulation's Download Server can request an account by sending an email to downloads@mvrsimulation.com.

If you require additional functionality from CIGI than what is described in this appendix, contact support@mvrsimulation.com. MVRsimulation continuously enhances its CIGI support to meet the needs of its customers.

Overview

VRSG implements the following CIGI version 4.0 messages:

- IG Control
- Line of Sight Segment Request
- Line of Sight Vector Request
- Conformal Clamped Entity Control
- Hat/Hot Request
- Start of Frame
- Line of Sight Response
- Line Of Sight Extended Response
- Entity Control
- Hat/Hot Response

- View Define
- View Control
- Weather Control
- Celestial Sphere Control
- Collision Detection Segment Definition
- Component Control
- Rate Control
- Symbol Text Define
- Symbol Circle Define
- Hat/Hot Extended Response
- Articulated Part Control
- Short Articulated Part Control
- Atmosphere Control
- Collision Detection Segment Notification
- Short Component Control
- Symbol Control
- Symbol Line Define
- Symbol Short Control

VRSG can receive CIGI messages once it has been launched into visualization mode (by clicking Start VRSG on the Dashboard, or adding the *-autostart* command to the shortcut that starts VRSG).

By default, VRSG receives CIGI messages on UDP port 8021, and transmits CIGI messages on port 8022. VRSG does not broadcast CIGI messages; instead, VRSG sends CIGI messages to the IP endpoint address from which it receives CIGI messages.

VRSG is silent on the CIGI ports until a host attempts to communicate with it; VRSG will not issue a StartOfFrame (SOF) message until it receives an IG Control from a host.

You can override the default CIGI ports with the following DWORD registry variables:

- cigiRecvPort (the default port VRSG receives messages is 8021)
- cigiSendPort (the default port VRSG sends messages is 8022)

See the appendix "VRSG Registry Variables" for more information.

CIGI text define messages expect the UTF-8 encoding rules. Multiple bytes are needed in a string to represent character codes larger than 127.

VRSG controls database loading using normal methods, so VRSG will always send zero for the database number in the StartOfFrame message. By default, each CIGI *view* corresponds to a unique VRSG channel (computer). VRSG's entity ID, which you set in the Advanced Startup Parameters dialog box, defines the view ID for CIGI. A VRSG channel can be divided up into multiple viewports, with each viewport assigned to a different CIGI view ID. For more information, see the section "Working with viewports."

Multiple channels

You can control multiple channels from your CIGI host. It is not necessary to establish a unique and redundant CIGI connection with each VRSG channel -- multiple channels can be controlled via broadcast. The host should choose which VRSG channel is the master that it will sync to in response to IG Start of Frame messages. The host should ignore Start of Frame messages from all other channels. The host will be able to determine the originator of the Start of Frame message by examining the source IP address of the UDP packet.

You can disable the sending of Start of Frame messages from specific channels by adding the switch “-cigi_quiet” to the channels’ command line. Channels that are started this way will not transmit any CIGI messages back to the host. Disabling Start of Frame messages from certain channels is useful in a multi-channel environment where the host only wants to receive a Start of Frame message from a single (master) channel.

Entity control and special effects

CIGI uses Entity Control packets to create and control special effects and animations. The VRSG model for special effects is different, using a “fire-and-forget” approach to special effects. For consistency with CIGI, VRSG will allow you to create and place special effects using the CIGI Entity Control message.

Unlike CIGI, VRSG uses a separate namespaces for effects and entity types. To make the distinction between a special effect and a normal entity control, VRSG examines the high-order bit (0x8000) of the entity type. If this bit is set, the Entity Control is assumed to be for a special effect. The lower-order 15 bits (0x7fff) are interpreted as the special effect index, which is a zero-based index into the set of effects enumerated in the file \MVRsimulation\VRSG\Effects\ClientMap.ini. (See the chapter “Configuring Models and Events” for more information about ClientMap.ini.)

If the high order bit of the entity type is clear, the entity type is assumed to be for a normal entity. CIGI integer names for visual models are assigned to visual models in Models\ModelMap.ini using the “-type=” switch. For example, to assign the CIGI name 24 to the M1 tank entity, an entry in ModelMap.ini could appear as follows:

```
1 1 225 1 1 0 0 Vehicle M1A2.M2.US.desert.hpy -type=24
```

The DIS enumeration that precedes the model is not used for CIGI purposes. It is a good practice, however, to supply a valid DIS enumeration for the model. The domain and kind values of the DIS enumeration affect an entity's ground clamping behavior.

Articulated parts

MVRsimulation models follow the DIS convention for articulated parts and use DIS enumerations to identify articulated parts. In the DIS convention, an articulated part is identified by a 32-bit integer that is a multiple of 32. The low-order 5 bits of an articulated part ID is reserved for enumerations that describe the degree-of-freedom being controlled.

With CIGI, the articulated part ID and the degree-of-freedom being controlled are stored in separate data members. CIGI only allows an 8-bit namespace for part IDs. To convert a DIS part ID to a CIGI part ID, simply divide the DIS part ID by 32. For example, the main turret of a tank would be identified by part id 4096. When addressing this part via CIGI, use 4096 / 32 = 128.

You can determine the IDs assigned to articulated parts by opening the model in MVRsimulation’s Model Viewer, and inspecting the part IDs enumerated in the Articulated Parts menu. (as described in the chapter “Previewing Models, Effects, and Terrain”). Where possible, MVRsimulation's visual models follow the DIS standards for articulated part ID assignments.

Mission functions

You can control which VRSG channel processes mission functions and generates responses by selecting the Enable Mission Functions checkbox located on the Startup Parameters tab on the VRSG Dashboard. When this option is selected, VRSG processes mission functions requests and generates these potential responses:

- Hat/Hot Response
- Hat/Hot Extended Response
- Line of Sight Response
- Line Of Sight Extended Response
- Collision Detection Segment Notification

In a multi-channel environment, it is not necessary to have more than one channel processing mission functions. MVRsimulation recommends enabling mission functions on the lightest loaded channel, such as an upward-looking out-the-window visual channel.

Weather control

Global rain and snow effects can be turned on via the CIGI weather control messages, using the enumerations for layer of CIGI_WeatherLayerIDRain or CIGI_WeatherLayerIDSnow. The severity field controls the intensity, using values 0 (minimum) to 5 (maximum). (Using volumetric cloud instances for local precipitation is not required.)

Clouds

You can use the weather control to define up to two cloud layers. Setting a cloud type to zero disables a cloud layer. A cloud type of 15 indicates volumetric clouds. If volumetric clouds are selected, only 1 cloud layer will be available. You can control the texture used for solid cloud decks with the cloudType enumeration. Adjusting the coverage of volumetric clouds (layer id 15) requires a component control using a class of `CIGI_ComponentClassRegionalWeather` and a component ID of `CIGI_ComponentIdCloudCoverage`. The first data field contains the coverage (0..8) corresponding to the settings on the Dashboard.

For conventional non-volumetric cloud layers, the cloud type field selects the texture applied to the cloud layer top or bottom. VRSG loads textures with names of the form cloud0.rgb, cloud1.rgb, and so on, from the \MVRsimulation\VRSG\Textures directory. The cloud type field selects which of these textures is used for the cloud layer. The following table shows the cloud types of the cloud textures that are delivered with VRSG:

CIGI cloud type	Texture used	Layer coverage
1	\MVRsimulation\VRSG\Textures\Cloud0.rgb	100%
2	\MVRsimulation\VRSG\Textures\Cloud1.rgb	75%
3	\MVRsimulation\VRSG\Textures\Cloud2.rgb	50%
4	\MVRsimulation\VRSG\Textures\Cloud3.rgb	25%
15	N/A	Randomly scattered volumetric clouds.

You can add more cloud types to VRSG by adding more cloud*.rgb textures to the \MVRsimulation\VRSG\Textures directory. MVRsimulation provides multiple cloud layer texture types corresponding to various degrees of cloud coverage. Using per-built textures for different degrees of cloud coverage offers a better visual result than real-time modulation of a single texture. For this reason, the coverage parameter of the weather control is not used. Use the cloud type to select the degree of the intended cloud coverage. The automatic random scatter of volumetric clouds is disabled once the host asserts control over a cloud instance.

Volumetric clouds will move in response to the wind direction and velocity sent via the CIGI_AtmosphereControl.

Fog and haze

To adjust the haze distance/height use the visibilityRange field of the AtmosphereControl message.

For ground fog, send a WeatherControl with the layer Id set to `CIGI_WeatherLayerIDGroundFog`. The top altitude would be the baseElevation + thickness, and the density controlled by the visibilityRange field.

Fog color can be changed via `CIGI_ComponentIdLightingColors`. With this control, you also control ambient and diffuse colors.

