VR-Link for Unity

Users Guide

VT MÄK
A company of VT Systems

VR-Link for Unity

Users Guide

VT MÄK
A company of VT Systems

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

VT MÄK
150 Cambridge Park Drive, 3rd Floor
Cambridge, MA 02140 USA

Voice: 617-876-8085
Fax: 617-876-9208

info@mak.com

www.mak.com

Revision VRU-1.2-1-140408

Contents

Preface

Chapter 1. VR-Link for Unity

Chapter 2. Installing VR-Link for Unity

MÄK Products Glossary

Preface

This guide is a brief introduction to VR-Link for Unity to get you started. For developer documentation, please see the online API documentation.

How this Manual is Organized

The chapters are organized as follows:

Chapter 1, *Introduction*, describes the VR-Link for Unity.

Chapter 2, *Installing and Configuring VR-Link*, explains how to install VR-Link for Unity and other required and optional software.

The *MÄK Products Glossary* defines important terms.

MÄK Products

The VR-Link for Unity is a member of the VT MÄK line of software products designed to streamline the process of developing and using networked simulated environments. The VT MÄK product line includes the following:

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

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

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

  You can create and view local entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. VR-Forces also functions as a plan view display for viewing remote entities taking part in an exercise. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.
- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes three end-user applications (VR-Vantage Stealth, VR-Vantage PVD, and VR-Vantage IG) and the VR-Vantage Toolkit.
  - VR-Vantage Stealth displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to entities so that it moves as they do.
  - VR-Vantage IG is a configurable desktop image generator (IG) for out the window (OTW) scenes and remote camera views. It has most of the features of the Stealth, but is optimized for its IG function.
  - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MÄK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MÄK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.

- VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- VR-inTerra. VR-inTerra is a C++ API for adding terrain agility to applications. It can load, page, or stream terrain from a wide variety of formats or sources into a single, consistent run-time representation, consisting of a collision graph and vector network.
- DI-Guy™. The DI-Guy product line is a set of software tools for real-time human visualization, simulation, and artificial intelligence. Every DI-Guy software offering comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy enables the easy creation of crowds and individuals who are terrain aware, autonomous, and react intelligently to ongoing events. Save time, money and create outstanding simulations with DI-Guy. The DI-Guy product line includes the following products:
  - The DI-Guy SDK. Embed the DI-Guy library in your real-time application and populate your world with lifelike human characters.
  - DI-Guy Scenario™. Author and visualize human performances in a rich, user-friendly graphical environment. Use DI-Guy Scenario as an end visualization application or save scenarios and load them into your DI-Guy SDK enabled application.
  - DI-Guy AI. Generate crowds of autonomous characters to quickly populate your worlds with hundreds and thousands of terrain-aware, collision avoiding DI-Guys. Used as a module on top of DI-Guy Scenario and DI-Guy SDK.
  - Expressive Faces Module. Enable DI-Guy characters to have faces that display emotion, eyes that look in directions and blink, and lips that sync to sound files.
  - DI-Guy Motion Editor. Create or customize motions to your particular needs in an easy-to-use graphical application.

How to Contact Us

For VR-Link for Unity technical support, information about upgrades, and information about other MÄK products, you can contact us in the following ways:

Telephone

Call or fax us at:	Voice:	617-876-8085 (extension 3 for support)
	Fax:	617-876-9208

E-mail

Sales and upgrade information:	info@mak.com
Technical support:	support@mak.com
VR-Vantage support:	vrv-support@mak.com

Internet

MÄK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses.html
Product version and platform information:	www.mak.com/support/product-versions.html
For the free, unlicensed MÄK RTI:	www.mak.com/resources/bonus-material/cat_view/16-bonus-materials/24-mak-high-performance-rti.html
MÄK Community Forum:	www.mak.com/community-forum/1-forum.html

Post

Send postal correspondence to:	VT MÄK 150 Cambridge Park Drive, 3rd Floor Cambridge, MA, USA 02140

When requesting support, please tell us the product you are using, the version, and the platform on which you are running.

Document Conventions

This manual uses the following typographic conventions:

`Monospaced`	Indicates commands or values you enter.
`Monospaced Bold`	Indicates a key on the keyboard.
`Monospaced Italic`	Indicates command variables that you replace with appropriate values.
Blue text	A hypertext link to another location in this manual or another manual in the documentation set.
`{ }`	Indicates required arguments.
`[ ]`	Indicates optional arguments.
|	Separates options in a command where only one option may be chosen at a time.
`( | )`	In command syntax, indicates equivalent alternatives for a command-line option, for example, `(-h | --help)`.
/	Indicates a directory. Since MÄK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
Italic	Indicates a file name, pathname, or a class name.
sans Serif	Indicates a parameter or argument.
➤	Indicates a one-step procedure.
Menu → Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose **File → Save**.
	Click the icon to run a tutorial video in the default browser.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

Directory names are preceded with dot and slash characters that show their position with respect to the VR-Link for Unity home directory. For example, the directory *vrl1.2/doc* appears in the text as *./doc*.

DI-Guy Conventions

Table 1-1 lists the conventions used for entity coordinates in DI-Guy documentation.

Table 1-1: Coordinate system conventions

Convention	Indicates
XYZ	X = forward, Y = to the left Z = up
Yaw, Roll, Pitch	Orientation is applied in the order YRP. Yaw = rz – orientation about the vertical axis Roll = rx – orientation about the fore-aft axis Pitch = ry – orientation nose down, nose up

Mouse Button Naming Conventions

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

Third Party Licenses

MÄK software products may use code from third parties. This section contains the license documentation required by these third parties.

