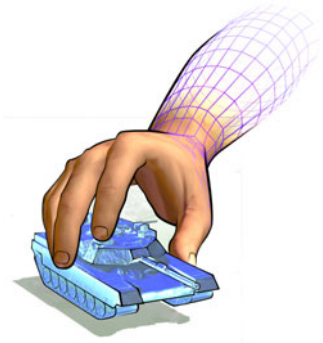

VR-Forces 3.9 Release Notes

This release note contains the following release-specific information for VR-Forces Release 3.9:

Document ID: VRF-3.9-3-060222

Systems Supported

VR-Forces 3.9 is available for the platforms listed in Table 1. For the availability of other platforms, contact your MÄK salesperson. For toolkit users, application code must be built with the indicated compilers in order to link to VR-Forces libraries.

Table 1: Platforms supported

Operating System	Compiler
Red Hat Enterprise Linux Workstation 3	GNU C++ (GCC) 3.2.3
Red Hat Enterprise Linux Workstation 4	GNU C++ (GCC) 3.4.4
Red Hat Fedora Core 3	GNU C++ (GCC) 3.4.4
Windows 2000 SP2, XP	Microsoft Visual C++ 6.0, Service Pack 5 Microsoft Visual C++ 7.1

If you are building VR-Forces with Visual C++ 7.1 on Windows 2000, you must install the latest version of the directX SDK (obtainable from Microsoft).

Patching the Microsoft Visual C++ 6.0 Compiler

VR-Forces is transitioning to use of the Standard Template Library, particularly for lists and map objects. Microsoft Visual C++ version 6.0 has a bug with the STL that is fixed with the supplied *xtree-msvc++patch* file. If you use MS Visual C++ version 6.0, you must install a patch to a standard header file. The patch comes from Dinkumware, Ltd (the company that developed the Standard C++ Library header files for Microsoft), but for convenience, we are distributing it with VR-Forces releases.

The patch fixes a bug that would cause applications to crash if they pass std::sets or std::maps from an application to a DLL or vice-versa, something that IEEE 1516 federates must do. For details about the patch go to http://www.dinkumware.com/vc_fixes.html.

To install the patch:

1. In the top level VR-Forces directory, is a file called *XTREE-msvc++patch*. Rename this file *XTREE*.
2. Make a back-up version of the original *XTREE*.
3. In the Visual Studio *include* directory (for example, *C:\Program Files\Microsoft Visual Studio\VC98\Include*), replace the original version of *XTREE* with the patched version.

Building on Linux

The example Makefiles require a patched version of gmake 3.80, which has been included for your convenience in the *./mkbin* directory. This version of gmake is based on version 3.80, but includes a memory fix that is needed to use our makefiles. The source code for the modified gmake is freely available from *http://ftp.mak.com/out/gmake3.80-patch1.tar.gz*. This patch will be included in future versions of gmake.

Before you compile the examples, go to the top level of your installation and create a symbolic link 'VR-Link' to your VR-Link installation and another called 'RTI' to your RTI installation.

Disk Space Requirements

A full installation of VR-Forces requires approximately 866 MB of disk space. Terrain databases use approximately 137 MB and documentation (primarily the class reference) uses 275 MB.

Compiler Compatibility on Windows

MÄK provides versions of product releases that have been compiled with Microsoft Visual C++ 6.0 and 7.1. When you run MÄK products together on the same computer, for example, VR-Forces and the Stealth, we strongly recommend that you run versions compiled with the same compiler. Mixing products compiled with different versions of the compiler can result in program instability.

MÄK Product Compatibility

To build VR-Forces applications, you must link with VR-Link 3.9.6. If you are building for HLA and want to link with the MÄK RTI, use MÄK RTI 2.4.2.

VR-Forces 3.9 is a parallel release to MÄK Plan View Display 2.9. It is not compatible with previous releases of VR-Forces.

Qt Release Compatibility

If you want to do development using the VR-Forces GUI API, you need to use Qt, a cross-platform GUI toolkit from Trolltech. The VR-Forces GUI was built with Qt release 3.3.5. A VR-Forces developer's license includes a Qt development license. MÄK provides you with a Qt license key. However, you must download the correct version of Qt from *www.trolltech.com*.

Third-Party Library Requirements

DtTerrainProfileWidget is based, in part, on the work of the Qwt project (*http://qwt.sf.net*). To compile examples or user projects, you must download the QWT distribution listed on their home page. Set the MAK_QWTDIR environment variable to point to the location of the QWT library.

Network Compatibility

HLA only

VR-Forces 3.9 was built against VR-Link 3.9.6 and is compliant with:

- RPR-FOM 1.0 and a subset of 2.0 (draft 6 and 17)
- MÄK RTI 2.4.2
- DMSO RTI NG 6.

If you need to run the Linux version of VR-Forces with the DMSO RTI, contact MÄK technical support.

VR-Forces 3.9 is compatible with applications that use earlier versions of VR-Link if they support versions of the RPR FOM listed above.

DIS only

VR-Forces 3.9 supports DIS 2.0.4, IEEE 1278.1, 2.1.4, and IEEE 1278.1a, and can therefore interoperate with DIS applications of any of these versions.

Backwards Compatibility

Backwards compatibility issues are noted throughout the New Features and Updates sections.

RPR FOM Versions Supported

VR-Forces 3.9 has built-in support for versions 1.0 and 2.0 (draft 6 and 17) of the RPR FOM. It also supports VR-Link's ability to support alternative FOMs through the FOM Mapper. By default, VR-Forces 3.9 uses RPR FOM 1.0.

If you are building an application with the VR-Forces toolkit and you want to specify a version of the RPR FOM through code, pass the version number (for example, 2.0006) to the *DtRprFomMapper* constructor and pass the resulting object to the *DtExerciseConn* constructor. Also, make sure you are using a federation execution name that corresponds to the right FED file. For example:

```
DtExerciseConn conn("VR-Link20006", "MyApp", new DtRprFomMapper(2.0006));
```

In order to support RPR FOM 1.0, we have added the following extensions (which are supported in later versions of the RPR FOM):

- EnvironmentProcess
- GriddedData
- EmbeddedSystem.IFF
- EmbeddedSystem.IFF.NatoIFF
- EmbeddedSystem.IFF.NatoIFF.NatoIFFTransponder
- EmbeddedSystem.IFF.NatoIFF.NatoIFFInterrogator
- EmbeddedSystem.IFF.SovietIFF
- EmbeddedSystem.IFF.SovietIFF.SovietIFFTransponder
- EmbeddedSystem.IFF.SovietIFF.SovietIFFInterrogator
- EmbeddedSystem.IFF.RRB
- BaseEntity.AggregateEntity
- ObjectRoot.BaseEntity.PhysicalEntity.Lifeform.

For both RPR FOM 1.0 and 2.0 VR-Forces 3.9 relies on the LgrControl and ViewControl custom MÄK extensions.

To use the embarkation feature in HLA, you must use RPR FOM 2, draft 17. The Windows installation includes entries on the Start menu that automatically configure VR-Forces to start with the correct FOM Mapper. If you want to use the DMSO RTI-NG with embarkation, see the instructions in "Requirements for Using Embarkation in HLA," on page 13.

New Features and Updates

This release of VR-Forces has many updates and new features. See the Table of Contents on page 1 for a complete listing of subject headings. The most significant changes are as follows:

- Support for embarkation and disembarkation.
- Takeoff and landing tasks for rotary-wing and fixed-wing entities.
- Toolbars have been consolidated into a utility toolbar with Navigation and Objects tabs.
- Parameters in vrfSim.mtl, vrfGui.mtl, and pvd.mtl have been renamed.
- Command line options have been added and revised.
- Support for MetaFlight terrain databases.
- The ImageSplitter utility has been replaced with the more functional the MÄK Terrain Database Tool.
- You can now localize text displayed on the map.
- A new sensor model allows you to easily add sensors in various sensing domains. The GUI can display sensor contacts.
- The GUI and simulation APIs have been updated to support embarkation, the new sensor scheme, and the new toolbar configuration.
- Joystick support has been improved.
- Many other changes to the simulation, GUI, and terrain APIs.

All new GUI and configuration features are fully documented in *VR-Forces User's Guide* and *VR-Forces Configuration Guide*.

Changes to the VR-Forces Application

This section describes changes to the VR-Forces GUI and configuration. These changes are documented in *VR-Forces User's Guide* and *VR-Forces Configuration Guide*.

Startup and Configuration Changes

This section describes new configuration options, command line options and changes to startup behavior.

Scenarios With Missing Object Maps Can Now be Remapped

If a scenario is missing its object map (*.omp*) file, you can remap the objects and create a new object map file. By default in combined mode, all the entities are remapped to the combined mode back-end. By default in separate mode, the remapping dialog box lists the missing back-end as "UNKNOWN".

Changes to Configuration Files

- The parameters in *pvd.mtl*, *vrfGui.mtl*, and *vrfSim.mtl* have all been renamed. (The changes were made to conform to the requirements of the new command line processing code in VR-Link.) The PVD and VRF prefixes have been removed. Some parameters have been replaced, for example, ignoreAdvisories has been replaced with **useAdvisories**. Therefore, you will not be able to copy configuration files used in previous versions of VR-Forces to release 3.9.
- **hideOffScreenSymbols** is now set to true by default, for performance reasons.
- **remoteControllerReceivePort**, in *vrfGui.mtl*, sets the port used by the *DtVrfRemoteController's* file transporter to receive order of battle and plan files from a back-end in response to a save scenario request. If the port requested cannot be opened, the next open port is used.
- **cgfDispatcherReceivePort**, in *vrfSim.mtl*, sets the port used by the *DtCgfDispatcher's* file transporter to receive order of battle and plan files from a *DtVrfRemoteController*. If the port requested cannot be opened, the next open port will be used.
- **fedFile**, in *pvd.mtl*, specifies the FED file to use.
- **processSignalInteractions**, in *pvd.mtl*, enables or disables processing of signal interactions for transmitters.
- **commandLineDiscoveryTimeout**, in *vrfGui.mtl*, specifies, in milliseconds, how long the front end waits to discover a back-end when loading a scenario from the command line.
- **useAbsoluteTimeStamps**, in *pvd.mtl*, allows you to specify use of relative or absolute time stamps. (DIS only)

New Parameter in symbolmap-ttf.mtl

The *symbolmap-ttf.mtl* file has a new optional parameter, which specifies a scale factor (from 0 to 1) to scale the font on top of the font size (it will be multiplied against the font size to get a new font size). The parameter must be specified before the optional parameters for the background character and the color and transparency list for the font. If you have added the background character to any entries in *symbolmap-ttf.mtl*, insert a scale factor of 1.0 before the background character value to keep the scale correct.