Component controls

VRSG supports the use of component control messages to control certain aspects of VRSG not explicitly addressed by other CIGI messages. For a complete list of supported CIGI extensions via component controls, see the aforementioned CIGI.h header.

Sky model

Set the sky model by sending a component control with the component class field set to `CIGI_ComponentClassCelestialSphere`. The instance ID and component ID fields are ignored. The state field contains the sky model index. This zero-based index corresponds to the set of available sky models listed on the Environment tab in the VRSG Dashboard. Stars are rendered as light points. By setting the sky model to "none" (zero), light-point stars will be rendered at night. The celestial sphere control sets the date and time. The date and time in conjunction with the current location is used to control an ephemeris model, which computes sun position, moon position, and moon phase.

Entity scale

Set the scale of an entity model by sending a component control with the component class field set to `CIGI_ComponentClassEntity` and the component ID field set to `CIGI_ComponentIdEntityScale`. The instance ID field corresponds to the model ID The first 32-bit data word of the component control is interpeted as a 32-bit IEEE floating point number representing the entity scale.

Entity appearance mask

Set the DIS appearance mask of an entity by sending a component control with the component class field set to CIGI_ComponentClassEntity and the component ID field set to `CIGI_ComponentIdAppearanceMask`. The instance ID field corresponds to the model ID. The first 32-bit data word of the component control is interpreted the 32-bit DIS appearance mask.

Runway light intensity

You can control the intensity of runway light models on a per-model basis by sending a component control with the component class field set to `CIGI_ComponentClassGlobalTerrainSurface` and the component ID field set to `CIGI_ComponentIdRunwayLights`. The instance ID field corresponds to the model ID. The state field represents the light point intensity in the range of 0 to 100 (full intensity).

World relative view orientation

Normally the yaw, pitch, and roll angles of a view are interpreted to be relative to the entity the view is attached to (or the view group). VRSG offers a component control to change this interpretation to be world-relative. In this mode, the yaw angle is interpreted as relative to the north-facing direction (compass heading), and the pitch angle is relative to the horizontal ground plane. To enable this mode, send a component control with the component ID set to CIGI_ComponentIdWorldRelativeViewOrientation. The component class should be set to `CIGI_ComponentClassView`. The instanceId field is the ID of the view being controlled. The state field controls whether the world-relative effect is enabled (1), or operating as normal entity-relative (0).

Spectrum and sensor channel modes

You can set the visual spectrum of a sensor channel using the viewType field of the View Define packet. The following enumerations for the viewType are supported: 1 (Electro-Optic), 1 (Visual Out-The-Window), 3 (IR White Hot), or 4 (IR Black Hot).

Working with viewports

CIGI does not explicitly address the notion of viewports, but instead has the abstract notion of *views*. CIGI views are mapped to VRSG viewports. This could be multiple viewports on a single computer, single viewports on multiple computers, or some combination. An integer ID space is used to address these views. By default, VRSG assigns the whole channel to a CIGI view. The entity ID assigned to the channel on the Advanced Startup Parameters dialog box accessed from the Dashboard's Startup Parameters tab sets the CIGI view ID that the channel will respond to.

To have multiple viewports with CIGI, you create a file which can create viewports and map them to CIGI view IDs. To set up multiple viewports, create a file named CIGI_Viewports.txt or CIGI_Viewports_<hostname>.txt in the VRSG installation directory (\MVRsimulation\VRSG\). For example, if your computer's hostname is "IG1", you would name the file CIGI_Viewports_IG1.txt. This way, unique files per host can exist in environments where VRSG's file structure is stored on a shared server.

This file contains a list of each CIGI view and its viewport rectangle, with one entry per viewport. The syntax of each line is:

```
CIGI_ID X Y Width Height
```

For example, if you had a 3840 x 1200 frame buffer, you could partition it into two halves with a CIGI_Viewports.txt file with the following contents:

```
0    0  0 1920 1200
1 1920  0 1920 1200
```

In this example, CIGI View 0 would write to the left half of the double-wide frame buffer, and CIGI View 1 would write to the right half.

Note that viewports do not need to be spatially separate. They can overlap, for a Picture-In-Picture type of effect. Viewports are drawn in the order they are listed in the file, so the last one listed will be drawn on top of the others.

The CIGI_Viewports.txt file is used to establish the viewports on a channel with their initial size and origin. VRSG offers a component control interface that allows the host to move or resize a viewport dynamically. See the CIGI.h header for details about this and other custom CIGI interfaces. A CIGI host can suspend VRGS's rendering to a viewport by sending a View Define message with a zero field-of-view angle for all 4 angles to the associated CIGI view. VRSG will continue to draw overlays from CIGI symbology opcodes into views with zero field-of-view angles, but no 3D scene will be rendered. This enables a host to create overlay-only viewports that render overlay graphics with no 3D scene, which can overlay underlying viewports.

You can enable trackers on a per-viewport basis with the keyword `-enableTracker` as shown in this example:

```
1    0 0 640 400 -enableTracker
2  680 0 640 400
```

New in VRSG 7 is a CIGI component control, CIGI_ComponentIdOwnshipForViewport, which allows each viewport to have a unique ownship entity.

Symbology opcodes

VRSG has support for opcodes to control symbology/overlays, in addition to some special VRSG-specific extensions. The following sections describe these extensions.

Fonts

VRSG supports font IDs in the range of 0..255. Font 0 will use VRSG’s default internal font, which is Lucida Console, a fixed-pitch font. Font IDs 17..255 will also map to this default internal font.

Font IDs in the range of 1..16 correspond to the CIGI 3.3 standard, but can be overridden with registry variables. If you choose to override any of these fonts, use the variables located under HKEY_CURRENT_USER\SOFTWARE\MVRsimulation, Inc.\MVRsimulation Vrsg\Parameters. The following key names are supported:

cigiFontBoldN – where N is a value between 1 and 16. This should be a DWORD value where a value of 1 indicates bold, or 0 for a normal weight text.

cigiFontItalicN – where N is a value between 1 and 16. This should be a DWORD value where a value of 1 indicates italic, or 0 for a non-italic text.

cigiFontProportionalN – where N is a value between 1 and 16. This should be a DWORD value where a value of 1 indicates a variable-pitched font, or 0 for a fixed-pitch font.

cigiFontNameN - where N is a value between 1 and 16. This should be a STRING variable containing the name of the desired font (e.g. “Times New Roman”).

Textured polygons

VRSG supports a message for textured polygons. The structure of this message is defined in the CIGI.h header file.

Textured polygons provide an integer value to indicate the texture to apply to the primitive. You can provide these textures to VRSG by naming the textures “cigi_texture_N.rgb”, where N is the texture integer identifier provided in the textured polygon message. These textures may be stored in the \MVRsimulation\VRSG\Textures folder, or in any folder in VRSG’s search path.

Texture IDs with the high bit set (0x8000) have a special meaning. These do not refer to external files. Instead, they correspond to view IDs. If the high bit is set, the low-order 15 bits will be interpreted as a CIGI view ID. The texture applied to the primitive will be the contents of the scene rendered to the view. Note this assumes that the overlay is associated with a CIGI View that is not the same view as referred to by the texture ID. For example, if you have VRSG configured to render multiple CIGI views, the views are rendered in the order given in CIGI_Viewports.txt. If a textured polygon references a CIGI view as its texture, the textured polygon must be drawn in a view defined later in CIGI_Viewports.txt. In other words, a textured polygon cannot reference a view that has not yet been drawn in the current frame.

Overlay-only views

Overlay-only views can be defined in CIGI_Viewports.txt. Normally these views would be defined after all other CIGI views that contain normal 3D imagery. To create an overlay-only CIGI view, the host must send a View Control with field-of-view values of zero for all half angles. VRSG will not clear the viewport or render any 3D geometry onto such a view. The only way to render onto such a view is through the symbology opcodes.

Overlay-only views are useful for sensor-fusion applications, where textured polygons can be rendered referencing previously rendered views, using alpha blending to control the blend between the images.

Sample hosts

MVRsimulation offers sample CIGI hosts to help jumpstart your CIGI development. The sample hosts are provided with full source code and are buildable under Microsoft Visual

Studio. Our sample hosts cover typical needs of UAV and fixed-wing aircraft simulation. To request our sample hosts, contact support@mvrsimulation.com.