Boost License

VR-Link, and all MÄK software that uses VR-Link uses some code that is distributed under the Boost License. All header files that contain Boost code are properly attributed. The Boost web site is: www.boost.org.

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

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

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

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libXML and libICONV

VR-Link and all MÄK software that uses VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MÄK Products. MÄK has made no modifications to these libraries. For more information about these libraries please see the following web sites:

- The LGPL license is available at: http://www.gnu.org/licenses/lgpl.html.
- Information about IconV is at: http://www.gnu.org/software/libiconv/.
- Information about LibXML is at: http://xmlsoft.org/.

Lua

Some MÄK products use the Lua programming language (www.lua.org). Its license is as follows:

Copyright © 1994–2012 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1. VR-Link for Unity

VR-Link for Unity is a Unity Asset package that provides DIS/HLA interoperability for Unity.

1.1. Introduction

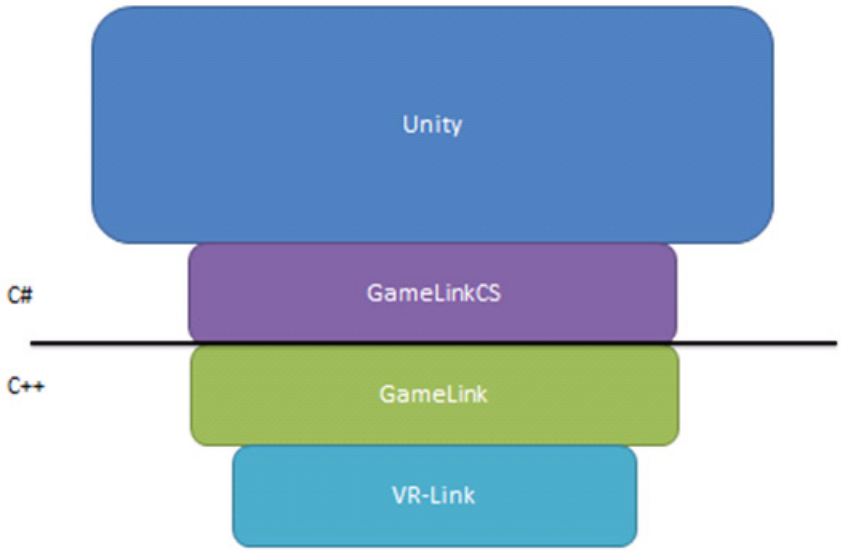
VR-Link for Unity is a Unity Asset package that provides DIS/HLA interoperability for Unity. It enables your Unity based application to participate in a DIS or HLA exercise. We support DIS, HLA 1.3, 1516, and 1516-Evolved. We support several HLA FOMs (Federation Object Models) through VR-Link's FOM Mapper ability. VR-Link for Unity is based on MAK's existing VR-Link product - our industry-leading toolkit for simulation interoperability (http://www.mak.com

1.2. Architecture Overview

VR-Link for Unity is composed of several layers. The bulk of the heavy lifting is accomplished in a C++ library called Game Link, which adds Unity-specific coordinate conversions and entity management logic on top of our standard VR-Link toolkit. Above the C++ Game Link library is a set of C# bindings called GameLinkCS, which allows the Unity application to access the power of VR-Link and GameLink directly from C#, JavaScript, etc. GameLinkCS communicates with the underlying C++ libraries through a message-passing mechanism.

The GameLinkCS layer is responsible for passing messages between Unity and VR-Link:

- For messages coming from DIS/HLA, GameLinkCS receives messages from VR-Link, and stores current state in Unity game object components. It automatically updates transforms to move objects around in the scene, based on VR-Link's dead-reckoning, smoothing, and ground clamping (if necessary). Users can extend the built-in message handler to fit the needs of their game design. For events (e.g. Fire and Detonate), GameLinkCS provides a mechanism to register callbacks or custom handlers.
- For objects owned by Unity, GameLinkCS provides a "Publish" component which is responsible for passing current state across the language boundary to VR-Link. VR-Link then publishes your objects to the DIS/HLA network. For events/interactions, GameLinkCS provides a C# API to fill out and send the appropriate messages.