Localization of Text in the Map Area, Menus, and List Boxes on Windows

VR-Forces now supports localization of language for text items that can be entered via dialog boxes (names, labels, overlay tab names) and as items found in files such as the OPE, *vrf.ent*, and overlay menu files.

To enable this support for Windows, open the Control Panel and select Regional and Language Options. On the Advanced tab, select the language you wish to display these text items in for "Language for Non-Unicode Programs". You also need to be sure you have the keyboard software installed for the language you want to type in. With the correct keyboard selected and the correct language selected, you can now type directly into the text files to translate the English names or directly into the dialog boxes. These changes will also be saved to the *.oob* file for the scenario.

New Startup (Command Line) Options

VR-Forces has the following new startup options:

- `-d vrfsim_MTL_filename` (default: *vrfSim.mtl*) specifies the configuration file loaded by the VR-Forces back-end (*vrfSim.exe*).
- `--daemon` (and corresponding **daemon** *vrfSim.mtl* configuration option) starts *vrfSim* as a daemon process (Linux only). This means that it will not attempt to read characters from a console, and will fork a background process on startup. *vrfSim* will cleanly exit on receipt of the SIGTERM, SIGQUIT, SIGINT signals. By default, daemon mode is off.
- `--defaultConfigFile filename` specifies the configuration file that vrfGui loads (default: *vrfGui.mtl*.)
- The `-F ttl` option is no longer supported. To specify multicast time-to-live, use the `--mcastTtl` option.
- VR-Forces supports the standard VR-Link options described in Table 2 and Table 3.

Table 2: Default HLA startup options

Parameter	Syntax	Default
Execution name	`{-x | --execName} exec_name`	VR-Link
Federate name	`{-N | --federateName} federate_name`	VR-Link
FED file name	`{-F | --fedFileName} fedfile_name`	VR-Link.fed
FOM Mapper library name	`{-f | --fomMapperLib} libname`	
FOM Mapper initialization data	`--fomMapperInitData data`	
Notification level	`{-n | --notifyLevel} level`	2
RPR FOM version	`--rprFomVersion version_number`	1.0

Table 3: Default DIS startup options

Parameter	Syntax	Default
DIS port	`{-P | --disPort} port`	3000
Exercise ID	`{-x | --exerciseId} ID`	1
Application number	`{-a | --appNumber} number`	2
Site ID	`--siteId ID`	1
Destination address	`{-A | --destAddrString} address`	0.0.0.0
Send buffer size	`--sendBufferSize size`	-1 (use system default)
Receive buffer size	`--recvBufferSize size`	-1 (use system default)
Multicast TTL	`--mcastTtl ttl`	-1 (use system default)
Multicast address	`{-S | --mcastAddresses} addresses`	
Notify level	`{-n | --notifyLevel} level`	2

Font Size Setting for Linux

Because the default font sizes from some distributions are too large, the font size is now loaded from *./config/qtrc*. The default is now a sans serif 10 point font. To edit the font setting, use Qt's qtconfig utility to edit your *qtrc* file (by default it is stored in *~/.qt/qtrc*) and then copy the new file to the *./config* directory.

Batch Scenario Files

Batch files have a new keyword, **simulation-run-duration**, that allows the batch file to run for the specified amount of simulation time, rather than wall clock time. This can speed up execution of batch runs if you want to run the scenarios faster than real time.

Entity Task and Set Data Request Changes

This section describes new and changed tasks and set data requests.

Embarking and Disembarking Entities

Embarkation places entities into or onto other entities, for example, dismounted infantry in a truck, trucks on an LCAC, or aircraft on an aircraft carrier. While embarked, the entities travel with the entity on which they are embarked. By default, embarked entities are not visible on the map. However, you can make them visible.

You can attach objects to entities so that embarked entities can move about. For example, an aircraft could taxi along a route on an aircraft carrier, or a infantryman could walk to a waypoint on the flight deck.

To embark or disembark entities, you use the embark and disembark tasks, described in Chapter 8, *Assigning Tasks and Set Data Requests* in *VR-Forces User's Guide*, or you can embark and disembark instantly, as described in "Embarking and Disembarking Entities," on page 7-13 in *VR-Forces User's Guide*. To attach objects to entities, you edit their properties, as described in "Attaching an Object to an Entity," on page 9-13 in *VR-Forces User's Guide*.

The embarkation status of entities is displayed in the Embarkation View of the Objects toolbar.

The Display Options dialog now has an additional check box to turn on/off the display of this label and display of embarked objects.

A new example scenario, embarkdemo is included in the release. "The Embarkdemo Scenario," on page A-16, in *VR-Forces User's Guide* explains how to create the embarkdemo scenario.

Requirements for Using Embarkation in HLA

To use embarkation in HLA federations, you must use RPR FOM 2.0, draft 17 or later. In Windows, the Start menu has options to run VR-Forces using RPR FOM 2. For details about specifying the RPR FOM version on the command line, see "Command Line Options for HLA Federations," on page 4-28, in *VR-Forces User's Guide*.

If you are using the DMSO RTI-NG, only version 6.0 supports embarkation and you must configure it as follows:

```
(FederationSection
...
;; PARAMETER: FederationSection.AllowReentrantUpdatesAndInteractions
 ;; DESCRIPTION:
 ;; MAY BE SPECIFIED AT THE FEDERATE LEVEL
 ;; RANGE: {Yes,No}
 ;; DEFAULT_VALUE: No
 ;;
 (AllowReentrantUpdatesAndInteractions Yes)

...
)
```

Embarkation Capability Table

Table 4 lists the entities that support embarkation and the number of embarked entities they can hold. High fidelity entities (HF column) have polygonal models of the entity and embarkation slots defined in such a way that the embarking entities can smoothly embark on the host entity when viewed in the MÄK Stealth.

Table 4: Embarkation support

Entity	Can Carry	HF
M2A2 Bradley IFV	6 DI	
M113 Armored Utility Vehicle	11 DI	
M-939A2 5-Ton Truck	14 DI	x
MT-LB Multipurpose Armored Vehicle	11 DI	
BTR-80 Armored Personnel Carrier	11 DI	
BMP-2 AVF	7 DI	
C-130 Hercules	92 DI or 3 ground vehicles	
CH-46E Sea Knight	11 DI	
CH-53E Super Stallion	55 DI	
UH-60 Blackhawk	11 DI	
OH-58 Kiowa	6 DI	
Mi-2 Hoplite	11 DI	
LCAC Landing Craft Air Cushion	12 HMMWVs or 4 trucks or 1 M1A2 or 1 generic ground vehicle	x
LSD-49 Harpers Ferry	2 LCACs and 2 helicopters	x
Guided Missile Destroyer	2 helicopters	
Aircraft Carrier	12 fighters or 9 fighters and 2 cargo planes. Three helicopters. 70 planes below deck.	x

Fixed-Wing Takeoff and Landing Tasks

New tasks, Fixed-Wing Land and Fixed-Wing Takeoff have been added to allow fixed-wing entities to take off and land from a specified runway route. These tasks automate the process for takeoff and landing supported in previous versions of VR-Forces. For details, see the task procedures in Chapter 8, *Assigning Tasks and Set Data Requests* in *VR-Forces User's Guide*.

Rotary-Wing Landing Task

The Rotary-Wing Landing task allows rotary-wing entities to land at a specified point. For details, see the task procedures in Chapter 8, *Assigning Tasks and Set Data Requests* in *VR-Forces User's Guide*.

Specifying Collision Avoidance Types

A new set data request, Set Collision Avoid Types, allows you to specify whether or not a particular entity should avoid entities, buildings, and vegetation. For details, see the task procedures in Chapter 8, *Assigning Tasks and Set Data Requests* in *VR-Forces User's Guide*.

Additional Lifeform Postures

The Set Posture dialog box in has the following new postures:

- Swimming
- Parachuting
- Jumping
- Sitting
- Squatting
- Crouching
- Wading.

Backward Compatibility of Ground Entity Move To Behavior

In previous version of VR-Forces, civilian vehicles were configured to follow paths when moving to a location. In VR-Forces 3.9, when you give a ground vehicle a Move To task, you can specify direct movement or path planning. As a result of this change, if you run a scenario created with a previous version of 3.9 and that scenario has civilian vehicles with plans containing Move To tasks, the civilian vehicles default to direct movement, rather than path following.

Entities can Move Along a Route Starting at the End

The Move Along Route task now has a check box that lets you specify that the entity move along the route from the end to the beginning, rather than beginning to end.

Calculation of the Location of the Aggregate Icon

Aggregate location (and therefore location of aggregate icons) is now calculated as the center of the locations of the aggregate subordinates, as follows:

- Live subordinates that are not independently tasked
- If there are no live subordinates that are not independently tasked, then the center of live subordinates that are independently tasked
- If all subordinates are destroyed, the center of all of them.

Change to How Aggregates Move Along Routes

Aggregates now move along a route when the leading edge of the formation is at the first point of the route. It finishes when the leading edge reaches the last vertex in the route. In previous releases, aggregates started with the aggregate centered on the first vertex and ended with the aggregate centered on the last vertex. To override this behavior, subclass *DtAggregateMoveAlongController* (*aggregateMoveAlongController.h*).

Display and Display Options

This section describes changes to display options and navigating the map.