Troubleshooting

If your CIGI code is not working with VRSG, consider whether the problem is caused by one of the following:

- You have a problem with your network settings (exercise ID, UDP port, VRSG site and host ID values match that of the DIS simulation). To eliminate the possibility of network configuration problems, try running your host on the same machine that VRSG is running on and send packets to the loopback address.
- VRSG is being blocked by the Windows Firewall. When you run a program that opens a socket, Windows asks whether you want to unblock or continue to block the program. The initial user of VRSG at your site might have selected the "Keep Blocking" option. You can reverse this option in the Control Panel by opening the Windows Firewall. Click the Exceptions tab and locate *MVRsimulation VRSG* in the list of exceptions. If *MVRsimulation VRSG* is not in the list, you can add it to the list of exceptions by choosing Add Program. Make sure your host is also included in the exceptions list.
- WireShark is a useful tool for debugging CIGI interactions between VRSG and a host. WireShark is a general network analyzer, but it is CIGI-aware and will parse and display the contents of CIGI messages from the host or VRSG. You can obtain the WireShark tool from the website: www.wireshark.org/download.html.
- If the VRSG scene is not responding to view controls, but you are receiving Start-of-Frame messages and otherwise appear to have a valid connection, the likely cause is VRSG is not set to a CIGI view ID that is being controlled by your host. If you are not using CIGI_Viewports.txt, the CIGI view ID is set from the Entity ID field under Startup Parameters > Advanced. If you are using CIGI_Viewports.txt, ensure that the view ID your host is controlling is enumerated in CIGI_Viewports.txt.

APPENDIX G

Communication Ports

This appendix lists the default communication ports used by MVRsimulation products.

VRSG ports

All but one VRSG port setting are configurable. Most can be changed via a setting in the VRSG Dashboard or by setting a registry variable. (For the latter settings, see the appendix "VRSG Registry Variables.")

Component/Protocol	**Default port**	**Note**
DIS UDP	3000	VRSG Dashboard setting.
CIGI UDP	8021 (read), 8022 (send)	Registry variable setting.
H.264 and MPEG UDP	No default	VRSG Dashboard setting.
MUSE, DIS	5000	VRSG Dashboard setting.
Radar UDP	6011	VrsgRadarICD header file.
Radar Host UDP	6012	VrsgRadarICD header file.
Scenario Editor	3000	Network Settings; must be the same DIS UDP port as VRSG.
Viewpoint Protocol	5069 (read), 5070 (send)	MVRsimulation's viewpoint protocol for legacy simulators.
License Authentication UDP	8017	Not configurable.

Terrain Tools ports

The Terrain Tools Build Manager uses two communication ports for building terrain tiles. One port is for carrying out build requests on all client machines and providing status information to the Build Manager user interface. The other port is for the Build Manager's web user interface, where you manage the terrain build configuration and monitor build status on all client machines. Both ports are preferences that can be changed per machine in the Build Manager window.

Component/Protocol	Default port	Note
Build Manager UDP/TCP-IP	8900	Build Configuration Preferences setting for a selected client machine.
Build Manager web-based user interface TCP-IP	8901	Build Configuration Preferences setting for a selected client machine.

APPENDIX H

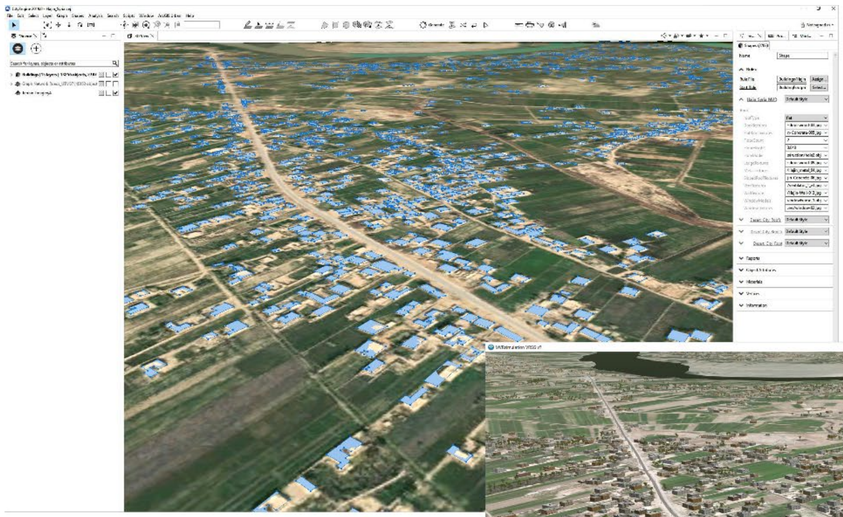
Adding CityEngine Generated Models to 3D Terrain

You can use CityEngine to generate a 3D urban model of a large number of buildings to supplement terrain compiled in Terrain Tools, for rendering in VRSG. CityEngine is a standalone software product for rapidly creating procedural 3D urban models.

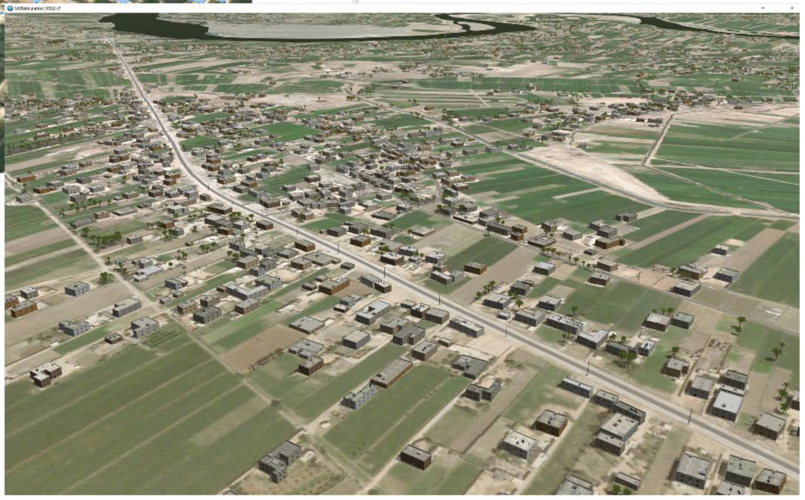
CityEngine generates procedurally created models from building footprints and road vectors using Open Street Map (OSM) imagery data and your provided elevation or heightmap data. These building footprints and road vectors are extruded and textured from OSM data using Computer Generated Architecture (CGA) rule files. A single CGA rule file can be used to generate many 3D models using feature attribute information stored in the OSM data.

You can create cultural content from CityEngine to render in VRSG, in one of two simple workflows:

- In MVRsimulation Terrain Tools version 1.6 and above, use the CityEngine feature type for compiling extruded building features using a CityEngine rule package (.rpk) to enable adding large areas of dense culture to 3D terrain quickly and easily. An RPK is a compressed 7-Zip archive package, created in CityEngine, which contains compiled Computer Generated Architecture (CGA) rule files (*.cgb), and all referenced assets and data. RPKs can be applied to any building footprints. Using an RPK within Terrain Tools does not require the use of CityEngine, if you can obtain an appropriate RPK to compile with your terrain. This workflow is described in the *MVRsimulation Terrain Tools User's Guide.*
- In CityEngine, you output an urban model of an area of dense culture to FBX or Collada's DAE format (for converting the latter to FLT format in Presagis Creator), and then use MVRsimulation's FBX or FLT model conversion utility to convert the model to MVRsimulation's model format. Then, reference the urban model in a cultural feature file (vrsg.clt) like any other model that you want VRSG to load at runtime. This appendix describes this workflow, using OSM imagery data sources.



Left: The city of Hajin, Syria's building footprints in CityEngine prior to exporting an urban model of 13,326 realistic 3D buildings in FBX format for conversion to MVRsimulation's model format.



Below: The resulting city rendered in VRSG. The terrain was built with 50 cm imagery in MVRsimulation Terrain Tools.

Considerations for setting up the CityEngine scene

When you set up your CityEngine scene of the urban model you intend to convert to MVRsimulation's model format and render in VRSG, keep in mind these three considerations:

- If you intend to export your building models in FBX format, you must have CityEngine Advanced, which supports exporting models in FBX format. (CityEngine Basic does not export models in FBX format.). Both CityEngine Basic and Advanced export models in DAE format, which you can import into Presagis Creator for exporting the model in OpenFlight format.
- When you import source data into the CityEngine scene file, you must specify the source data's coordinate system as projected UTM WGS1984, and then choose the corresponding UTM zone. Again, this elevation data must be in the UTM WGS84 projection. (VRSG natively supports geocentric WGS1984, but will support UTM WGS 1984 when the –utmModel flag accompanies the model's entry in the vrsg.clt cultural feature file, as described later.)
- Break up large city footprints into sections approximately 5 km x 5 km to ensure optimal performance in VRSG for rendering large models. (You can use ArcGIS Pro to split up a

building footprint shapefile into smaller models.) You would list all these models as separate entries in the terrain's vrsg.clt cultural feature file.

- Be sure to import your elevation source data (or a Terrain Tools-generated heightmap) into the City Engine scene file, as this data will be used as the height map for your models. Once you add the elevation source data or heightmap, select the option "Align shapes to the terrain" to properly align your models on the terrain elevation. Be sure the elevation or heightmap layer is selected as the height map when you align your shapes.