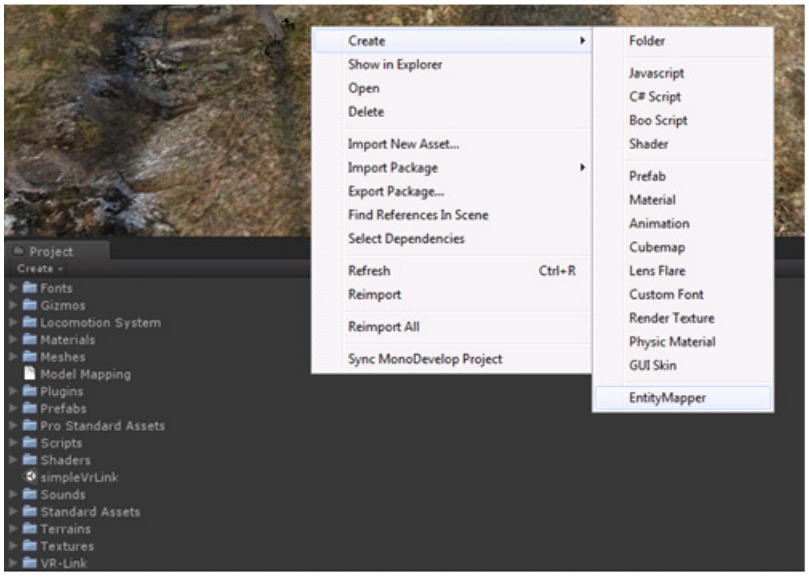


1.3. Integrating VR-Link for Unity into an Application

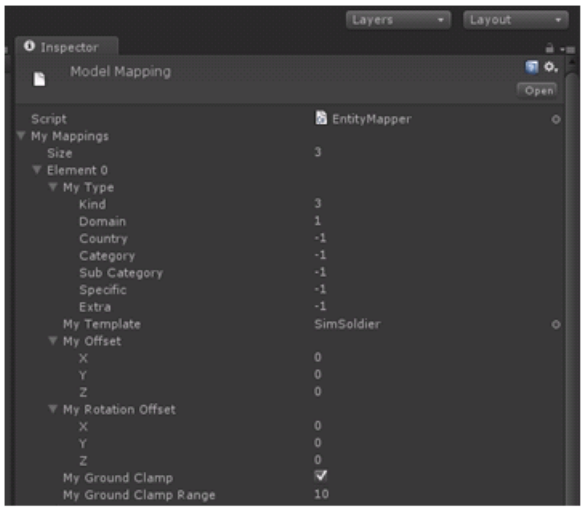
To add VR-Link for Unity to your application, simply import the Asset. This will add the VR-Link directory to your Assets folder. The VR-Link folder contains all the C# scripts for interfacing with the C++ layer as well as the classes for managing the simulated entity behaviors and terrain referencing. It will also add the VR-Link DLLs and gamelink subfolder to the Plugins directory. These are the C++ libraries that are used.

1.4. Model Mapping

One of the first components that needs to be created when developing with VR-Link for Unity is the Model Mapper. component is added to the create menu and will be available for use once VR-Link for Unity is installed. The Model Mapper object allows you to map DIS type enumerations to a Unity Prefab (or template) object. When a discovery message is received for this particular entity type, the simulation manager (see below) will create a new Game Object based on the assigned template. The Model Mapper class is used for both entities and for detonations. It is possible to have separate instance for entities and detonations or use the same instance of the Model Mapper for both.



The model map is a list of ModelMap classes. By adjusting the size of the list you are able to add or remove mappings. The Model Mapper allows for a position and rotation offset to be used with the entity; to set whether or not the entity should be ground clamped; and to set the range over which the ground clamping should take place.



This list allows for wildcards (-1) in the entity type.

Model maps may be loaded to and from disk. Files are saved in XML format.

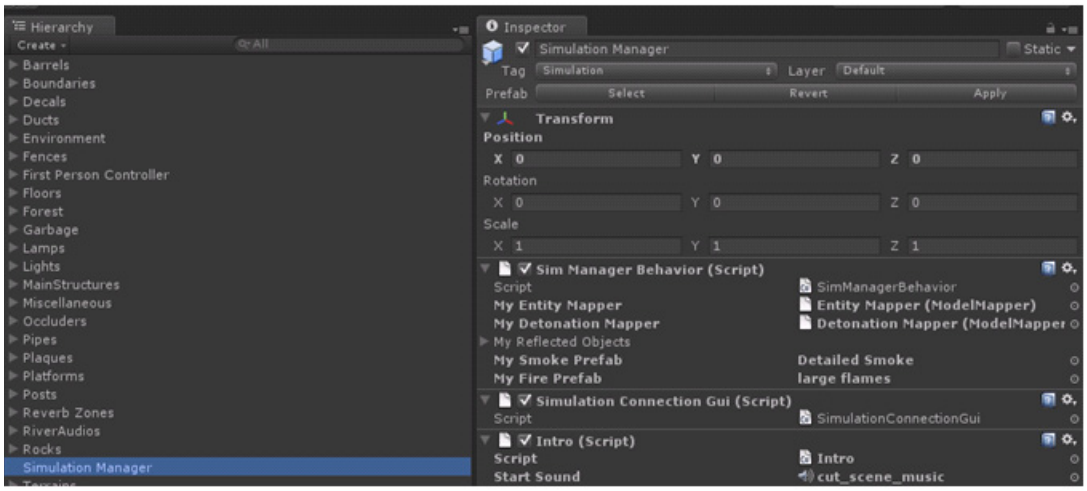
! Any references to prefabs in your model map need to have the prefab in the "Resources" directory in your project, otherwise, they will not be able to be found again during loading of a model mapping file.

1.5. Simulation Manager

The Simulation Manager behavior is the heart of VR-Link for Unity within C#. It provides the C# API to VR-Link's libraries for connecting to an exercise, handling message dispatch and creating entities. There is a prefab Simulation Manager Game Object provided with VR-Link for Unity.

After creating the Model Mapper, we create a Simulation Manager by creating a Unity empty Game Object and adding a Simulation Manager Behavior (script) to it. This script will set the tag of the attached entity to "Simulation" on startup.

The Simulation Manager behavior needs a Model Mapper for entities and a Model Mapper for detonations as inputs, so drag and drop the Model Mapper that is appropriate for the inputs to the slots on the script.



The simulation manager also needs to know the smoke and fire prefabs to use to initialize any damage that models may need to display.

1.6. Initiating a Connection

Once you create the Simulation Manager, you will need to create a simulation connection. This is done by sending an InitConnection message through the Simulation Manager. The Simulation Manager also has public Connect() functions to make this easier. This function is overloaded for DIS and HLA. The appropriate parameters need to be set. Only one connection can be active at a time, but it is possible to switch between DIS and HLA without shutting down the Unity Application. Disconnecting from the distributed simulation is done by calling shutdown() on the Simulation Manager, or destroying the Game Object which holds the Simulation Manager behavior.