Toolbars and Menu Options have been Reorganized

All toolbars except the simulation control toolbars and the Minimap Color toolbar are now grouped on two tabs of a Utility Panel. The functions of the Echelon View, Entity List, and Object List have been combined in an Objects toolbar that also includes an Embarkation View and Selection Groups View. Other changes to the GUI are as follows:

- The Intervisibility Options menu item is now on the Intervisibility menu.
- Transmitter options, hazard options, and range rings options have been removed from the Symbols tab of the Display Options dialog box and added to the Transmitter Options dialog box, Hazard Options dialog box, and a new Range Rings dialog box, respectively.
- The process for locking and hiding overlays has changed. Each overlay tab now has two icons: an eye and a lock. If the eye is open, the items in the overlay tab layer are visible. If the lock is open, items on the overlay can be edited; when the lock is closed, objects cannot be edited. This change is also reflected in the Overlay Manager.
- The Information toolbar can now show all polygons at a particular terrain location. The toolbar lists the soil types at the selected point. If there are multiple soil types, you can view their location data individually (including altitude). For details, see "Displaying Information about a Point," on page 14-3, in *VR-Forces User's Guide*.
- The Object Count toolbar is a new toolbar that displays the number of entities, aggregates, and control objects in the exercise.

- Checkpoint status is now displayed on the status bar. There is no longer an independent Checkpoint Status toolbar.
- The Terrain Database Information dialog box has been added to show basic information about the terrain database that is currently open. For details, see "Displaying Information About a Terrain Database," on page 14-2, in *VR-Forces User's Guide*

Iterating through the Entity and Object Lists

You can iterate through the entity and environmental object lists in the Objects toolbar using the following keys:

- Next entity – period (.)
- Previous entity – comma (,)
- Next environmental object – greater than sign (>)
- Previous environmental object – less than sign (<).

VR-Forces iterates through the current views of these lists. If you change the filters in effect, the list of objects through which you can iterate changes.

Show or Hide Stealth Icon

You can now show or hide the stealth icon, by choosing Objects → View Objects → Stealth Icon.

Zoom to Extents

You can select an object or objects and zoom the map to the extents of those objects. If no objects are selected, Zoom to Extents acts as Reset to Global View. See "Zooming to Extents" on page 5-11 in *VR-Forces User's Guide*.

Overlay Label

A new label, Overlay, has been added to the list of labels on the Symbols tab of the Display Options dialog box. This has been added to support being able to place any object on an overlay.

New Emitter Beam Option

A check box has been added to the Emitter Beam Options dialog box that allows you to select whether or not the range is to be used as a scale factor to a calculated beam radius or as the actual radius. If the Use As Scale Factor check box is checked (the default) the Radius Scale value is applied to a range calculated value (based on azimuth, elevation, power and exponent). This functionality is used to match the Stealth calculation of emitter beams. If this Use As Scale Factor is cleared, the Radius Scale label changes to Beam Range and the value is treated as an absolute beam distance, in meters, for drawing the beam.

Displaying Entity Information

Double clicking on an entity in any of the views on the Objects toolbar displays entity information.

Processing Signal Interactions

The Transmitter Options dialog box has a new check box that enables or disables the processing of signals and transmitters.

Overlay and Terrain Database Changes

This section describes changes to management of overlay objects and various terrain database changes.

Undo/Redo Support

VR-Forces supports Undo and Redo for changes to object location made in the front-end (movement, vertex movement, vertex addition or deletion). See "Undoing Object Vertex Edits," on page 9-21 in *VR-Forces User's Guide* for details.

Ability to Group All Objects on Overlays

You can now group all objects (entities, aggregates, control objects) on overlays. Since you can hide overlays, grouping objects this way lets you selectively hide and display subsets of the object types that you can display and hide from the View menu. These overlays are local (not published on the network). Once on an overlay, these objects can be moved around using the Overlay Manager. See "Assigning Objects to Overlays," on page 6-17 in *VR-Forces User's Guide* for details.

Geotiff Support

Geotiffs (TIFF files with geocentric data) are now supported.

Save Overlays in Geocentric Coordinates

Overlay files are now saved in geocentric coordinates by default. A check box has been added to the Save As dialog to allow you to turn off this feature and still save in local coordinates.

Multiresolution Paper Map Addition

Paper maps have a new keyword, **Cache_Images**. When it is enabled, images are kept in memory when they are not on screen, rather than being discarded, which is the default behavior. Keeping the images in memory increases drawing performance. However, if images are large, do not use this option as memory could become an issue. The *Berkeley.map* and *Makland.map* terrains have this feature enabled.

New TdbTool GUI

The TdbTool utility now has a graphical user interface. It allows you to import terrain databases and save them to GDB, as you can do in the VR-Forces GUI. It incorporates the functions of the Image Splitter, which is no longer provided, and it allows you to edit the soil types of polygons. Other features of the tdbtool are still available from the command line interface.

Specifying Spatial Subdivisions Dynamically

Spatial subdivisions are a tuning parameter for terrain databases and objects. You typically specify spatial subdivision parameters when you convert a terrain database to GDB format using the tdbtool. However, you can now adjust the spatial subdivision of the terrain or objects during runtime. The default settings for spatial subdivisions are appropriate for most simulations. However, fine tuning them may benefit scenarios with have large numbers of objects, particularly if they are spread out. For details, see "Configuring Spatial Subdivisions," on page 1-15 in *VR-Forces Configuration Guide*.

GUI API Changes

This section describes changes to the GUI API.

Showing and Hiding Symbols

Because of the ever increasing number of filters and ways to show or hide symbols, an optional parameter has been added to the show() and hide() member functions that dictate which module or feature is trying to show or hide a symbol. The setKeepHidden() member function was an effort to try to keep symbols hidden, however, it was not successful, because there were too many filters or features that wanted to keep a symbol hidden. So, for release 3.9, setKeepHidden() has been removed.

The *symbolModules.h* file contains the list of current features and modules that may want to hide symbols:

```
define DtNewSymbolModule(newType, value) const DtSymbolModule newType =
  (DtSymbolModule)((int)DtUserSymbolModuleStart + value);
enum DtSymbolModule {
  DtAnySymbolModule = -1,
  DtViewDriverSymbolModule = 1,
  DtAggregateFilterSymbolModule = 2,
  DtDragDropHandlerSymbolModule = 3,
  DtVertexEditorHandlerSymbolModule = 4,
  DtCreationHandlerSymbolModule = 5,
  DtForceSymbolModule = 6,
  DtCircleResizingHandlerSymbolModule = 7,
  DtOverlayFilterSymbolModule = 8,
  DtEmbarkationSymbolModule = 9,
  DtHealthFilterSymbolModule = 10,
  DtEntityTypeFilterSymbolModule = 11,
  DtSymbolUpdaterSymbolModule = 12,
  DtViewObjectsFilterSymbolModule = 13,
  DtUserSymbolModuleStart = 1000
};
```

The default module *DtAnySymbolModule* acts as show/hide does today. If you call hide() with that module, it tries to hide the specified symbol. However, the symbol might be shown by another filter or feature. If you call show() with that module, the cache of requests to hide the symbol is cleared and the symbol is displayed. However, the symbol can subsequently be hidden by another call to hide().

The fogOfWar example has been updated to show how to use this new feature. The following code snippet is called when a new symbol is crated. It either adds the symbol to the display or hides it, depending on the value of the variable “show”.

```
DtNewSymbolModule(DtFogOfWarSymbolModule, 0);
if (show)
{
  if (toEffect->hiddenByModule(DtFogOfWarSymbolModule))
  {
    toEffect->show(DtFogOfWarSymbolModule);
    // Only do this if we are using the main map area
    if (myMapArea && dlg)
```

```
      {
        fApp->removeFogOfWarItem(modelData);
        changed = true;
      }
    }
    }
  else
  {
    if (!toEffect->hiddenByModule(DtFogOfWarSymbolModule))
    {
      toEffect->hide(DtFogOfWarSymbolModule);
      // Only do this if we are using the main map area
      if (myMapArea && dlg)
      {
        fApp->addFogOfWarItem(modelData);
        changed = true;
      }
    }
  }
```

Check your code closely for any calls to show() or hide() and take advantage of this new parameter. If you want your symbol to stay hidden, define a new module for hiding it by and call hide() with that module. The symbol will not be shown again until all modules that have hidden the symbol have called a matching show() on it, so, if you always want the symbol to stay hidden, call hide() on it once with your module and do not call a matching show.

Also, if you have code that looks like this:

```
if (mySymbol->isShown()) { mySymbol->hide(); }
```

remove the `->isShown` check, because you want the symbol to know that you request it to stay hidden. Until you register your module with the symbol, it could be shown at any time.

If you want to know whether or not your module has been added to the symbol to hide it, you can call the symbols hiddenByModule() member function with your module identifier. If it returns true, it has hidden it (see the fogOfWar example).

Notes

- The show() and hide() functions now use a module "*DtFogOfWarSymbolModule*" to tell that symbol that the fog of war filter wants it to stay hidden, and a matching call to show when the fog of war filter wants to remove it from consideration from hiding the symbol.
- Multiple calls to hide a given module have no effect (the module will only be hidden once.) Similarly, a call to show a module that is visible has no effect.

Overlay View Manager

The setOn() member function and layerOnStateChanged signal are no longer used by the Overlay Manager (and all related overlay items). They have been replaced by setEditable() and layerEditableStateChanged().

New View Menu Action Groups

New action groups have been added to organize the view menu:

- myTerrainControlDisplayGroup organizes all terrain related view items
- mySimulationControlDisplayGroup organizes all simulation related items.

Help Menu

The enum for the Help Menu has been changed from DtHelpMenu to DtAppHelp-Menu.

Changes Made to Implement the Objects Toolbar

The Objects toolbar consolidates several formerly independent entity and object views and adds new ones.