Exporting the model from CityEngine

When you export your urban model from CityEngine, be sure to set the following:

- For exporting a model to FBX format, choose Advanced Settings > File Type, and then select Text from the drop-down menu to export the model in ASCII format.
- Choose Geometry Settings > Global Offset, and then select Center to center the origin of the model before exporting the model.
- Choose General Settings > Terrain Layers, and then select 'Do not export any terrain layers.'
- Choose AutoDesk FBX as the output format, so that you can use MVRsimulation's FBX conversion utility to convert the model to MVRsimulation's model format.
- Choose Collada DAE as the output format, if your modeling tool is Presagis Creator (you can choose a Collada version as well). You would import the DAE model into Presagis Creator to convert it to FLT format, or simply use MVRsimulation's HPX plugin for Creator.

CityEngine does not export levels of detail (LODs) with a model. If you want your resulting model to have LODs when it is rendered in VRSG, you will need to add them in the appropriate model editing application (such as Autodesk Maya or 3ds Max for FBX models; Presagis Creator for FLT models) prior to converting the model to MVRsimulation's model format.

Converting FBX or FLT model to MVRsimulation model format

MVRsimulation's Fbx2Hpx command-line utility, which is delivered with VRSG, converts an FBX-formatted model to MVRsimulation's HPX model format for rendering in VRSG. The utility is located in \MVRsimulation\Common\Util\Fbx2Hpx.

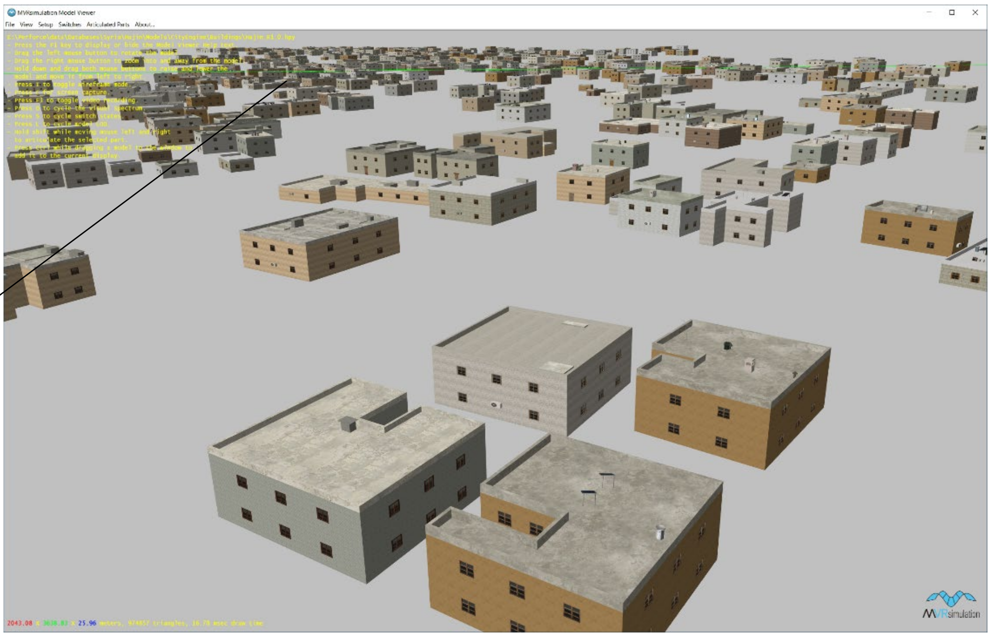
To use the FBX conversion utility, type "Fbx2Hpx" in a command-line window, followed by the name of the model you want to convert, with or without parameters. For more information about using the utility, see the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats".

To convert an OpenFlight format model to an MVRsimulation HPX-formatted model, you can use one of the following methods:

- Within Presagis Creator, use MVRsimulation's hpxPlugin.dll converter. Simply install this plugin in the \Plugins directory of Creator and then restart Creator. The MVRsimulation HPX model format will be available as an export option.
- Use MVRsimulation's oflt2Hpx.exe converter, by typing "oflt2Hpx" in a command-line window, followed by the name of the model you want to convert, with or without parameters. MVRsimulation has developed three command-line options for this converter specifically for optimizing the rendering of CityEngine generated models in VRSG. See the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats" for more information.

These utilities are delivered with VRSG, and are located in \MVRsimulation\Common\Util\OpenFlight. For more information about using the FLT conversion utilities, see the chapter "Converting FBX and OpenFlight Formats to MVRsimulation Runtime Formats."

After converting the urban model to MVRsimulation's HPX format, you can inspect the model in MVRsimulation Model Viewer to ensure that the origin of the model is where it is expected and that the model converted properly.



```
E:\Perforce\data\Databases\Syria\Hajin\Models\CityEngine\Buildings\Hajin_01_0.hpx
- Press the F1 key to display or hide the Model Viewer Help text.
- Drag the left mouse button to rotate the model.
- Drag the right mouse button to zoom into and away from the model.
- Hold down and drag both mouse buttons to raise and lower the
  model and move it from left to right.
- Press I to toggle wireframe mode.
- Press C for screen capture.
- Press F3 to toggle video recording.
- Press B to cycle the visual spectrum.
- Press S to cycle switch states.
- Press L to cycle model LOD.
- Hold shift while moving mouse left and right
  to articulate the selected part.
- Press Ctrl while dragging a model to the window to
  add it to the current display.
```

The Model Viewer displays the X (red), Y (green) and Z (blue) axes, which meet at the origin of the model.

For information about Model Viewer, see the chapter "Previewing Models, Effects and Terrain."

Visualizing the model on 3D terrain in VRSG

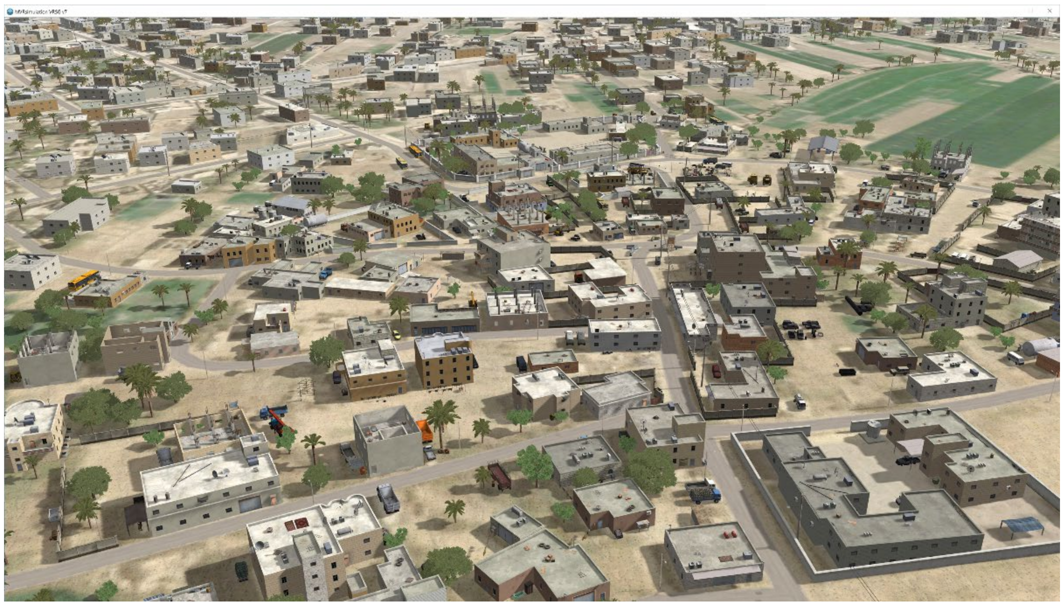
To visualize the converted urban model on 3D terrain in VRSG, you first add an entry for the model to the vrsg.clt cultural feature file used for the terrain. This entry must be added to vrsg.clt by hand. Although CityEngine-generated models are useful to display on the terrain in Scenario Editor while you are adding other content, these models are too big to place on the terrain directly in Scenario Editor by a drag-drop action. As CityEngine-generated models are tied to a specific footprint and elevation, there is no need to manipulate them in Scenario Editor.

To add an entry to a terrain's vrsg.clt file:

1. Locate the coordinates of the origin of the model.
2. In a text editor, create a new entry for the model in the terrain's vrsg.clt cultural feature file (or an appropriate vrsg.clt file, if your terrain uses more than one). In this entry, add the coordinates of the origin of the model. Add the –utmModel flag to the end of the entry to ensure proper alignment of this UTM-projected model in VRSG. For example:

```
N25 12 35.962 E055 16 39.895 6.00 0.00 0.00 0.00
Dubai_Buildings.hpy -utmModel
```

3. Start VRSG.
4. On the Dashboard's Startup Parameters tab, set the search path, ensuring that the vrsg.clt file and model are both in the search path.
5. Launch the VRSG visualization session.
6. Navigate to the area of interest and inspect the model.