1.7. Reflected Game Objects

As the Unity application runs, it will automatically update the Simulation Manager behavior, which in turn calls tick() on VR-Link. This allows for VR-Link to discover and update any entities from the simulation and notify Unity of new entities. When a new entity is discovered, a discovery message is sent to the Simulation Manager, which looks up in its Model Map the appropriate type of Unity Prefab to use as a template for creating a new Game Object. After a Game Object is created, the Simulation Manager gives the new Game Object a Reflected Behavior if one does not already exist, and registers that Gamer Object to receive Entity State update messages for the corresponding DIS/HLA entity.

Entity State messages are already preprocessed by VR-Link for Unity (on the C++ side) to convert coordinates, orientations, and vectors (velocity and acceleration) into the Unity coordinate system before the updates are pushed across to C#. When the Reflected Behavior receives an Entity State update, it pushes the entity's new position and orientation into the Game Object's transform. Entity orientation is presented as heading, pitch, and roll of the entity. Entity velocity and acceleration is a vector in the Unity coordinate system. See the Terrain setup section for more information about coordinate systems and conversion.

Users will want to observe the Entity State message that is saved as a component of each Game Object's Reflected Behavior for changes, in order to update any animations, particle systems, lights, etc. on their Game Object.

1.8. Publishing Game Objects

Publishing Unity Game Objects as DIS/HLA entities is done by attaching a Publish Behavior (script) to a particular Game Object. Attributes such as Entity Type and Marking Text can be edited in the Unity editor, or set programmatically. The Publish Behavior script also contains an Entity State message that is sent to VR-Link every frame. The Publish behavior automatically pulls the current location and orientation from the Game Object's transform, and updates the Entity State message with this information. Publishing any additional state changes needs to be done by the user, by setting values in the Entity State message each frame.

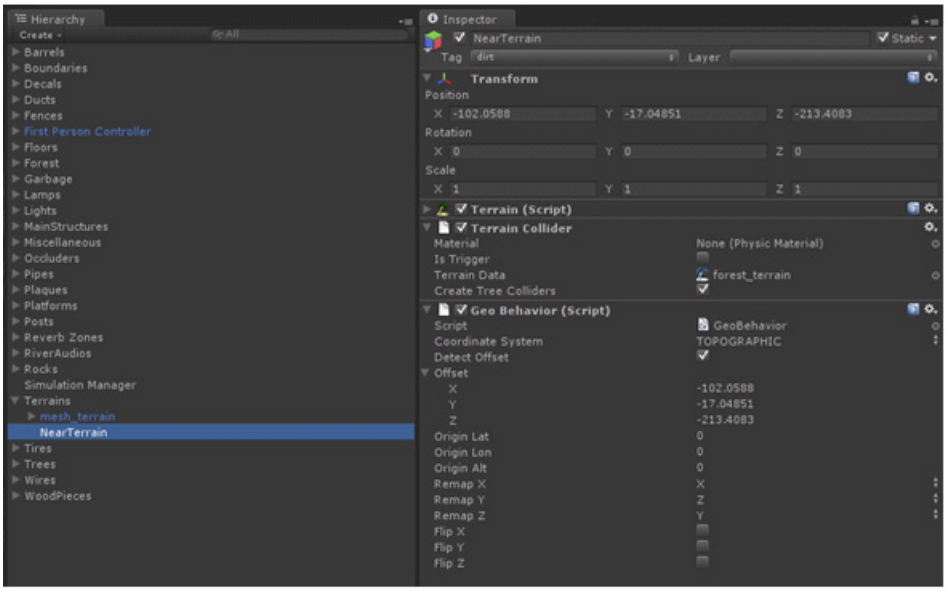
Position, velocity, acceleration and orientation are transmitted in the Unity coordinate system and converted to the appropriate DIS/HLA coordinate representation in the C++ side and should be transparent to the Unity developer.

1.9. Terrain and Coordinate Systems

The Unity Terrain Object needs to have a Geospecific Behavior (script) attached as a component. This behavior allows for VR-Link for Unity to properly translate between the DIS/HLA geocentric coordinate system, and a coordinate system Unity is able to use.

In this Geospecific Behavior, there is an option to select the coordinate system. The two options are TOPOGRAPHIC and PASSTHROUGH.

When TOPOGRAPHIC is selected it is assumed that the Unity terrain is a flat earth projection of a geospecific place. The origin Latitude, Longitude and Altitude fields are the topographic reference used to determine the geospecific location of the of the origin of the Unity terrain. These are passed to VR-Link so that VR-Link can convert between DIS/HLA geocentric coordinates, and the local Unity topographic system. The axis remapping and axis flipping fields have no effect in TOPOGRAPHIC mode.

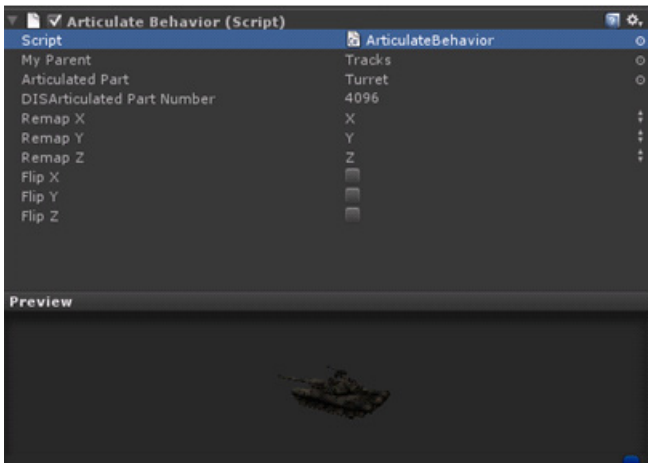


PASSTHROUGH is the option to use when your DIS/HLA applications are not using the default DIS/HLA geocentric coordinate system - for example, when DIS/HLA application have already been designed to work with a Unity coordinate system. In this mode VR-Link for Unity will pass coordinates from HLA/DIS Entity State updates directly through to Unity, without performing any coordinate conversion. With PASSTHROUGH, it is also possible to add an offset, remap or flip an axis value to account for different ways of representing coordinates within a given system. It is also possible to automatically detect the offset (transform) of the Unity terrain's origin from the origin of the Unity coordinate system, and pass that value to VR-Link, so that VR-Link can account for the offset. This will override any values put in the offset dialog.