The Echelon Panel is now Part of the Objects Toolbar

The Echelon View has been integrated into the Objects toolbar. It also now supports display of embarkation relationships. If you have written code to make the Echelon Panel dockable, you can remove it. The following classes have changed:

- The *DtPvdEchelonPanel* class has been removed and replaced with *DtPvdEchelon-View*
- The *DtVrfEchelonPanel* class has been removed and replaced with *DtVrfEchelon-View*
- In the *DtVrfGuiEchelonViewHandler* the **myEchelonPanel** member variable has been replaced by **myHierarchyPanel** to prevent confusion with class name.

If you have overridden code to create your own version of the echelon tree, panel, or view, you must update your code. You do not need to change any code as part of your EchelonTree subclass, because the changes are mostly reorganization to account for the new Embarkation View.

Previously, you needed to subclass the echelon panel to create a new echelon tree. Now, you subclass the echelon view. So, if your code looked like:

```
# ifndef MYECHELONPANEL_H_
# define MYECHELONPANEL_H_
# include "vrfEchPanel.h"

class MyEchelonPanel : public DtVrfEchelonPanel
{
  Q_OBJECT
public:
  MyEchelonPanel( QWidget* parent = 0, const char* name = 0,
    WFlags fl = 0 );
  static DtTMEchelonPanel* create( QWidget* parent = 0,
    const char* name = 0, WFlags fl = 0 );
protected:
  virtual DtTMEchelonTree* createEchelonTree(QWidget* parent);
```

```
};

# endif
# include "myEchelonPanel.h"
# include "myEchelonTree.h"

MyEchelonPanel::MyEchelonPanel( QWidget* parent, const char* name,
  WFlags fl) :
  DtVrfEchelonPanel(parent, name, fl)
{ }
DtTMEchelonTree* MyEchelonPanel::createEchelonTree(QWidget *parent)
{
  return new MyEchelonTree(parent);
}
DtTMEchelonPanel* MyEchelonPanel::create( QWidget* parent,
  const char* name, WFlags fl)
{
  return new MyEchelonPanel(parent, name, fl);
}
```

it should now look like:

```
# ifndef MYECHELONVIEW_H_
# define MYECHELONVIEW_H_
# include "vrfEchView.h"

class MyEchelonView : public DtVrfEchelonView
{
  Q_OBJECT
public:
  MyEchelonView(const DtEchelonHierarchy *hier,
    const DtEchelonViewOptions *opt = NULL);
  static DtEchelonView* create(const DtEchelonHierarchy *hier,
    const DtEchelonViewOptions *opt = NULL);
protected:
  virtual DtTMEchelonTree* createEchelonTree(QWidget* parent);
};

# endif
# include "myEchelonView.h"
# include "myEchelonTree.h"

MyEchelonView::MyEchelonView(const DtEchelonHierarchy *hier,
  const DtEchelonViewOptions *opt) :
  DtVrfEchelonView(hier, opt)
{ }
DtTMEchelonTree* MyEchelonView::createEchelonTree(QWidget *parent)
{
  return new MyEchelonTree(parent);
}
DtEchelonView* MyEchelonView::create(const DtEchelonHierarchy *hier,
  const DtEchelonViewOptions *opt)
{
  return new MyEchelonView(hier, opt);
}
```

Also, instead of creating the echelon panel through the factory, you now do this:

```
factory.setEchelonViewCreatorFcn(MyEchelonView::create);
```

The Entity List and Environmental List are Toolbars

The entity and environmental lists are now part of the Object toolbar. If you have written code to make these items dockable, you can remove that code. As part of this change, the entity list and environmental list handlers have been removed and replaced with a factory created *DtObjectsList* class. This item is created as part of window creation and initialization.

Tdb and Papermap Drawers

The *DtTdbDrawer*, *DtQtDrawer*, and *DtPaperMapDrawer* classes have been moved from the *DtTerrainApplication* level to be created as instances of the terrain background. These objects used to exist at the application level (that is, one instance per application.) They are now created at the map level (one instance per map.) Creation of these objects has been moved to the factory. If you have subclassed these objects and created them as a subclass of the application, you can install a new static create() member function, as follows:

```
factory.setTdbDrawerCreatorFcn(MyTdbDrawer::create);
```

No other code in your subclass needs to be changed.

DtTerrainLocationEntryWidget

The *DtTerrainLocationEntryWidget* class's init() member function now takes the coordinate system collection as the first parameter. This can be referenced as DtTMApplication::app()->coordinateSystemCollection().

Object Mapping Classes

The lookup functions in *DtMappingList* and *DtAddressMap* have been changed to return a boolean for success of the lookup and to put the result in an address parameter. This was done to allow a null simulation address to be put into the map to support remapping of missing object maps as detailed in "Scenarios With Missing Object Maps Can Now be Remapped," on page 9.

Setting Font Size on Linux

The font size was hard coded in DtPvdApplication::initApplicationFontSize(), which was removed in favor of loading the font specifications using QSettings in DtTMApplication::initFromQSettings(). For more information see "Font Size Setting for Linux," on page 12.

Overlay Objects can be Considered Control Objects

The *DtPvdApplication* now has functions to specify which entity mapper keys are considered to be control objects. This can be done by calling the addControlObject(), removeControlObject() and initControlObjects() member functions. This information will be use when determining which control objects can be parented on overlay layers locally or are initially published on the "map area".

DtNameLabelSymbol Adds Layer Member

Since all symbols can now be placed on overlays, the *DtNameLabelSymbol* now has a "layer" symbol associated with it.

dropEvent() Renamed

The dropEvent() member function in *DtVrfGuiDragDropHandler* has been renamed to dropSymbols(). All code can stay the same.

Terrain Selection Changed

Terrain selection has changed from organizing selected vector features one-to-one (terrain edge to selected terrain edge and terrain node model data to selected terrain node) to organizing around fid codes of the features. Each fid code selected will get a *DtTerrainModelData* representation and, that model data will contain all the features selected that match that fid code. When drawn, composite symbols are created to contain a single symbol for each terrain feature selected by fid code.

The *DtTerrainEdgeModelData* and *DtTerrainNodeModelData* classes have been removed.

Changes to *DtTerrainLineConfigWidget*

DtTerrainLineConfigWidget used to maintain copies of all line symbols that were being edited in order to correctly highlight segments. Because of changes in how terrain segments are kept in model datas, this has been simplified to a single instance of a line symbol that will configure itself to cover the entire selected edge. If you want to override this behavior, you can override the new setupEdgeSymbol(), setupEdgeModelData() and selectEdge() member functions.

Terrain API

This section describes changes to the Terrain API.

Openflight Reader

The OpenFlight reader now loads:

- Mesh nodes with triangle strip primitives. (Triangle strips are the only primitive available and supported by Multigen-Paradigm at this time.)
- Maps with both localized and unlocalized origins.

An OpenFlight terrain database may be composed of multiple OpenFlight files (or tiles.) In unlocalized databases, all OpenFlight files are built with respect to one origin. The OpenFlight reader loads terrains with "Local Origin Offset" set to zero or set to the same value for all the tile files.

In localized databases, each OpenFlight file contains its own origin information. The "Local Origin Offset" values can be different in the tile files. In the Figure 1, each cell represents a terrain file. The dot represents an origin.

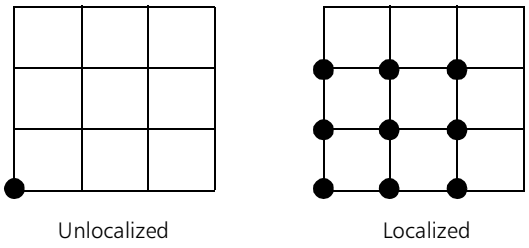

Figure 1. Unlocalized and localized terrains

MetaFlight Reader

VR-Forces can now load MetaFlight files. The MetaFlight reader has the following abilities and restrictions:

- It loads maps with both localized and unlocalized origins.
- Coordinate system information is extracted from the first Openflight file loaded.
- It only loads files that contain a GeometryGridDataset node. FileListDatasets, VirtualTextureDataset, ImageGridDataset, VectorGridDataset, and GeneralGridDataset are not supported.
- It loads the highest level of detail available.
- It does not support paging.

- MetaFlight files use a token replacement scheme to define the path to files. The tokens identify a data item that can identify an element inside the file. The loader processes the DSNAME, DSTITLE, LEVEL, COL, and ROW tokens. It also checks for the MFT_DIR environment variable, which is a variable set by Vega Prime.
- If the <Volume> and <Rootpath> entries in the <GeometryGridDataset> tag are set, the loader uses their values to build a full path to search for the files. If they are empty, the loader looks for the files relative to the location of the *.mft* file.

Changes to Shapefile Support

A new class, *DtShapeProjection*, contains a tree list of *DtShapeProjectionInformation*. This information is read from the shape *.prj* file and is used to convert a shapefile created in one projection to the current database projection. If a projection type is not supported by the *DtShapeProjection* class, you can subclass this class and install the creator to be created in the *DtShapefile* class. See the *DtShapeProjection* header file (*shapeProjectionInformation.h*) for the member function to override if you want to perform your own coordinate projection.

Changes to GDB

Coordinate system nodes now save out spheroid information (including datum shift). The UTM_Coordinate_System also saves out the Transverse Mercator scale factor.

FltMetafileRecord and DfdDfadMetafileRecord Additions

The following items have been added to the FltMetafileRecord and DfdDfadMetafile-Record record types to allow for definition of a reference ellipsoid:

- Reference_Ellipsoid_Semi_Major_Axis
- Reference_Ellipsoid_Semi_Minor_Axis
- Inverse_Flattening
- DatumShift x y z
- Projection_Central_Scale.

If both Inverse_Flattening and Reference_Ellipsoid_Semi_Minor_Axis are defined, the Inverse_Flattening value takes precedence and the minor axis is calculated as:

```
semiMinor = msemiMajor - (semiMajor / Inverse_Flattening)
```

DtExtentInitializer Respects Z Values

The *DtExtentInitializer* class now respects the altitude (Z) values of the specified extent in the *DtExtentMetaFileRecord*. The altitude of the vertices is determined by the altitude of the left, back, bottom vertex and the right, front, upper vertex. These are used as the SW and NE vertices respectively. The altitude of the SE vertex matches the altitude of the SW vertex and the altitude of the NW vertex equals the altitude of the NE vertex. Figure 2 illustrates the polygons that get created from the extents.