APPENDIX I

License and Warranty

MVRsimulation Inc. Software License Agreement

Version 45665 (10 February 2025)

DO NOT INSTALL THIS SOFTWARE UNLESS YOU HAVE READ, UNDERSTOOD AND AGREE TO THESE TERMS

MVRsimulation Inc. ("**MVR**"), a Massachusetts corporation, is supplying a copy of its VRSG®, Scenario Editor or Terrain Tools software to You (as that term is defined below) on the terms set forth herein and in the Purchase Order (as defined below). This License Agreement is entered into as of the date first written above between MVRsimulation and the undersigned licensee ("**you**"), with an address indicated below your signature line. Now, therefore, you and MVRsimulation agree as follows:

Article 1 DEFINITIONS: As used herein:

"**Additional Materials**" means software not constituting Software (as defined below), Data Files not specified in a Purchase Order or created as part of the License Authentication process, and libraries (including Model libraries) not specified in a Purchase Order.

"**Authentication Files**" means the files in .C2V, .V2C or .A2C format that are exchanged in the course of a License Authentication or the reinstallation of the Software onto another computer.

"**Data Files**" means data files supplied by MVR under a Purchase Order or as Additional Materials or created by you or MVR as part of the License Authentication process.

"**Demonstration Video**" includes any visual display involving VRSG, Scenario Editor or Terrain Tools or the Data Files that can be viewed by a third party, whether hosted on a public server, circulated in a format that permits individual viewing, or demonstrated at your facility or that of a third party.

"**Documentation**" means documentation published by MVR describing the functions and operation of the Software.

"**Dongle ID**" means a unique digital identifier of a physical dongle transmitted as part of the authentication process described in Section 2.02.

"**Initial Maintenance Term**" is a one year term beginning on the date on which you first receive a long-term License Authentication or, if different, the maintenance term is specified in the Purchase Order.

"**License Authentication**" means the authentication described in section 2.02 below.

"**Licensed Materials**" means the Materials and Additional Materials.

"**License ID**" means a unique digital identifier of a physical computer installation transmitted as part of the authentication process described in Section 2.02.

"**Maintenance Term**" means the Initial Maintenance Term as it may be extended from time to time pursuant to Section 3.04.

"**Materials**" means the Software, related Documentation, and any Data Files, Models or libraries supplied by MVR under a Purchase Order.

"**Models"** means the three-dimensional representations of vehicles, weapons, characters, and cultural features such as buildings, foliage, signage and street elements made available by MVR.

"**Proprietary Files**" means Models or Data Files provided in any proprietary file format of MVR, including without limitation files provided in the .VIR, .MDY, .MDX, .MDS, .HPZ, .HPY, .HPX, .TEX, .C2V, .V2C and/or .A2C format.

"**Purchase Order**" means the purchase order, quotation or online order form, including the Standard Terms, pursuant to which you have agreed to purchase and MVR has agreed to sell licenses of Software.

"**Refresh**": A License Authentication may be "**Refreshed"** if you transfer your Software to a new computer or request the right to use additional Software features beyond those in your original Purchase Order. This refreshing may take the form of reprogramming of a dongle or the exchange of new Authentication Files.

"**Scenario Editor**" means MVR's VRSG Scenario Editor software product.

"**Software**" means the MVR software (which may include VRSG) identified in the Purchase Order and any upgrades, updates, new releases, versions, corrections or revisions thereto which MVR makes or has previously made available to you.

"**Standard Terms**" means MVR's standard terms and conditions available at www.mvrsimulation.com/howtobuy/standard_terms_and_conditions.html or, if attached as an exhibit to the Purchase Order or this license agreement, in the form so attached.

"**Terrain Tools**" means MVR's Terrain Tools for Esri ArcGIS software product.

"**Updates**" means updates, upgrades and new releases and subreleases of VRSG, Scenario Editor or Terrain Tools software product.

"**VRSG**" means MVR's Virtual Reality Scene Generator software product.

"**You**" means the legal entity or governmental agency on behalf of which the user of this copy of VRSG, Scenario Editor or Terrain Tools is acting in installing and using this copy of VRSG, Scenario Editor or Terrain Tools. This definition applies even when "you" appears with a lower case 'y'.

Article 2 LICENSE

Section 2.01 **Grant.** MVR grants you a non-exclusive license (i) to install the Licensed Materials onto a computer under your control, whether the computer is accessed directly or over a private network and when installed over a private network, and (ii) for you or your employee (and, if you are an accredited college or university, your students) to use the Licensed Materials and Software so installed for your own internal business, operational or educational purposes, and, if you are acquiring this license in connection with a United States government contract identified in the Purchase Order, for the purpose of fulfilling your obligations under that contract, provided, however that the Licensed Materials may only be run or otherwise used on a computer that has been authenticated, as described in Section 2.02.

Section 2.02 **Authentication.** Authentication of a computer may occur either as a result of an exchange of Authentication Files or by your use of a dongle obtained from MVR. If your Purchase Order refers to authentication by means of Authentication Files, the exchange of Authentication Files will occur when you install the Software on a computer. As part of the installation process, the Software will collect and compile unique identity information of the physical computer either (i) into a .C2V file, or (ii) into a License ID output to the computer's video display, to be transmitted via email to MVR. Upon receipt of the .C2V file or the License ID code, MVR will create and send by reply e-mail either a .V2C or .A2C Authentication File which you will be able to use to complete the authentication process. MVR's transmission to you of the Authentication File will trigger the commencement of the Initial Maintenance Term for the copy of VRSG, Scenario Editor or Terrain Tools to which it is related.

For users whose authentication is to be done by means of dongle(s), MVR will provide the number of dongles stated on the Purchase Order. An individual dongle may not be used by more than a single user or on more than a single computer simultaneously. As part of the installation process, you will be prompted to send the applicable Dongle ID(s) via email to MVR. Upon receipt of these Dongle ID(s), MVR will send by reply emailan unlock code for each dongle for which a Dongle ID has been received. You will be able to use these unlock code(s) to complete the authentication process. MVR's transmission to you of the unlock code for a dongle will trigger the commencement of the Initial Maintenance Term for the copy of VRSG, Scenario Editor or Terrain Tools to which it is related.

Although MVR may provide you with temporary License Authentication (which may involve a temporary unlock code for a dongle or a .A2C or .V2C authentication file with a temporary duration), MVR shall have no obligation to extend any such temporary License Authentication, provide additional temporary License Authentication or provide you with a permanent License Authentication unless and until you have paid in full for the Software and any other services or materials identified in the Purchase Order. Accordingly, your use of the Software may be interrupted until such payment has actually been received and processed by MVR.

Section 2.03 **Restrictions**. You may not do or permit any other party to do any of the following:

a) Use the Licensed Materials other than as specifically permitted in this Agreement or permit any other person to do so;

b) Make the Licensed Materials available to, persons other than those described in clause (ii) of Section 2.01 or more concurrent users than the number of licenses that you have purchased;

c) Make the Licensed Materials available for use on a non-Authenticated computer;

d) Permit simultaneous use of the Licensed Materials by more users than are authorized by the License Authentication, whether over a computer network, or on a virtual or emulated computer or otherwise;

e) Attempt to (a) alter, merge, modify, adapt, or translate any Licensed Materials (excluding only Data Files that are not Proprietary Files), (b) decompile, reverse engineer, disassemble, derive, or otherwise reduce the Software to a human-perceivable form, or (c) develop any software that would permit an end user to read or access any Proprietary Files. The parties understand that, as used herein, the term "reverse engineer" shall include, without limitation, any use of benchmarking information or incremental output from the Software to determine MVR source code, algorithms or data format, or for the purpose of recreating the Software (including without limitation any MVR Files) or creating software or files substantially similar thereto;

f) Use any Licensed Materials for the purpose of training, developing or operating any form of artificial intelligence, whether artificial narrow intelligence (or "weak AI), artificial general intelligence (or "strong" AI), including without limitation using Licensed Materials to train large language models, without a specific, separate license from MVR authorizing such use;

g) Bypass the copy protection code or any other technological measure that controls access to the Licensed Materials;

h) Make copies of any of the Licensed Materials other than one copy for back-up or archival purposes or use a back-up copy other than as a replacement for the original copy. You must include on any back-up copy all copyright and other notices included on the Materials or Additional Materials;

i) Export, re-export or use the Licensed Materials or any copy thereof in violation of the export control laws of the United States of America or any other country;

j) Use any dongle or Authentication File supplied to you by MVR in connection with the license of the Software in any manner other than in connection with the use of the Materials as permitted hereunder;

k) Use the Software to test or analyze the performance or user interface of the Software in order to develop or improve a product which competes with the Software;

l) Publish or provide to third parties performance characteristics relating to the Licensed Materials without the express written consent of MVR;

m) Demonstrate the Licensed Materials in public or private forums without using (i) a platform that provides the Software sufficient performance capacity to operate at peak capacity, and (ii) the most recent versions of any Software and Data Files;

n) Use or circulate any Demonstration Video without first obtaining the approval of MVR, which approval MVR may withhold if it reasonably determines that the Demonstration Video will not correctly reflect the peak performance of the Software and Data Files;

o) Publish or provide this document to third parties in electronic or printed form;

p) Create or distribute derivative works based upon the Licensed Materials (including without limitation databases derived from files provided to you by MVR).