Future versions of this behavior will have a custom editor dialog to show only the fields that impact the current mode of operation and allow for runtime changes. Currently this only initializes VR-Link on startup.

1.10. Articulated Parts

Articulated parts from reflected entities are supported through an ArticulatedPart script. Attaching this script to prefab models and configuring it will allow reflected entities to move their articulated parts based on network data.



The script needs the articulated part that it is responsible for moving, the parent that the part is attached to and the DIS Articulated Part ID that it should listen for. It is possible to remap axis and flip axis data in the event that the model used does not follow the same coordinate system as the articulated parts data. Please see the DIS standard and SISO Enumerations document for more information about the Articulated Part coordinate system and DIS Articulated Part numbers.

1.11. Extensibility and Plugins

The VR-Link for Unity C++ architecture is built in a manner that allows for the User to extend its capability through the use of plugins. For example, users are able to add support for additional HLA Objects and Interactions, additional DIS PDUs, or addition attributes of existing objects.

Please see the Developers Guide and class documentation for more information about how to use the SDK.

1.12. Licensing

VR-Link for Unity has two licensing schemes. When running in the editor, it will attempt to check out a FlexLM license from a license server, or local license file. If VR-Link for Unity running in the editor cannot find a license, it will run in trial mode with a limitation of only running for 10 minutes. Applications built with Unity that use the VR-Link for Unity package will need to be digitally signed by the *signer.exe* application, which is found in the bin directory. The *signer.exe* application is also license managed just as other VT MAK products are using FlexLM.

To digitally sign an application, run the *signer.exe* application from the command line with the path to your applications data directory as the parameter to the *signer.exe* application. This will sign the DLLs used for VR-Link for Unity for use with this specific game. If you make changes to your game, then you will need to resign the application. Applications that are not signed will run in a trial mode for 10 minutes.

1.13. Diagnostics

VR-Link for Unity will produce a log file in the project directory every run with the name "gamelink.log." Please see this log during development to identity any errors.

2. Installing VR-Link for Unity

This chapter covers the following topics:

2.1. Installing and Setting Up the MÄK License Manager

Before you can use a MÄK product, you must obtain a valid license file, install the MÄK License Manager, and configure the license server and client machines. The License Manager uses a client-server architecture, so you do not need to install the License Manager on every computer on which you install MÄK products. You only need to install it on the computer that you will use as the license server.

If you have already installed the License Manager for another MÄK product, you do not have to install it again. You just need to make sure you have licenses for your newly installed products.

The License Manager installer is included on MÄK installation media. It is separate from the product installers. You can download the installers from our web site at http://www.mak.com/license-support.html or you can download directly from:

- Windows: ftp://ftp.mak.com/out/MAKLicenseManager-win-setup.exe (32 bit systems)
  ftp://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe (64 bit systems)
- Linux: ftp://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz

Complete installation and configuration instructions are included with the License Manager installer. Instructions are also available at the license support page.

Some customers use dongle licenses instead of running a license server. Instructions for using dongles are in the License Manager documentation.

2.1.1. Specifying the License Server

The first time you run a MÄK application on a particular computer, the License Setup dialog box opens (Figure 2-1). It prompts you to enter the hostname of the license server and optionally, a port number.

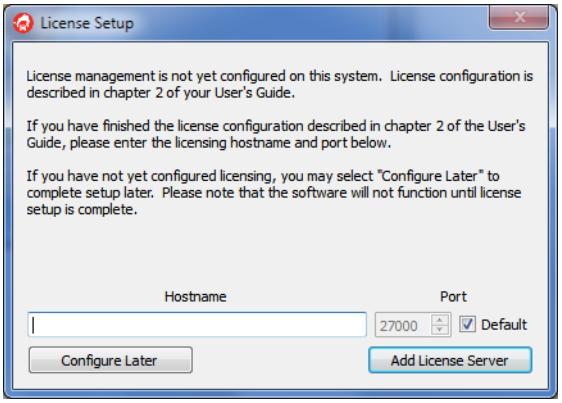


Figure 2-1. License Setup dialog box

If you do not know the hostname of the license server, click Configure Later. When you have the hostname, you can start the application again and complete the dialog box. You will not be able to run any MÄK applications until you set up license management.

If you know the hostname, type it in the Hostname box. Then click Add License Server. The application will start.

- If you are running MÄK products on the license server machine, it is also a client, so you must specify the license server on that machine too.
- If you change the license server, the saved configuration will no longer be valid and the License Setup dialog box will open the next time you start a MÄK application.
- You can clear the saved license configuration by deleting the cache file. On Windows XP, it is *C:\Documents and Settings\user_name\Application Data\MAK\license<n>.xml*. On Windows 7, it is *C:\Users\user_name\AppData\Roaming\MAK\licenses<n>.xml*. (The *AppData* directory is hidden by default.) On Linux, it is *.mak/license<n>.xml*. (There may be more than one cache file, for example, *licenses1.xml* and *licenses2.xml*.)

The MAKLMGRD_LICENSE_FILE Environment Variable

As an alternative to using the License Setup procedure described in the previous section, you can configure the license server in an environment variable. The MAKLMGRD_LICENSE_FILE environment variable identifies the server machine. If you set this environment variable, it overrides the settings stored by the License Setup procedure.