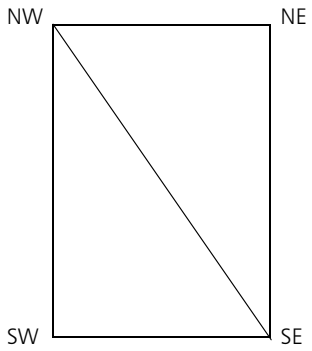

Figure 2. Extent polygons

You can now specify a maximum latitude or longitude that is smaller than the minimum latitude or longitude. VR-Forces wraps around until it is larger. The extent initializes the maximum to be within +- 180 or 90 respectively (for example, min = 150, max = -150).

For example, this allows simulation of the area covered by +150 degrees to -150 degrees.

GDB File Format Changes

The GDB now writes out, and reads, in spheroid and shift information. The UTM_Coordinate_System now writes out scale factor information.

DtGdbNode Class Hierarchy Interface Changes

The allInstancesOf() member function has been changed to take a std::set as the container instead of a std::list. This ensures that no element being accumulated can be counted more than once.

DtTerrainDatabase Interface Changes

In the API const member functions that returned non-const references or pointers to internal members have been replaced with two functions, a const version and a non-const version, which return a const and non-const pointer or reference respectively.

This is an important change for the following reasons:

- Const member functions should never return a non-const reference or pointer to an internal member, as that reference or pointer can then be used to modify the const object, which is an error and incorrect.
- To maintain consistency in the interfaces to our objects, it was important to standardize the accessors.
- In preparation for other concerns, such as multithreading, it was important to better define when objects can be modified and when they cannot be modified.

The most likely result of these changes is that the coordinateSystem() function will be the source of new compile errors. In many places in the VR-Forces code base, a non-const coordinate system pointer was retrieved from a const terrain database. To fix compile errors, define the temporary object, member variable, or function parameter as a const coordinate system pointer instead of a non-const pointer.

Coordinate System Class Hierarchy Changes

The coordinate system classes have been renamed, with the exception of the base *Coordinate_System* class. This has been done to clean up inconsistencies in the names compared to the rest of our code base and to help with their clarity and usability.

Library Changes

The *libproj* library has been added to do projection conversions for geotiff paper maps.

Simulation API Changes

This section describes changes to the Simulation API and Remote Control API.

The Physical World

VR-Forces 3.9 introduces the concept of the physical world. It encompasses anything with a physical presence in the virtual world. Practically speaking, this means any object that has a bounding volume – entities, terrain polygons, vector obstructions, and environmental objects. The physical world model is required for embarkation. It also results in improved efficiency for line-of-sight calculations and obstruction avoidance. All models have been updated to work with the physical world. VR-Forces is backwards compatible with the older models, but you must update to the physical world to take advantage of its improvements.

The physical world is implemented through the DtPhysicalWorld class. You can extend the API to change what is considered to be part of the physical world.

Object Parameters Database and Related Configuration Changes

This section lists changes to the object parameters database and how entities are configured.

- The hit probabilities for the M16 and AK47 rifles have been updated to be much lower to better reflect the probability of hitting the target when firing either of those weapons. This is a behavior change, and may result in slightly different scenario behavior.
- The pseudo-aggregate-move-actuator descriptor has been removed from all aggregate *.ope* files because the component corresponding to this parameter entry has been removed. Its functionality has been replaced by the pseudo-aggregate-extent-actuator.
- Entity OPE files configured with a target-detect-sensor have a new parameter named target-all-priority-lists. The meaning of this parameter is described in "New Parameter target-all-priority-lists Added to Target Detection Sensor," on page 32.
- The F-117 configuration files are now separated from the generic attack plane, and include reduced radar signature compared to other planes.

Firepower and Mobility Kills

All basic VR-Forces models now check the firepower and mobility kill appearance bits, and will not fire weapons or move if the respective bits are true.

Mobility kills are treated by VR-Forces models as failures of the drive system of the entity, so they will coast to a stop, or whatever is appropriate for the entity type, instead of stopping immediately.

New Sensor Model System

A new method for modeling sensors within VR-Forces has been added, called the Signature Sensor System. Now, entities are configured with signature values for various sensor domains, such as visual, radar and infrared. Propagation models are configured which model how these signatures decrease over distance, and sensors configured on other entities use these signature to determine what entities they can see.

VR-Forces is configured with Visual, Radar, Infrared, and Sonar sensor domains by default. You can add your own sensor domains though the configuration files. For details, see "Sensors," on page 5-35 in *VR-Forces Developer's Guide* and "Configuring Sensors," on page 8-2 in *VR-Forces Configuration Guide*.

Improved Multisensor/Multiweapon Capability

The *DtTargetDetectionSensor* has been replaced with *DtSignatureObjectSensor* (and related sensors) and a new component called *DtTargetSelectionController*. It is possible to hook as many sensors as desired to the target selection controller, and to have the target selection controller choose target priorities independently for as many different weapon systems as desired. It can choose targets for each weapon based on target type, relative position of the target to the entity, or a combination of both. Using this, it is much easier to configure multiple weapon systems on a single entity.

Spot Report Generator

A new component, *DtSpotReportGeneratorController*, is included with VR-Forces. It generates and sends spot report radio messages based on the outputs of sensors configured on an entity.

New and Changed Entity Components

New Parameter target-all-priority-lists Added to Target Detection Sensor

The weapon and launch controllers no longer target entities for which they do not have ammunition. This change allows you to target entities in lower priority lists that an entity might have ammunition for.

The Target Detection Sensor has a new parameter, target-all-priority-lists. When target-all-priority-lists is false (the default) if a target is found in a priority list, then lower priority lists are not checked, regardless of whether or not the entity has the resources to fire at the identified target. Therefore, if an entity runs out of ammunition for higher priority targets, but still has ammunition suitable for lower priority targets, it might never use that ammunition because high priority targets are still being found and the lower priority lists are not being checked.

If this value is true (default is false), then all entity lists are checked for possible targets. Also, an entity will fire at low priority targets that are in its sector of responsibility before it fires at high priority targets that are not in its sector of responsibility. However this setting degrades performance. Use use this option only if you know that you have an entity that has multiple weapons for a single controller that might want to target lower priority entities when it runs out of ammunition for higher priority entities.

New Parameters for the Turret Scan Controller

The Turret Scan Controller has two new parameters, minimum-scan-width and scan-sor. minimum-scan-width specifies the minimum width (in radians) of the sector of responsibility (SOR) within which the turret will scan back and fourth. If the SOR width is smaller than this value (default to 10 degrees), with turret will be commanded to a stationary position in the center of the SOR. When the SOR is larger, the turret will scan within the sector, as before.

scan-sor specifies whether or not the Turret Scan Controller should actively slew the turret over the current sector of responsibility (old behavior) or should recenter the turret and hold when the turret is not otherwise in use. Default: true.

The Rotary-Wing and Subsurface max-altitude Parameter has Moved

In rotary-wing and subsurface entities, the max-altitude parameter has moved from the flight-kinematics descriptor to the state repository parameters. This allows the actuator and state repository to clamp the object to the max-altitude. This change affects the following classes:

- *DtRotaryWingEntityParameters* – adds maxAltitude() and setMaxAltitude()
- *DtRotaryWingActuatorDescriptor* – removes maxAltitude(), setMaxAltitude(), and the myMaxAltitude() variable, which is now referenced from the state repository.

flaps-deceleration ***Parameter Added to Fixed-Wing Model***

The fixed-wing model has a new parameter, **flaps-deceleration**, which specifies the additional deceleration caused by deploying the flaps on the fixed-wing entity. The fixed-wing pilot controller will now deploy the flaps when it is attempting to slow down significantly.

Entities Resume Tasked by Superior Status Upon Task Completion

By default, entities now resume being tasked by superior (equivalent to clearing their independently tasked status) when they complete an independent task. In previous releases, entities would retain their independently tasked status indefinitely after completing a task. The only way to set an entity to be tasked by its superior again was to send a *DtSetTaskedBySuperior* set data request message to the entity (or select Tasked by Superior from the VR-Forces GUI). Now it will resume tasked by superior status immediately upon task completion.

You can override this new default behavior and revert to the old behavior (entities retain their independently tasked status upon task completion) by setting the new **tasked-by-superior-upon-task-complete** parameter to false.

Parameters for Fixed-Wing Landing

A fixed-wing landing approach route is created from the following descriptor information:

- **approach-distance**: The distance from the runway at which the approach route should end.
- **approach-altitude**: The height above the terrain at which the approach route should end.
- **landing-speed-ratio**: The percentage of lift the plane's speed should be in order to land. Default: 5% greater than lift.
- **publish-approach-route**: Specifies whether the approach route should be published on the network. Default: true.
- **landing-descent-rate**: The rate at which a plane can descend, in meters per second. This parameter is used to calculate the length of approach from the entity's altitude.
- **landing-gear-down-distance**: The percentage away from the end point at which to turn on landing lights (which puts down the landing gear).

Parameters for DtFixedWingTakeoffController

The point to move to when a fixed-wing entity takes off is determined by the following new parameters:

- **takeoff-target-distance**: The distance away from the takeoff route to fly to after takeoff
- **target-altitude**: The height above the terrain to fly to.

Embarkation Controller

New controllers to handle embarkation have been added to the back-end (along with associated tasks and sets to handle embarkation and disembarkation). There is now a base level embarkation controller (*DtEmbarkationController*) (*embarkationController.h*) that handles both the embarkation of an object onto a parent and the disembarkation from that parent. The *DtEmbarkationController* is a state machine that handles the embarkation of an object from its current location to an embarked location. See the class documentation for *DtEmbarkationController* for details about how it works.