Section 2.04 **Updates to License Agreement.** If you renew, revive or extend the Maintenance Term, Refresh your License Authentication, or request and receive any Updates or Additional Materials, you may be prompted to accept the then-current version of this Agreement, which if accepted shall apply to all Licensed Material. You will not be able to use Updates or Additional Materials or extend or revive the Maintenance Term without accepting the then-current version of this Agreement unless MVR then expressly permits it (which it may refuse in its absolute discretion), but any extension of the Maintenance Term previously paid for will continue in effect under the existing Agreement until it expires.

Section 2.05 **Ownership and Copyright.** Title and copyright to the Licensed Materials (including, without limitation, any databases, libraries, images, "applets", photographs, animations, video, audio or music and text incorporated therein and any hardware keys provided in connection therewith) and all copies thereof remain with MVR and/or its licensors. The Licensed Materials are copyrighted and are protected by United States copyright laws and international treaty provisions. You may not remove the copyright and other proprietary rights notices from any the Licensed Materials. You agree to prevent any unauthorized copying of the Licensed Materials. Except as expressly provided herein, MVR does not grant any express or implied right to you in the Licensed Materials or under the patents, copyrights, trademarks, or trade secret information of MVR or its licensors.

Article 3 MAINTENANCE AND SUPPORT

Section 3.01 **Initial Support.** Except as you and MVR may otherwise have mutually agreed in writing, MVR will provide technical support and maintenance in connection with your use of the Materials during the Initial Maintenance Term. If MVR later provides you with any subsequent License Authentication (as may arise if the Software is transferred to a replacement computer, or MVR provides you with a replacement dongle or any new unlock code for a dongle), your subsequent License Authentication shall not restart or extend the Maintenance Term. If you have purchased multiple seats of the Software and have therefore received multiple License Authentications, the Materials accessed with each License Authentication will have a separate Maintenance Term, based on the date on which MVR first provides you with a long-term License Authentication for the applicable copy of the Software. Upon the expiration or termination of the Maintenance Term, MVR will have no obligation to provide maintenance or support for the Materials. MVR may temporarily suspend its support in the event of interruptions beyond its reasonable control, such as may arise in the event of flood, earthquake, terrorist attack, failure of third party communications systems and the like.

Section 3.02 **Nature of Support.** All maintenance and support will be provided remotely during MVR's normal business hours. The maintenance and support provided by MVR hereunder will consist of:

(a) responding promptly via e-mail during the Maintenance Term to any questions regarding reports of errors or defects in the Software;

(b) providing assistance via e-mail and telephonically during the Maintenance Term relating to the installation and use of the Software;

(c) if you first received the Materials electronically, permitting you to re-install the Materials electronically on the same or a different computer until the end of the Maintenance Term or, if longer, until the fifth anniversary of the end of the Initial Maintenance Term. However, if you extend the Maintenance Term beyond six years, MVR reserves the right to provide you with a later version of the Materials than the version that was specified in your Purchase Order; and

(d) providing access during the Maintenance Term to (1) any "Additional Materials" generally released to MVR's customers, (2) error corrections (i.e. patches) and updates intended to fix reported errors, and (3) all product updates, upgrades and enhancements to the Software that MVR generally releases to its customers during the Maintenance Term. If you request that MVR perform maintenance or support on site at your premises or outside normal business hours, you will be responsible for all reasonable expenses incurred by MVR in connection with performance thereof and MVR reserves the right to charge its then prevailing rates per hour of service provided, or to refuse to provide such service.

Section 3.03 **Exclusions and Limitations.** MVR's maintenance and support obligations do not include: custom programming, training, implementation, database changes or support, product upgrades for which MVR generally imposes a separate price or fee, any requests for content revisions or changes of any kind, or any other matters not specifically covered hereunder. In addition, the provision of maintenance or support hereunder is subject to commercial availability and technological compatibility and the absence of any actual or threatened litigation between you and MVR. MVR provides no guarantee that the Software or any upgrades or updates to the Software provided pursuant to this Maintenance Agreement will function or perform when used on equipment that does not conform to the specifications indicated in the documentation relating thereto. MVR shall have no obligation to provide maintenance or support if you use or attempt to use the Software with hardware that fails to meet the minimum requirements indicated in the Documentation or otherwise modify, revise or transfer the software other than as permitted herein or without MVR's prior written permission.

Section 3.04 **Extension of Maintenance Term.** At any time before the expiration of the Maintenance Term associated with a particular copy of the Software, you may extend such Maintenance Term for successive one (1) year terms, provided, however, that such extensions shall no longer be available or applicable if MVR has generally discontinued maintenance of the licensed version of the Software (in which case MVR shall refund any amounts that you have already paid for extensions that have not yet commenced). For each annual extension of the Maintenance Term associated with a copy of the Software, you shall pay MVR the applicable maintenance fee set forth in the Purchase Order for extension of the Maintenance Term or, if an applicable maintenance fee is not specified, the price then generally charged by MVR for the maintenance and support services described herein. All payments for extensions purchased hereunder must be received by MVR before the expiration of the then current Maintenance Term.

Section 3.05 **Termination of Maintenance Term.** Except when the software is transferred to U.S. Government Customer, the Maintenance Term will terminate automatically in the event that you (a) materially breach any term of this Agreement (including without limitation any payment obligation contained herein), or (b) violate or infringe any of MVR's intellectual property in any manner unless such proposed termination is governed by the Contract Disputes Act of 1978, in which case MVR may pursue its rights in the manner prescribed in that Act and the regulations promulgated thereunder.

Section 3.06 **Effect of Termination or Expiration.** The Software and any updates, upgrades and enhancements thereto that are installed during the Maintenance Term will continue to be accessible after the expiration or termination of the Maintenance Term. However, even though you may be able to download and install product updates, upgrades and enhancements to the Software after the expiration or termination of the Maintenance

Term, these updates, upgrades and enhancements will not function on your computer unless and until you have revived and reinstated the Maintenance Term as described below.

Section 3.07 **Revival.** In the event that you wish to receive maintenance and support after the expiration or termination of the Maintenance Term, you may request that the Maintenance Term be revived and reinstated for a new one (1) year term. MVR may, accept or refuse such a request in its sole and absolute discretion. Upon notice that MVR has agreed to revive the Maintenance Term, you shall pay to MVR a maintenance fee at its then-applicable rates for (a) the one (1) year term commencing as of the date when MVR receives the payment and (b) any gap period (measured in months) between the end of the expired or terminated Maintenance Term and the commencement of the newly revived one-year term described in clause (a).

Article 4 SAFEKEEPING OF DONGLES AND AUTHENTICATION FILES

If you have been provided a dongle or an Authentication File format in connection with your license of Software, the safekeeping and preservation of the dongle or Authentication File is your responsibility. If you need a replacement dongle or Authentication File, MVR will provide one at its then-current rates for such replacement but only if MVR is satisfied that there has been no attempt to use it in a manner not permitted under this Agreement**. For that reason, MVR strongly recommends that you back up your .A2C or .V2C files to a separate storage device and retain that backup file for archival purposes.** To obtain a replacement dongle, you must physically return it (including the casing with identifiable labeling and all mechanical and electrical parts) to MVR so that MVR can troubleshoot any damage to it.If your Authentication File has become unusable, you will likewise have to return it to MVR and certify that you have not kept any copies of it. MVR may in its discretion provide you with a dongle or Authentication File that can only be used for a short period of time when it sends you the invoice for the replacement. Your ability to receive a permanent License Authentication is subject to MVR's receipt of payment in full of the invoice; per Section 2.02, paragraph 3. If you wish to transfer Software to a new computer or change the operating system of your existing computer, see Section 7.02 for instructions as to that process.

Article 5 LIMITED WARRANTIES; DISCLAIMERS; REMEDIES; INDEMNITY

Section 5.01 **Limited Warranties.** MVR warrants that:

(a) The Software will perform substantially in accordance with the technical functionality set forth in the Documentation during the Maintenance Term; and
(b) Any physical media on which the Materials are delivered will be free from defects in material and workmanship that will prevent you from loading the Software on your computer for a period of sixty (60) days from the date of shipment to you.

These warranties shall be null and void in the case of any defect caused by any of the following: (i) modification of the Materials by any party other than MVR; (ii) use of the Materials with hardware or software other than that supplied or recommended by MVR; (iii) other improper or unauthorized use of the Materials by you; (iv) failures or defects in third party software or hardware; or (v) external factors such as, but not limited to, power failures or electrical surges.