The syntax for the environment variable is: `@Server_name`. For example, if the server machine is oak, set the environment variable to `@oak`.

The following sections explain how to set environment variables on the different platforms that MÄK products run on.

Windows

To add the MAKLMGRD_LICENSE_FILE in Windows:

1. Open the Control Panel.
2. On Windows 7, click System and Security. On Windows XP, click Performance and Maintenance.
3. Click System.
4. In the sidebar menu, click the Advanced System Settings. The System Properties dialog box opens.
5. Click the Environment Variables button. The Environment Variables dialog box opens.
6. Click the New button. The New System Variable dialog box opens.
7. In the Variable Name field, enter MAKLMGRD_LICENSE_FILE.
8. In the Variable Value field, enter `@server_name`, where `server_name` is the name of the license server.
9. Click OK to back out of each dialog box and set the variable.

Different versions of Windows have slightly different wording of the various Control Panel options.

Linux

On Linux, you set environment variables in your *.cshrc* (or equivalent startup file). Set the variable similarly to the following example:

```
setenv MAKLMGRD_LICENSE_FILE @oak
```

If you are using the sh or bash shells, you set environment variables in your *.profile* file (or *.bashrc*). Set the variable similarly to the following example:

```
MAKLMGRD_LICENSE_FILE=@oak
export MAKLMGRD_LICENSE_FILE
```

Do not put spaces around the equal (=) sign.

You are ready to run the license server and use your new licenses or MÄK products.

2.2. Installing an RTI

An RTI is a software library (and perhaps supporting executables) that implements an HLA Interface Specification. In HLA, applications exchange FOM data through RTI calls, which means that all HLA applications need to use an RTI.

Because of differences in the low-level network mechanisms used by different RTI implementations (which include, but are not limited to packet layout), applications that want to interoperate in the same federation execution must use the same RTI implementation.

Because RTIs are usually provided as dynamic libraries that implement a fixed API, a federation can often switch from one RTI implementation to another between runs (without even recompiling the applications), but during each run, all participants must agree on which RTI to use, much as they must also agree on which FOM to use.

For the most recent information about the RTI versions supported by MÄK products, please see the release notes for your MÄK application.

2.2.1. Installing the MÄK RTI

To install the MÄK RTI, follow the instructions in Chapter 2 of *MÄK RTI Users Guide*.

Configuring Your System to Use the MÄK RTI

The RTI dynamic libraries must be located somewhere on your dynamic library search path. The path is specified in the PATH environment variable.

The MÄK RTI needs to know where to find the following configuration files:

- The federation configuration file (*.fed*, *.fdd*, or *.xml*) for your federation execution (required).
- The RID file (*rid.mtl*) (optional).

Put the configuration files in the directory from which you are running, or set the environment variable RTI_CONFIG to the directory that contains them.

Running Applications with the MÄK RTI

In many cases, you do not need an RTI server, such as the rtiexec, or a central application to use the MÄK RTI, however you can run the rtiexec if you want to. (It is required to use certain features of the MÄK RTI.) Be sure the license server is running, then start the applications, and they should be able to communicate. For more information, please see your RTI documentation.

2.3. Configuring the Network for DIS Applications

In a typical DIS exercise, VR-Link for Unity applications communicate with other DIS applications using broadcast UDP/IP over a local area network. In such a configuration:

- All applications in the exercise use the same IP broadcast address, and each application's host machine has a network interface device configured for that address.
- All machines must share a common netmask.
- All participating applications must use the same UDP port.

Typically, machines on a LAN will already have the correct broadcast addresses, so VR-Link for Unity applications will be able to communicate with each other immediately. However, if applications are unable to achieve two-way communication, the network devices may be improperly configured.

2.3.1. Broadcast Address and Netmask (UNIX)

Network devices should be configured so that all machines agree on the netmask and on the part of the network address (inet) covered by the netmask. On UNIX-based systems, you can use the `ifconfig` command to examine and set the configuration of network devices. For example, to set the netmask use:

```
ifconfig device_name netmask 0xFFFF0000
```

Some machines have more than one network device capable of broadcasting. Unless your application is designed to specify a specific device or address (which is possible using VR-Link for Unity routines), a VR-Link for Unity application will use the broadcast address of the first device listed in the interface table by default.

Please see your system administrator if you need help with network issues.

MÄK Products Glossary

This glossary defines terms used in MÄK product documentation.

absolute dead reckoning

A method of dead-reckoning in which an application uses the time stamps contained within state updates if the sender is using absolute time stamping. Contrast with relative dead reckoning.

aggregate

The combination of individual entities, aggregates, or both into a single object. For example, an organizational group such as a platoon.

AMO

Application Management Object (AMO).

Application Management Object (AMO)

An AMO is a standard TENA object that is supposed to be used by every TENA application. It gives other applications information about the local application. The VR-Link exercise connection, by default, publishes an AMO object, updates it at a specified rate (the rate is part of the AMO's state repository; the default is 30 seconds), and updates the number of objects published and proxies (TENA reflected objects) held.

API

Application Programming Interface.

articulated part

A part of an entity that is capable of movement relative to the entity, such as a turret or gun.

attached part

An object that is attached to another object, such as a rocket attached to a jet.

body coordinate frame

A coordinate framework centered on an entity. To represent the orientation of an entity, an application uses Euler angles or a rotation matrix that rotates from a reference frame to the body coordinate frame.

channel

A channel renders the view of an observer in a 3D visualization application.

clipping

A feature that prevents the display of terrain and objects or parts of objects, that are closer than or farther than specific distances from the eyepoint.

clipping planes

The range of distances from the eyepoint (near clipping and far clipping) in which an application clips objects.

culling

The process of discarding database objects which are not within view, and therefore need not be rendered and displayed.