Controllers Derived from DtEmbarkationController

The following controllers have been subclassed from *DtEmbarkationController* to add the following specific overriding behavior:

DtRotaryWingEmbarkationController

DtRotaryWingEmbarkationController overrides the DtAtEmbarkationPoint() member function to immediately set the state to *DtStoppedAtEmbarkationPoint*, because rotary-wing entities do not recognize the *DtStopMoving* task.

DtAggregateEmbarkationController

DtAggregateEmbarkationController passes tasks to currently taskable subordinates. Sets are passed down to all subordinates. The controller also adds the checkEmbarkation-State() member function, which is called every tick regardless of whether or not the aggregate is tasked. The pseudo-aggregate is considered embarked if all available taskable subordinates are embarked. The aggregate is embarked on the same object as the current leader of the aggregate. If any subordinate disembarks, the pseudo-aggregate also disembarks.

DtOccupancyDirectorController

DtOccupancyDirectorController allocates reserved embarkation slots and determines which (if any) objects can be embarked on an object. This controller responds to messages of type DtIfRequestEmbarkationInformation. When the request is received, the controller looks to see if there are any slots available, and, if so, reserves it (if requested) and then returns a DtIfEmbarkationInformation structure with the reservation information. This controller also provides disembarkation information and responds to disembarkation requests. If an entity cannot embark either because the object does not support the requesting object type or because the slots are full, an information return structure is sent with the DtEmbarkCannotEmbark result status.

DtAggregateOccupancyDirectorController

DtAggregateOccupancyDirectorController passes the request for embarkation to the leader of the aggregate. The leader of the aggregate is then responsible for determining who can embark upon it and where.

DtFixedWingLandingController

DtFixedWingLandingController creates the approach route and controls the landing of the plane. See "Parameters for Fixed-Wing Landing," on page 33 for a list of parameters that specify the approach route.

DtFixedWingTakeoffController

Like the landing controller, this controller tasks a plane to move along the takeoff route using the *DtFlyAlongRoute* task. During the taxiing, the plane's speed is set to be the takeoff speed (default: 30% above lift). When it reaches the end of the route, the plane is tasked to fly to a point above the ground, using *DtFlyToLocationTask*. (*DtFlyToLocationTask*, like *DtFlyAlongRouteTask* also has the **useFixedPitch** flag, which is also true).

No information needs to be published since the plane is simply flying to a point. Once the task is complete, if the plane has no other task it circles around the takeoff target point.

Changes in Models and State Repositories Due to Embarkation

Because entities can now be embarked on other entities, and can be tasked to perform operations on those entities, if you model dynamics of an entity, you must change your models to take advantage of this behavior.

Any existing code that assumes that entities are only able to be attached to terrain will continue to work, but this code will not work with the embarkation feature. We recommend that you update your code to remove this assumption.

You cannot assume that an entity is solely attached to the terrain. An entity can now be attached to any other model (except for environmental objects) in the system. When an entity is embarked, you need to know the following information about an entity:

- Terrain intersections
- Velocity relative to the host
- Acceleration relative to the host.

The following code changes need to be made in order to account for those three items:

- Terrain Intersections:
  In your models, if you reference mySimTerrain->terrainDatabase(), change the reference to myEntity->terrainDatabase(). The *DtVrfObject* now has a pointer to a *DtObjectGeometricModel* object. This object wraps both the base terrain (mySim-Terrain->terrainDatabase) and any attached entity. If the object in question is attached to another entity, then an intersection will first be done on the attached entity. If an intersection is found, that will be returned, else the sim terrain will be checked. If the entity is not attached to another entity, then only the sim terrain will be checked. Note that even though this model is subclassed from *DtTerrain*, it is done that way only for the API. Any terrain calls made to that object directly to try and deform the terrain will ultimately fail.
- Velocity relative to the host:
  New member function calls, relativeVelocity() and setRelativeVelocity() replace any references you might have to localvelocity() and setLocalVelocity(). The return and set values are still in local (database) coordinates, but if the entity is attached, the return value of relativeVelocity() will be the velocity of the entity only, whereas if you call localVelocity(), it will be the sum of the velocity of the entity and any objects it might be attached to. You must also call setRelativeVelocity() to preserve the notion that this velocity is for the entity only. If the entity is not attached, these calls will pass through to localKinematicState().velocity() and localKinematic-State().setVelocity();
- Acceleration relative to the host:
  Same as velocity relative to the host, except calls are named relativeAcceleration() and setRelativeAcceleration().

Attaching Environmental Objects to Entities

You can now attach environmental objects (routes, waypoints, areas, and so on) to entities. For example, a route can be placed on an aircraft carrier for a DI to traverse. To support this, an optional parameter (initially the empty string) has been added to createOverlayObject() (before the userData pointer) that can contain the name of the object to attach to. The sendVrfOverlayObjectModifyMsg() member function also has an **attachTo** and **keepExistingAttachment** parameters. This is so the create can assign the attachment and future modifies can keep any attachment that has been created (or, can be changed).

Initializing Extended State Repositories

The functionality of extending state repositories at the base class level has been expanded to allow initialization of the state repository extensions from an object parameters database file.

Previously, you could create derived *DtVrfStateRepositoryUserExtension* classes, which extended the base *DtVrfObjectStateRepository*. These derived classes were a logical extension to an entity's state repository, and therefore were saved and restored as part of the entity in the order of battle file. This capability has been extended to enable you to initialize your derived *DtVrfStateRepositoryUserExtension* from an OPE file.

To take advantage of this capability, you can configure initial values for the readable/writeable data members of your derived *DtVrfStateRepositoryUserExtension* by adding a user-extension parameter entry to an entity's parameter database file. For example, if you have created a derived *DtVrfStateRepositoryUserExtension* of type "my-state-repository-user-extension", that has a single readable/writeable string data member with the *DtReaderWriter* name "my-tag", you could configure its initial value in the OPE file as follows:

```
(state-repository-extension-type "my-state-repository-user-extension")
(user-extension
 (my-tag "some-initial-value")
)
```

To implement this new functionality, some changes to the existing *DtVrfStateRepositoryUserExtension* class API were needed. If you have implemented a derived *DtVrfStateRepositoryUserExtension* classes, you must make the following modifications to your derived class:

- Constructors and the create() member function now take a pointer to a DtVrfObjectStateRepository* as an argument, instead of a reference.
- You must now also implement a clone() member function in your derived class.

Assume Tasks Have Been Removed

All of the assume tasks in VR-Forces have been removed, and all code that requests, sends, and processes these tasks has been removed. The assume tasks were used in the aggregate models in versions of VR-Forces prior to version 3.8. In version 3.8, the aggregate models were completely redesigned. The assume tasks and the mechanism for reporting and handling these them are no longer needed or used.

The following task classes have been removed:

- *DtAssumeMoveTo* (*asmMoveTo.h*)
- *DtAssumeMoveAlong* (*asmMoveAlong.h*)
- *DtAssumeFollowEntity* (*asmFollowEnt.h*)
- *DtAssumePatrolRoute* (*asmPtrlRoute.h*)
- *DtAssumePatrolTwoPoints* (*asmPatrolPts.h*)
- *DtAssumeWait* (*asmWaitTask.h*)
- *DtAssumeWaitDuration* (*asmWtDurTask.h*)
- *DtAssumeWaitElapsed* (*asmWtElpTask.h*)
- *DtAssumeMoveToLocationTask* (*asmMvToLocTask.h*).

If you have overridden the member function DtControllerComponent::reportTaskFor-Assumption(), you should remove it from your code.

Changes Supporting Plan Language Improvements

In order to support self-referencing plans and pattern matching conditional statements, the following changes were made

- The myModifier, myAnyFlag and myResult members and their associated accessors and mutators were moved down to *DtSimSubordinateSearch* from all its subclasses. Any user-created *DtSimSubordinateSearch* subclasses should also remove these members, using the conditionSim example as a guide. The following subclasses were affected:
  - DtEntDestroyedSearch
  - DtEntInAreaSearch
  - DtEntLeftOfLineSearch
  - DtEntHasTargetSearch
  - DtEntUnderFireSearch
  - DtEntEmbarkationStateChangedSearch

- *DtEvaluator* subclasses call the new DtSimSubordinateSearch::search() member function from their evaluate() member functions. This new function supports more complex modifiers in order to support self-referential and pattern searches. Any user-created *DtEvaluator* subclasses should update their evaluate() member functions, using MyEvaluator::evaluate() from the conditionSim example as a guide. The following subclasses were affected:
  - DtEvalEntDestroyed
  - DtEvalEntEmbarkationStateChanged
  - DtEvalEntHasTarget
  - DtEvalEntInArea
  - DtEvalEntLeftOfLine
  - DtEvalEntUnderFire.

New and Changed Component Classes

This section describes controllers that have been added or changed. Also, see "Embarkation Controller," on page 34 and "New and Changed Entity Components," on page 32.

DtAggregateFormationController Changes

The generateListOfSubordinatesEligibleForFormation() member function now returns a DtSubordinateQuery result, which you must free when you are finished with it. isEligibleForFormation() has been removed in favor of new subordinate query API. getCurrentFormation() now takes a DtSubordinateQuery as a parameter, not a vector of strings.

Component Port Architecture Changes

Some *DtInputPort* functionality has been split into *DtSingleConnectionInputPort* to provide flexibility for the port architecture.

If you have created any new input port types by deriving from *DtInputPort*, change your classes to derive from *DtSingleConnectionInputPort* instead. No other changes are necessary.