Section 5.02 **Remedies.** If the Software fails to perform substantially in accordance with the Documentation, your sole remedy is to initiate a technical support ticket by contacting MVR at support@mvrsimulation.com; and MVR's sole obligation will be to provide reasonable commercial efforts to resolve the issue to your satisfaction in accordance with Section 3.02. Your sole and exclusive remedy with respect to any defective media shall be the right to return such media to MVR, and MVR's sole liability to you shall be the replacement of any defective media.

Section 5.03 **DISCLAIMERS.** EXCEPT AS SET FORTH ABOVE, THE MATERIALS ARE PROVIDED "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF SATISFACTORY QUALITY, MERCHANTABILITY, NONINFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, NO WARRANTY IS MADE THAT THE ENCLOSED SOFTWARE WILL GENERATE COMPUTER PROGRAMS WITH THE CHARACTERISTICS OR SPECIFICATIONS DESIRED BY YOU OR THAT THE MATERIALS WILL BE ERROR-FREE. THE WARRANTY PROVIDED HEREIN DOES NOT EXTEND TO ANY HARDWARE PURCHASED FROM MVR. ANY SUCH WARRANTY MUST BE PROVIDED IN A SEPARATE WRITING. THESE DISCLAIMERS OF WARRANTY CONSTITUTE AN ESSENTIAL PART OF THIS AGREEMENT.

Without limiting the foregoing, MVR provides no guarantees that the Software or any upgrades or updates to the Software provided as part of the maintenance and support described below will function or perform when used on equipment that does not conform to the specifications indicated in the Documentation relating thereto.

Because certain jurisdictions prohibit the waiver of certain warranties, the above disclaimer may not apply to you and you may have additional legal rights that vary by jurisdiction.

Section 5.04 **LIMITATION ON LIABILITY.** TO THE MAXIMUM EXTENT PERMITTED BY LAW, IN NO EVENT SHALL MVR OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION OF THE MATERIALS BE LIABLE FOR ANY DAMAGES OR LOSSES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT DAMAGES, INCIDENTAL OR CONSEQUENTIAL DAMAGES, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, LOSS OF INFORMATION AND/OR LOSS OF DATA), WHETHER ARISING IN TORT, CONTRACT OR OTHERWISE, ARISING OUT OF THE USE OF OR INABILITY TO USE THE MATERIALS, EVEN IF MVR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, MVR'S ENTIRE LIABILITY HEREUNDER SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID FOR THE MATERIALS.

Because some jurisdictions prohibit the exclusion or limitation of liability for damages, the above limitation may not apply to you and you may have other legal rights that vary by jurisdiction.

Section 5.05 **Indemnity.**

(a) MVR shall defend, indemnify, and hold harmless you and your permitted assignees (under Article 7), customers (collectively the **"Indemnitees"**) from and against any claims, damages, losses, costs, and expenses, including reasonable attorneys' fees, arising out of any action by a third party that is based upon a claim that any work performed or Software provided infringes or otherwise violates the intellectual property rights of any person or entity.

(b) If an Indemnitee's use of the Software is held to constitute infringement or is enjoined, MVR shall at its option and expense (i) procure for Indemnitee the right to continue using the Software, or (ii) replace or modify the same with Software that is non-infringing and provides equivalent functionality acceptable to the Indemnitee.

(c) MVR's obligation to indemnify Indemnitees under this Article shall not apply if the alleged infringement is based upon you or any Indemnitee's unauthorized modification of the Software or the use of the Software in combination with other Software or devices, where such combination caused the infringement and where infringement would not have occurred from your or an Indemnitee's use of the Software alone.

(d) <u>Commercial Customers.</u> If you are not an agency of the United States government, you agree to indemnify MVR and its affiliates against any loss, liability or expense (including reasonable legal fees) it incurs arising out of or in connection with any breach or violation of the terms of this Agreement by you or your use of the Materials.

(e) <u>Purchases By U.S. Government.</u> If you are the United States government or an agency thereof, then you authorize and consent to all use and manufacture, in performing this license, of any invention described in and covered by a United States patent,

(1) Embodied in the structure or composition of any article the delivery of which is accepted by the United States government under this contract; or,

(2) Used in machinery, tools, or methods whose use necessarily results from compliance by MVR with (i) specifications or written provisions forming a part of this contract or (ii) specific written instructions given by the Contracting Officer directing the manner of performance. The entire liability to the Government for infringement of a United States patent shall be determined solely by the provisions of the indemnity clause, if any, included in this license, and you assume liability for all other infringement to the extent of the authorization and consent hereinabove granted.

Article 6 <u>CONFIDENTIALITY</u>

You agree that the Materials, the object and source code of the Software, the algorithms used by the Software, the performance characteristics of the Software, and the algorithms and functioning of any dongles or Authentication Files provided to you in connection therewith (collectively, the "**Confidential Information**") are or contain the confidential and proprietary information and trade secrets of MVR and that MVR is providing the Materials to you in confidence. You shall not and you shall not permit others to reverse engineer the Software (including by analysis of benchmarking or output information) or to access the source code, algorithms or performance characteristics of the Software, the dongles or the Authentication Files. In addition, you agree (i) to preserve in strictest confidence all Confidential Information, (ii) not to disclose the Confidential Information to any third party except as expressly permitted herein, (iii) only to disclose the Confidential Information within your business organization to those employees (and, if you are an accredited college or university, your students) who have first agreed to be bound by the terms and conditions substantially similar to those contained herein, (iv) not to disclose any Confidential Information to any agents, contractors or consultants except if such disclosure is related to the authorized use of the Materials hereunder and after having received a commitment from such agents, contractors or consultants to be bound by substantially similar obligations with respect to such information as you are hereunder, and (v) not to use the Confidential Information for any reason except in connection with the authorized use of the Materials. You shall be responsible and liable for any unauthorized disclosure, publication or dissemination by any of your employees, students, agents or contractors of any Confidential Information. Confidential Information shall not include any information which: (a) you possess prior to the receipt hereof without obligation of confidentiality; (b) you rightfully receive from a third party without any obligation of confidentiality to such third party, and which such third party received without any obligation of confidentiality, direct or indirect, to MVR; or (c) is or becomes publicly available lawfully and without breach of any obligation to MVR by you. You may disclose Confidential Information if such disclosure is required under the terms of any statute, regulation, order, subpoena or document discovery request, provided that prior written notice of such disclosure is furnished to MVR as soon as practicable in order to afford MVR an opportunity to seek a protective order or otherwise contest or restrict such required disclosure. The parties agree to cooperate fully to limit disclosure in the event of any apparent legal requirement that Confidential Information be disclosed.

Article 7 TRANSFER AND ASSIGNMENT

Section 7.01 **Transfer or Assignment by Licensee.** Except as otherwise permitted by law or expressly permitted herein, you may not transfer or assign this Agreement or the Licensed Materials to another person without the prior written permission of MVR, except:

(i) if the Purchase Order identifies a U.S. Department of Defense ("DOD") contract in furtherance of which you have ordered the Materials, this license may be transferred to any other DOD contractor who needs the Materials in furtherance of a DOD contract or to the United States government agency for which the contract is being performed,

(ii) in connection with the sale of all or substantially all of your assets, this license may be transferred to the purchaser, and

(iii) if you are acting as a systems integrator for an end user identified in the Purchase Order, this license may be transferred to the end user;

provided, however, that (x) you are then in compliance with your payment obligations under any related Maintenance Agreement then in effect with respect to the Materials, and (y) the transferee provides MVR with:

(A) an unqualified, written acceptance of the terms of this License and any related maintenance agreement fifteen (15) days after the transfer (and, for transfers pursuant to clause (i), identifying the DOD contract for which the Materials are required), and

(B) the name, address, telephone number and e-mail address of an employee of the transferee authorized to communicate with MVR in connection with this License and any related maintenance agreement.

In no case shall any of the Licensed Materials or any related dongle be knowingly or intentionally licensed, transferred or assigned to terrorist sponsored organizations or to organizations which primarily reside within terrorist countries as defined by the United States of America Department of State.

Any transfer made pursuant to this Section must include all of the Licensed Materials and any related dongle and Authentication Files. You shall be solely responsible for any transfer being in compliance with United States export laws and regulations. Upon a transfer in compliance with this Section, the transferee shall thereafter be solely responsible for compliance with the terms of this license agreement (except for any breach or violation which predates the transfer, for which you shall remain responsible) and you will have no further obligation to indemnify MVR hereunder except with respect to your use of the Licensed Materials prior to the transfer. If you are acting as a systems integrator for an end user, you may only use the Licensed Materials to develop, install and support the systems for the end user and not for any other purpose.