Cartesian coordinates

A system for indicating location by means of three planes intersecting at right angles to one another at the origin.

dead-reckoning

A process by which an application calculates the expected location of an entity during periods between state updates, based on velocity, acceleration, and rotational velocity.

Defense Modeling and Simulation Office

The executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provides a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. Currently the DMSO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness.

DIS

Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

display engine

An object in an application that can render 3D data.

Distributed Interactive Simulation

The DIS protocol is a set of standards that govern how participating applications share information about the virtual world. The protocol specifies a set of packets, called protocol data units (PDUs), that communicate this information. Each PDU identifies the sender and contains other information depending on the PDU type. The DIS protocol also specifies when and how frequently PDUs are sent.

The DIS protocol has been superseded by the High-Level Architecture for use by the Department of Defense.

DMSO

Defense Modeling and Simulation Office.

emitter

A simulated device on an entity that emits electromagnetic radiation, for example, radar.

entity

An element in a simulation, such as a vehicle or a person, that is represented in the simulation through the issuance of state messages.

entity maintenance

A process by which the MÄK Data Logger compensates for discontinuities following a time jump. The Logger sends out interim state messages for entities that were present before the time jump so that they do not time out before the next update message arrives.

entity-type

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

The components of an entity-type are:

- Entity kind
- Domain
- Country
- Category
- Subcategory
- Specific
- Extra.

environmental process

According to the IEEE 1278.1a specification (DIS), environmental process PDUs communicate simple environment variables, small scale environmental updates, and embedded processes.

Euler angles

A set of three angles used to describe the orientation of an entity as a set of three successive rotations about three different orthogonal axes (x, y, and z). These angles specify successive rotations needed to transform from a reference coordinate system to the entity's body coordinate system.

event

An interaction between objects or between an object and the terrain, such as firing of a munition, or a collision of entities.

execution

The TENA term for one or more interacting simulation applications.

Execution Manager

An Execution Manager governs a TENA execution for applications joining, resigning, or changing subscriptions to that execution.

exercise

The DIS term for a one or more interacting simulation applications. Compare to federation execution in HLA.

exercise connection

The object (*DtExerciseConn*) through which a VR-Link application connects to the network. State messages are sent through the exercise connection and information about remote entities is received through the exercise connection. (All MÄK products are VR-Link applications.)

exercise ID

A numeric identification for a DIS simulation exercise.

FED file

Federation Execution Data file. The Federation Execution Data is the subset of FOM data needed and read by the RTI. VR-Link applications also read the FED file.

federate

A connection to the RTI. Typically a single simulation application can be thought of as a federate.

federation

A group of HLA federates capable of playing in the same federation execution.

Federation Object Model

Defines the data content of a federation execution.

federation execution

The federation execution represents the actual operation, over time, of a subset of the federates and the RTI initialization data taken from a particular federation. A federation execution is the logical equivalent of a DIS simulation exercise.

field of view

Field of view controls the perspective of the eyepoint.

A wide field of view creates an effect like that of a wide-angle camera lens. Objects appear smaller and farther away from the eyepoint, since the eyepoint coverage spans a wider area. Depth become exaggerated.

A narrow field of view creates an effect like that of a telephoto lens. Objects appear larger and closer to the eyepoint, and the overall scene depth appears flattened. The distances between objects appears compressed.

filter range

A setting that prevents distant entities from being processed while allowing distant terrain to appear normally.

FOM

Federation Object Model.

frame rate

The rate at which the application displays updated images.

ground clamping

A process by which the a 3D visualization application keeps an entity anchored to the surface of its terrain database, regardless of the altitude data contained in its entity state message.

geocentric coordinates

A coordinate system calculated with respect to the earth's center. The origin of the geocentric coordinate system is the center of the earth. The positive X-axis passes through the prime meridian at the equator; the positive Y-axis passes through 90 degrees east longitude at the equator; and the positive Z-axis passes though the north pole.

geodetic coordinates

A coordinate system in which position is determined relative to a reference ellipsoid, such as the surface of the earth at sea level. In MÄK applications, geodetic coordinates consist of latitude and longitude in radians, and altitude in meters above the reference ellipsoid.

gridded data

Data that has been processed in a rectangular array of points, in X, Y or latitude/longitude, at which single data values define a two dimensional function. According to the IEEE 1278.1a specification, gridded data transmits information about large-scale or high-fidelity spatially and temporally varying ambient fields and about environmental processes and features.

guise

An alternative entity type used to display an object depending on the force ID. For example, a tank could look like an M1A1 to friendly forces and a T72 to hostile forces.

head-up display

A set of indicators and readouts superimposed onto a graphics display. Also called an overlay.

heartbeat

In DIS, the frequency with which current PDUs are sent to the network regardless of whether or not the entity's state has changed.

High-Level Architecture

The High Level Architecture (HLA) for simulations is a U. S. Department of Defense (DOD)-wide initiative to provide an architecture to support interoperability and reuse of simulations. The HLA supersedes DIS for the DOD.

HLA

High-Level Architecture.

interaction

A message describing a simulation event. An interaction describes an event, it does not update an object's state.

Logger control PDUs

A set of DIS PDUs or HLA interactions that let you control the MÄK Logger remotely.

logical range

A suite of TENA Resources, sharing a common object model, that work together for a given range event. Similar in concept to the HLA Federation.

Local RTI Component

The libraries on a computer that implement an RTI. A federate communicates with the HLA network through the LRC.