Joystick Improvements

Many vehicles that did not support joystick control in previous releases are now configured for joystick support. The joystick controllers have been reconfigured with new descriptor classes that allow the axes and buttons used for various controllers to be remapped in the object parameters database. The defaults are still based on the Microsoft Sidewinder Precision 2 joystick, but by changing the mapping parameters you can now configure any type of controller. Table 5 lists the new classes and parameters:

Table 5: Joystick classes, descriptors, and parameters

Class	Descriptor	Descriptor Name	Parameters
DtGroundJoyMove-Controller	*DtGroundJoy-DeviceController-Descriptor*	ground-joy-controller-descriptor	◆ joy-steering-axis ◆ joy-invert-steering-axis ◆ joy-throttle-axis ◆ joy-invert-throttle-axis ◆ joy-brake-button ◆ joy-forward-gear-button ◆ joy-reverse-gear-button
DtGroundJoyTurret-Controller	*DtTurretJoyDevice-ControllerDescriptor*	turret-joy-controller-descriptor	◆ joy-slew-axis ◆ joy-invert-slew-axis
DtBallisticGunJoy-Controller	*DtBallisticGunJoy-ControllerDescriptor*	ballistic-gun-joy-ctrl-descriptor	◆ joy-fire-button ◆ joy-change-ammo-button ◆ joy-elevation-axis ◆ joy-invert-elevation-axis
DtFixedWingJoyFlight-Controller	*DtFixedWingJoy-DeviceController-Descriptor*	fixed-wing-joy-controller-descriptor	◆ joy-pitch-axis ◆ joy-invert-pitch-axis ◆ joy-roll-axis ◆ joy-invert-roll-axis ◆ joy-throttle-axis ◆ joy-invert-throttle-axis ◆ joy-pedal-axis ◆ joy-invert-pedal-axis
DtRotaryWingJoy-FlightController	*DtRotaryWingJoy-DeviceController-Descriptor*	rotary-wing-joy-controller-descriptor	◆ joy-pitch-axis ◆ joy-invert-pitch-axis ◆ joy-roll-axis ◆ joy-invert-roll-axis ◆ joy-yaw-axis ◆ joy-invert-yaw-axis ◆ joy-collective-axis ◆ joy-invert-collective-axis

The *DtBaseJoyDeviceControllerDescriptor* class has been removed. The parameters it attempted to change were global joystick parameters – thus the last controller to set them would set the values for all the controllers. These parameters can still be adjusted in *joyParam.dat*. Any object parameter entries that used base-joy-controller-descriptor must now be updated to use the new descriptor type, which corresponds to the controller type in Table 5.

In prior releases, multiple joystick controllers would call joyDeltas() on the *DtJoyDevice* to poll the joystick for new events. This was a flawed approach because it allowed one controller to consume events before the next had a chance to examine them. For this reason, joyDeltas() is now called only once per frame from DtJoyDeviceManager::tick(). The *DtJoyDeviceManager* is ticked from DtCgf::tick().

New Programming Hooks for Handling Unachievable Tasks

The API has new functions that enable developers to write code to handle situations in which a given task is unachievable. For example, say you have a ground vehicle and you give it a task to move to a waypoint, but there is already an entity parked at that waypoint. In this situation, the entity would never be able to successfully reach the waypoint. It would continuously try to reach that point, until it ran out of fuel. Rather than have this entity be forever stuck on this task, you would like to be able to add some reasoning to the entity, so that as it approaches the waypoint, it can evaluate the situation and take some more sensible alternative action, such as parking nearby.

Handling unachievable tasks is done on a per *DtTaskControllerComponent* basis. We have added two new member functions to the *DtTaskControllerComponent* class. These functions provide a place for you to add code for handling unachievable tasks. The functions are decideToGiveUpTask() and giveUpTask(). The decideToGiveUpTask() is a function that returns a boolean flag indicating whether or not the entity should give up its current task. It is called immediately after every tick when the simulation is running (in play mode) and entity is tasked. If it returns true, then it calls giveUpTask(). The giveUpTask() function provides a place where any special handling, such as calling taskComplete(), can be added. By default, the function decideToGiveUpTask() returns false, and the giveUpTask() function has an empty implementation.

Change to DtAggregateMoveIntoFormationController

The cancelMoveIntoFormation() member function in *DtAggregateMoveIntoFormationController* has been removed. It was not being called by any code in the VR-Forces back-end.

Handling Formation-changed Events in Aggregate Move-along and Patrol-Along Controllers

The DtAggregateMoveAlongController now always processes formation-changed events (generated by the DtFormationState), adjusting route-following assignments for its subordinates. Previously, it only performed this processing if it was currently tasked as a top-level task (not a subtask). This change now frees other controllers from having to handle formation-changed events when its using the DtAggregateMoveAlongController to carry out a subtask. The DtAggregatePatrolAlongController no longer registers for or handles formation changed events (it now relies on the DtAggregateMoveAlongController to handle these events). As a result, the following member function have been removed from the DtAggregatePatrolAlongController:

```
static void formationChangedCallback(
  const DtFormationState& formationState, void* usr);

static void objectManagerRemoveAndDeleteAllCallback(
  const DtVrfObjectManager& objectManager, void* usr);

virtual void processFormationChanged(
  const DtFormationState& formationState);

virtual void onObjectManagerRemoveAndDeleteAll(
  const DtVrfObjectManager& objectManager);
```

DtVrfObjectPredicate Changes

A new kind of predicate class DtNullVrfObjectPredicate (nullVrfObjPred.h) has been added. It is a special-purpose predicate that always returns true and is never intended to actually be evaluated. Rather it is used as a sentinel value when manipulating managed lists of objects.

Changed the names of the DtFilteredObjectList and DtFilteredObjectListManager classes to DtManagedObjectList and DtManagedObjectListManager, respectively. This was done to better reflect the purpose of these classes. Now that we have a DtNullVrfObjectPredicate, it is possible to request a list that is managed (object additions and removals are monitored), but not filtered.

Determining when an Object's Vertices Change

A new callback has been added for determining when a *DtVrfObject's* vertices change relative to one another. This callback is useful when you want to be notified if any of an object's vertices has moved with respect to one another, but the overall object has not moved. You can register for this callback by calling DtVrfObject::addRelativeVerticesChangedCallback() (in *vrfObject.h*) in an instance of the *DtVrfObject* you are interested in.

This callback is different from the existing vertex changed callback in *DtVrfObject* (addVerticesChangedCallback()). addVerticesChangedCallback() invokes callbacks whenever relative vertices change or if the overall object moves.

DtFormationTable can be Overridden

You can now override the contents of the formation-table member of the *DtFormationControllerDescriptor* class. This enables you to extend the content of formation tables. The myFormationMember is now dynamically allocated. To get the descriptor to read in a derived kind of *DtFormationTable*, you must create a derived *DtFormationControllerDescriptor* and override the createFormationTable() member function. In your implementation, have it create and return a new derived kind of *DtFormationTable*. For example,

```
DtFormationTable* DerivedFormationControllerDescriptor::
  createFormationTable(const DtSymbolString& name,
  DtReaderWriterRegistry* parentRegistry) const
{
  return new DerivedFormationTable(name, parentRegistry);
}
```

Support for Simulation Management PDUs in DIS

You can now use DIS simulation management (siman) start/stop PDUs to pause or start a VR-Forces scenario from an external application.

When a start or stop siman PDU is received, the receiving back-end checks the Receiving Entity ID to verify whether it should affect it or not. Table 6 lists acceptable values for Receiving Entity IDs

Table 6: Acceptable receiving entity IDs

site	host	ID	Result
ALL	ALL	ALL	Everyone (example: 65535:65535:65535)
ALL	ALL	0	All back-ends (example: 65535:65535:0)
n	m	0	Exact ID (example: 1:3001:0)

By default, VR-Forces only reacts to PDUs sent to {ALL,ALL,*} and {ALL,ALL,0}. This is because if VR-Forces has multiple back ends, processing a PDU sent to a specific back-end could leave VR-Forces in an inconsistent state. Processing exact IDs is appropriate for situations in which VR-Forces has only one back-end.

The **simMgmtPduProcessLevel** parameter in *vrfSim.mtl* tells VR-Forces which type of receiving entity IDs to process. Table 7 shows how VR-Forces processes start and stop PDUs based on the siman PDU process level.

Table 7: Siman PDU processing level

simMgmtPduProcessLevel	Everyone	All Backends	Exact ID
1			X
2	X	X	
3	X	X	X

For example, if the you want VR-Forces to only process siman PDUs sent to a specific back-end {a,b,0}, set **simMgmtPduProcessLevel** to 1.

When a back-end receives a Stop PDU, it stops immediately regardless of the value of the Real-World Time field in the PDU. The Reason and Frozen Behavior fields are ignored.

New, Changed, and Deprecated Examples

- The decideToGiveUpTask example demonstrates how to add a 'give up' capability to a ground vehicle entity's move to waypoint behavior. In the example, the criteria for giving up is a time limit of 10 seconds. If the entity does not reach the waypoint it is tasked to move toward within 10 seconds, it gives up.
- The Radar Warn Receiver example now uses a Patriot missile battery in its default configuration, which includes a radar publishing emitter system, in place of the customized SAM system. The radar warning receiver and response component are now configured on the red fixed-wing attack/strike entity instead of the blue one. The scanning radar component is no longer part of this example.
- The Image Splitter functionality has been absorbed into the new TdbTool. The Image Splitter example will continue to be provided to demonstrate manipulating terrain through the GUI, but will not be updated with new features to match the TdbTool.
- A sample MetaFlight terrain file has been included in this release – *./data/terrain/Makland-UTM-MFT/Makland-UTM-1-0.mft*.
- The convertParamDb example has been removed, because the format and content of parameter databases has expanded and changed since pre-3.7 versions to too great of an extent for this tool to be useful any longer. If you have parameter databases older than 3.7 that you want to upgrade to the 3.9 version, we recommend that you use the OPD editor to re-create your changes in the 3.9 version of the database.
- A new example called spotReport has been added to demonstrate how to use the new spot reporting capability in VR-Forces. In the example, an aggregate is configured with a new kind of component that listens for and displays spot reports from its subordinates.
- The new subtask example demonstrates how to use subtasks.
- Most examples now have a readme file in PDF format that provides an overview of the example.

Miscellaneous API Changes

The *DtPseudoAggregateMoveActuator* Class has been Removed

The *DtPseudoAggregateMoveActuator* class has been removed and its functionality assumed by the *DtPseudoAggregateActuator* class.