Section 7.02 **Transfer to Another Computer or Computer Operating System.** If you wish to transfer the Licensed Materials from one of your computers to another or install a new operating system on your existing computer, the process of re-installing the Licensed Materials will depend upon your method of authentication. If you have a hardware dongle, you may install that dongle in the new computer (or on the existing computer with its new operating system) and install (or re-install) the Licensed Materials on it. If you authenticated by means of Authentication Files, you may (i) transfer the hard drive containing the Licensed Materials from your old computer to your new one, or (ii) reinstall the Software on a new hard drive by requesting permission to effectuate such a transfer in an e-mail to MVR, which may impose a fee for permitting such a transfer. If MVR permits such a transfer, it will instruct you as to how to generate a License ID or .C2V file that will de- install the Licensed Materials from the original computer. You must then send a License ID or .C2V cancellation receipt that will be generated as part of the de-installation process to MVR by e-mail, and MVR shall (upon your payment of any fee required as part of the transfer process) generate a new .A2C or .V2C Authentication File for your use in installing the Licensed Materials on the replacement computer or on the existing computer with the new operating system. **If you do not have a backup copy of your Authentication File and are unable to generate a new License ID or .C2V file (which may happen if, for example, your original computer has become lost, stolen or inoperable) then you may need a replacement Authentication File. See Article 4 for terms applicable for obtaining one.**

Section 7.03 **Transfer by MVR.** MVR may not assign its rights and delegate its duties hereunder without your prior consent, which you may not unreasonably withhold, condition or delay, except that MVR may freely assign its rights and delegate its duties hereunder to a purchase of substantially all of its assets or to a merger partner in a merger in which MVR is not the surviving entity. After such an event, MVR shall have no further obligation to you hereunder except to provide you, or to cause its successor to provide you, with notice of the transaction within 30 days of its occurrence.

Article 8 SUPPLEMENTAL LICENSE TERMS

Section 8.01 **Additional Materials.** In certain cases MVR may from time to time provide Additional Materials at no extra charge to customers with whom MVR has an active maintenance relationship. The license of the Additional Materials granted herein is limited to integration with systems that use the Software and is revocable by MVR without cause in its sole and absolute discretion. Upon request by MVR, you must, within five (5) days of receiving such request, return to MVR all of your physical copies of the Additional Materials, destroy all electronic copies of the Additional Materials in your possession or control, and take such additional actions as MVR may reasonably request to ensure that no copies of the Additional Materials remain in your possession and control. MVR makes no representation or warranty regarding Additional Materials. You are responsible for compliance with any export laws and regulations applicable to any export or deemed export of them.

Section 8.02 **Third Party Materials.** MVR may include among the Licensed Materials software, libraries or databases provided by third parties ("**Third Party Materials**"). Although MVR makes these Third Party Materials available for your convenience, in certain cases you will not be able to use or access specific Third Party Materials with, or as part of, the Software until you have first accepted specific terms and conditions provided by the owner of such Third Party Materials (e.g., by executing a clickwrap or license agreement). Your use of any Third Party Materials provided by MVR will be subject to both the terms of this Agreement and any terms and conditions provided by the owner of such Third Party Materials.

Section 8.03 **U.S. Government Restricted And Limited Rights.** The Materials have been developed entirely at private expense and have been sold and offered for sale to non-governmental customers. The Software is "commercial computer software" as defined in DFARS 252.227-7014 (Feb. 2012) and in FAR 2.101(a), and "restricted computer software" as defined in FAR 27-401 (Oct. 2014) (or any equivalent agency regulation or contract clause). The Materials comprising computer software are provided with the rights set forth in FAR 52.227-19 (November 2007). The Materials comprising technical data are pre-existing technical data developed entirely at private expense and, are provided with the rights described in DFARS 252.227-7015(b) (Feb. 2014). The foregoing grants of Restricted and Limited Rights are only for the benefit of the United States government and its contractors and, in their hands, override any inconsistent restrictions set forth elsewhere in this License Agreement. The Materials may only be sold or transferred to an agency or instrumentality of the United States Government under prime contracts that effectively incorporate restrictions on government use, reproduction, or disclosure no less protective of MVR than the foregoing and any other attempted sale is null and void. Use, reproduction, or disclosure of the Materials by the government or its agents or contractors is subject to the restrictions set forth herein and/or therein, as applicable. Contractor and manufacturer are MVRsimulation Inc., 57 Union Avenue, Sudbury, MA 01776. Use of the Materials by the United States government constitutes acknowledgment of MVR's proprietary rights and of the limited data rights granted in them.

Article 9 TERMINATION

Upon any material violation of any of the provisions of this Agreement, your right to use the Licensed Materials shall automatically terminate without reimbursement and you shall be obligated, within thirty (30) days of receiving a notice of termination of this license from MVR, to return to MVR all of your copies of the Licensed Materials and any hardware keys provided to you in connection therewith, destroy all electronic copies of the Licensed Materials and Authentication Files in your possession or control, and take such additional actions as MVR may reasonably request to ensure that no copies of the Licensed Materials or Authentication Files remain in your possession and control. However, the foregoing shall not apply if the matter constitutes a dispute governed by the Contract Disputes Act of 1978, in which case MVR may pursue its rights in the manner prescribed in that Act and the regulations promulgated thereunder.

Article 10 GENERAL

Section 10.01 **Complete Agreement.** This Agreement and the Purchase Order constitute the entire agreement between you and MVR and supersedes all representations, understandings and other agreements between the parties with respect to the subject matter described herein or therein. In the event of an express inconsistency between this Agreement and the Purchase Order, the inconsistency shall be resolved by giving precedence to the inconsistent terms as follows:

(a) first, to any negotiated rider or addendum to a manually signed version of this Agreement (or any proper termination thereof), regardless of whether it is signed before or after the electronic acceptance of this Agreement (with such documents taking precedence with respect to each other in reverse chronological order of their effective dates);

(b) second, to terms specifically added to the Purchase Order as a result of negotiations between the parties;

(c) third, to the terms of this Agreement; and

(d) fourth to preprinted or standard terms of the Purchase Order that were not modified or included as a result of negotiations between the parties.

To establish that Purchase Order terms were negotiated, a party must produce e-mail or other written correspondence pre-dating the execution of the Purchase Order constituting or acknowledging such negotiations. If this agreement is presented in connection with your installation of an Update of VRSG, Scenario Editor or Terrain Tools, it shall supersede any prior electronic version of the license agreement that You accepted upon an earlier installation of this copy of VRSG, Scenario Editor or Terrain Tools or an Update to it but shall remain subject to the documents identified in clauses (a) and (b) above.

Section 10.02 **Amendment and Waiver.** Failure of a party to enforce any provision of this Agreement does not constitute and should not be construed as a waiver of such provision or the right to enforce such provision. This Agreement may be amended only by a writing executed by both parties or by your electronic acceptance of a more recent version of this license agreement provided to you by MVR.

Section 10.03 **Trademarks.** Nothing contained herein shall give you the right to use any of MVR's trademarks or trade names and you agree not to remove or alter any trademark, trade name, copyright or other proprietary notices, legends, symbols or labels appearing on or in any copies of the Materials.

Section 10.04 **Governing Law; Venue.** This Agreement is governed by the laws of the United States of America and the Commonwealth of Massachusetts, without giving effect to conflict of laws provisions thereof. Any action or proceeding brought by either party against the other arising out of or related to this Agreement shall be brought only (i) in a Massachusetts state court or federal district court for the District of Massachusetts, or, (ii) in the case of a proceeding brought by or against the United States government, the Federal Court of Claims or any successor thereto, and each of MVR and you hereby consent to the personal jurisdiction of such courts. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. If any provision of this Agreement is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable.

Section 10.05 **Customer Suggestions.** MVR considers developing product features requested by customers at no additional cost. MVR retains all rights to these features and may incorporate them in its commercial off- the-shelf products. For further information about MVR's practices in this regard, go to www.mvrsimulation.com/howtobuy/customerfeatures.html.

Section 10.06 **No Corrupt Practices.** You warrant that, in the course of obtaining this license of the Materials, or of selling any products (the "Purchaser Products") into which the Materials are to be integrated, (a) neither you nor your employees or agents have made, offered or promised to make or offer, any payment or any proffer of anything of value, including bribes, either directly or indirectly to any public official, regulatory authority or anyone else for the purpose of influencing, inducing or rewarding any act, omission or decision involving sales of the Materials or the Purchaser Products in order to gain an improper advantage, nor have you authorized or encouraged any other party to do so, and (b) you shall comply, and shall cause your employees and agents to comply, with all applicable anti-corruption and anti-bribery laws and regulations in the course of obtaining purchase orders, requisitions or other authorizations to purchase the Materials and the Purchaser Products. You shall notify MVR immediately upon becoming aware of any breach of your obligations under this Section.

To initiate a support ticket, please contact MVR at support@mvrsimulation.com.