Logical Range Object Model

The data definition file for TENA exercises. It contains the set of object and message definitions that may be used in the exercise.

LRC

Local RTI Component

LROM

Logical Range Object Model.

MÄK Technologies Lisp

An adaptation of the Lisp language used in configuration files for MÄK products.

message

A general term used to refer to DIS PDUs and HLA interactions and state updates. In TENA, a message is similar to an HLA interaction.

MTL

MÄK Technologies Lisp.

Network Naming Service

Acts as a registry for TENA Execution Managers.

NNS

Network Naming Service.

orientation clamping

Adjusts the pitch and roll of an entity so that it appears properly seated when the terrain is inclined. For example, if a tank is moving horizontally across the face of a hill, orientation clamping prevents the tank from appearing level, and therefore, partially embedded into the hillside. Used with ground clamping.

object

An element in a simulation that has persistence, as opposed to an interaction, which is a transient element.

object handle

An integer that an application uses to identify a particular object in RTI service calls. An object handle is meaningful only to a particular federate. The same object can be known to different federates by different object handles.

object name

A character string that can be used to identify an object. In HLA, the object name is known to the RTI, and the RTI provides functions to find out an object's name, given its handle, and vice versa. Object names can be chosen by applications that register the objects with the RTI, however if you do not want to choose names for objects, the RTI will assign names for you.

PDU

Protocol Data Unit.

Protocol Data Unit

A unit of data message (packet) that is passed on a network between DIS simulation applications.

proxy

In TENA, a reflected object (Stateful Distributed Object).

radio transmitter

A simulated device on an entity, capable of transmitting radio communications.

radio receiver

A simulated device on an entity, capable of receiving radio communications.

range resource

A participant in a TENA execution (similar in concept to the HLA federate).

Realtime Platform Reference FOM

An HLA reference FOM based on the DIS protocol.

recording

A Data Logger file that stores a history of the interactions in a simulation for playback.

reflected entity list

A list of entities simulated by remote applications (federates in HLA) and about which the local simulation has received information over the network.

reference FOM

A FOM designed to be used as a whole, or with modification, by a wide variety of similar federation executions.

relative dead reckoning

A method of dead reckoning in which an application approximates the location of an object based on the local time of receipt of the last state update message.

remote display engine

A display engine running in an application that is separate from the master application, and which is controlled by the master application.

RTI Initialization Data (RID)

The initialization data required by the RTI for operation.

Data required by an RTI during initialization, independent of the FOM being used. RID data is usually dependent on a specific implementation of the RTI.

RPR FOM

Realtime Platform Reference FOM.

RTI

Run-Time Infrastructure.

Run-Time Infrastructure

A library and other supporting software that implements the HLA interface specification. All federates communicate with one another in an HLA environment through RTI functions.

SDO

Stateful Distributed Object.

servant

A published object (SDO) in TENA.

Simulation Interoperability Standards Organization

A group that seeks to promote modeling and simulation interoperability and reuse for the benefit of diverse M&S communities, including developers, procurers, and users, world-wide.

simulation time

In VR-Forces, simulation time is used in dead-reckoning of remote entities and thresholding of local entities. Typically, simulation time is set once during each iteration of the application's main simulation loop so that all entities are dead-reckoned based on the same value of current time.

SISO

Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

A method of ensuring that transitions from an entity's dead-reckoned position to its actual position are not so abrupt as to be visually disconcerting.

state

The current status of an object, including location, direction of movement, extent of damage, and so on.

Stateful Distributed Object

An object in a TENA exercise.

tape

An alternative term for a Logger recording. Not a physical magnetic tape.

TENA

Test and Training Enabling Architecture.

TENA Middleware

"The software that links range resources together into a logical range. The "TENA Middleware" is the software component that is in between (that is, in the middle of) range resources during a logical range execution." – from www.tena-sda.org

terrain following

In a 3D visualization applicaiton, causes the observer (eyepoint) to maintain a constant distance above the terrain surface. The observer's height changes in tandem with the peaks and valleys as it passes over the geography.

Test and Training Enabling Architecture

Test and Training Enabling Architecture (TENA) is an interoperability architecture used to pass data in the form of object updates and messages (like HLA objects and interactions) from one application to another.

timeout

The period of time in which an application continues to display an entity after that entity's update messages have stopped appearing on the network. Time-outs are usually not used in HLA because there is no set frequency (no heartbeat) for transmitting messages.

timescale

A factor by which time is accelerated or slowed during playback of a Data Logger recording.

topographic coordinates

A right-handed Cartesian coordinate system whose X-Y plane is tangent to the earth's surface at the origin, with the positive X axis pointing north, the positive Y axis pointing east, and the positive Z axis pointing down. There are an infinite number of topographic coordinate systems – one for each point on the earth's surface.

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

A method used by to smooth positional discontinuities that could occur when new state updates arrive.

UDP port

A network channel through which an application sends and receives data for DIS exercises.

Universal Transverse Mercator

In general, a non-cartesian coordinate system in which the X, Y, and Z correspond to easting (nearly east), northing (nearly north), and height above an ellipsoid that approximates the surface of the earth.

UTM coordinates

Universal Transverse Mercator coordinates.

view control messages

A set of programmatic messages that let you control the VR-Vantage remotely.

view floor

In MÄK Stealth 6.x and earlier, a minimum height above the terrain in Absolute or Track mode, below which the eyepoint is not allowed to go. The view floor is measured relative to the terrain surface.

Link - Simulate - Visualize

VRU-1.2-1-140408

150 CAMBRIDGE PARK DRIVE, 3RD FLOOR CAMBRIDGE, MA 02140 617.876.8085 www.mak.com