Subtasks

The documentation for subtasks has been completely revised. See "Subtasks," on page 8-3 in *VR-Forces Developer's Guide*.

Modify Message Changes

Since all objects can now be part of an overlay layer, the ability to set the overlay parent has been added to the DtIfModifyVrfObject message. For overlay objects, setting this via the overlay modify message or the object modify message has the same effect.

Vrf Object Data Message Changes

The *DtIfVrfObjectData* message (*ifVrfObjData.h*) has been changed. It now includes a version number and an overlay parent name

Handling of *DtIfControl* Messages Consolidated in *DtCgfDispatcher*

The handling of *DtIfControl* messages (run, pause, rewind, run for duration) was previously spread out between the *DtExerciseClock* and *DtCgfDispatcher* classes. All processing of these messages is now handled exclusively in the *DtCgfDispatcher* class.

DtFormationState

The queryAll() member function now takes a DtFormationQueryResults object as an argument.

DtVrfObjectManager

DtVrfObjectManager::writeOrderOfBattleFile() member function used to have the criteria for which entities were written out embedded inside it. These criteria have been broken out into a separate member function, objectShouldBeSavedToOrderOfBattle().

DtScenario

The save() member function now takes a boolean value which determines if an object map filename gets saved in the scenario file. If false, the name is set to null (""). This was added to allow the *DtCgf* to perform a local save without having to create an object map.

Fixed-Wing Model Changes

The processDeleteVrfObject() and proccessWaypointChangedNow() member functions now take a *DtVrfObject* as a parameter rather than a *DtSimMessage*.

DtBoolean, DtTrue, and DtFalse Deprecated

To be consistent with changes made in recent releases of VR-Link, DtBoolean, DtTrue, and DtFalse have been replaced with bool, true, and false. No code changes are required. The old terminology is type-def'd to the new. You are encouraged to replace instances of DtBoolean in your code with bool.

Changes to Aggregate Location and Heading

Only healthy (not destroyed or mobility killed) and non-independently tasked subordinates are taken in consideration when computing aggregate location and heading. Version 3.8 included destroyed and independently tasked subordinates when computing average location and heading. See *pdfAggExtent.h* for details.

Individual Lifeforms Posture

Individual lifeforms now resume their assigned posture as appropriate after dynamically adjusting their posture to handle steep slopes. In the DtLifeformActuator, posture is adjusted to DtLifeformWalking or DtLifeformCrawling each tick, depending on the the current pitch of the terrain and the maximum slope the entity can tolerate. Previously, if the posture had been changed to handle a steep slope, it never resumed the original posture, even if the terrain leveled out again.

Changes to *DtSubordinateManager*

The objectAboutToBeRemovedCallback() and processObjectAboutToBeRemoved() member functions have been removed from *DtSubordinateManager*. They duplicated work done by *DtOrganizationView*. This change may improve performance.

The DtCgf::closeScenario() Member Function Now Pauses the Status Publisher

In previous releases, if the back-end was in run mode, when a scenario was closed, it would pause the simulation clock but leave the scenario in run mode. This could cause inconsistent behavior. Now, when the scenario is closed through the API, the back-end status publisher is put into pause mode.

Documentation Updates

The print and PDF versions of all VR-Forces manuals have been updated for this release. This section contains changes that were not available when the manuals were finalized. Change are organized by manual.

Updates to *VR-Forces User's Guide*

- The appendix that listed all windows and dialog boxes has been removed from *VR-Forces User's Guide*. This information is still available in the VR-Forces online help.
- Book-level topics in online help now have text associated with them, rather than a generic explanation of how to use online help.
- The Drawing Mode tab of the Terrain View Options dialog box has a check box that is not documented in *VR-Forces User's Guide*. When the check box, Use Default Soil Colors, is selected, polygons are drawn with the color associated with the soil type. When the check box is cleared, the polygons are displayed using the color that was assigned to the polygons when the database was imported to GDB. These hard-coded colors do not necessarily correspond to the soil type and polygons with the same soil type may have different hard-coded colors.

  The Use Default Soil Colors check box is not available unless the Polygons check box is selected.
- The subsections in Section 4.10.3 in *VR-Forces User's Guide* describes the command line options for HLA individually. However, if you specify a version of the RPR FOM (`--rprFomVersion`) other than the default (1.0), then you must also specify the appropriate FED file (`-F` option.) You may also need to specify other command line options, such as the federation name, to correctly run the application.
- *VR-Forces User's Guide* and online help incorrectly refer to the Checkpoint Status toolbar. This toolbar has been removed. Checkpoint status is displayed in the status bar.

Configuration Guide Changes

- The parameter and component descriptions have been removed from *VR-Forces Configuration Guide*. Parameter and component descriptions are now available in the online help for the OPD Editor. The information is generated from header files and should be more accurate and complete than the previous printed documentation.
- The last bullet in the description of support for MetaFlight files in Section 2.5.6 of the PDF version *VR-Forces Configuration Guide* should read as follows:

  If the <Volume> and <Rootpath> entries in the <GeometryGridDataset> tag are set, the loader uses their values to build a full path to search for the files. If they are empty, the loader looks for the files relative to the location of the *.mft* file.

  This bullet is correct in the printed version of the manual.
- To select polygons in the TDB Tool, you have the following options, in addition to those described in Section 2.3.3 of the PDF version of *VR-Forces Configuration Guide*:
  - Control-click or control-draw bounding box to select multiple individual polygons, areas, or both.
  - Alt-click or alt-draw bounding box to remove polygons from the selection.

Section 1.3.2, Configuring Statistics Reporting

You can enable logging of statistics per scenario (all runs, from the opening of the scenario to the close of the scenario). Set the **logFrameRateStatistics** parameter to 1 in *vrfSim.mtl*.

Section 1.10. Configuring Spatial Subdivisions

Setting the spatial subdivision settings through the GUI may be computationally expensive, and doing so during runtime may affect performance, because the simulation will wait until the subdivisions are reconfigured. It is best to do this while the simulation is paused.

The settings are not saved with the scenario or with the GDB file, so they will have to be reset. To set the terrain's spatial subdivision such that's saved with the GDB, the users should use the TdbTool.

Section 2.5.1. Database Formats Supported by the TDB Tool

The TDB Tool supports Shape files.

Section 2.5.2 Geocentric Terrain Databases

Because geocentric databases are only approximations of the curved surface of the earth, there can be odd interactions between the ground and object models if the sizes of the polygons are very large. Only the vertices are actually at the correct altitudes in a geocentric terrain, and as such, the interior of the polygons may report altitudes significantly lower than expected. This is an inherent problem in approximating a curved surface with flat surfaces, similar to trying to approximate a circle with an octagon.

Section 2.5.4 Importing Vector Data into GDB

The way that vector data is clipped has changed. The previous algorithm would improperly clip areal features to the terrain resulting in new segments in the terrain that potentially cut across the terrain. It also failed to process linear and areal features that were composed of points that all were outside the terrain extent, yet a segment of the feature intersected the terrain's extent. In this release, any areal feature that intersects the terrain in any way is left unclipped. Additionally, intersections of features segments and the terrain are properly detected and processed the same as all other intersections. Clipping is performed by default when importing vector data.

Planarizing Vector Data

This section is incorrect. Vector data is planarized by default. Planarizing data that does not need to be planarized should not cause any problems, but not planarizing malformed vector data results in an unusable vector network and system instability if the malformed vector data is used by LOS queries or path planning.

Section 2.5.5 Importing OpenFlight Data

The TDB Tool now supports switch nodes, and there is a command line parameter available to specify which child node you want imported.

We now support Flat Earth natively, so it is imported correctly.

Section 2.5.7 Optimizing Databases Using the TDB Tool

This isn't that important for this release, but we should move away from saying intersection tests, and more towards saying query. So we would query the terrain for the altitude, not perform an intersection test for example. Not that big of a deal, but it might be helpful should we extend the abstraction more in the future.

TDB Tool Command Line Options

This section describes command line options for the TDB Tool that are missing from *VR-Forces Configuration Guide* or are incorrect.

`-a` *`minLodDistance`* – sets the minimum LOD distance to the specified value (double). It is only useful when importing OpenFlight files.

`-h` | `--help` – displays the help screen.

`-i` *`filename1 ... filenamen`* – specifies the name or names of a vector file or files to import. All files must be of the same type – either DFD or DFAD files.

`-k` *`numLodLevelsToKeep`* – specifies the number of levels of references files to keep as externally referenced files. The number must be a non-negative integer. Any referenced file nested deeper than this limit will be loaded into the file in which it is referenced. This must only be used when importing OpenFlight files.

`-m` *`filename1 ... filenamen`* – specifies a list of GDB files to create a master GDB for. This will output a GDB (with the name specified by the `-o` parameter) that references all files specified as parameters to this argument.

`-o` *`filename`* – specifies the output filename of the resulting GDB file.

`--switchchild` – specifies which child node of a switch node to import for that node. Defaults: the first node. Only relevant when importing an OpenFlight file.

Section 2.6. Terrain Databases Provided with VR-Forces

Makland-geocentric.gdb does not have vector data.

Section 3.1. Terrain Metafiles

The TDB Tool supports ASCII files.

MAP files have the following purposes:

- To specify records, which specify vector or terrain data to import into GDB format and how to import the data. As part of this process, the MAP file could also simply reference an existing GDB file as well as the import options, usually optimizations or transformations, to perform when loading the specified native GDB file or files.
- To specify an image or images to associate with a specified terrain database.

I'll cover detailing the individual parameters of the metafile records in another email, but some extra information is here:

Section 3.1.1. DTED Terrain Database Records

The following parameters have the indicated default values:

- LowColor: 0, 100, 0, 0
- High Color: 250, 250, 255, 0
- Low Color Height: 0.0
- High Color Height: 3000.0
- Origin latitude and longitude: 0 degrees.

Section 3.1.2. FLT Terrain Database Records:

OpenFlight terrain metafile records have the following new parameters:

- **NumberOfExternalLevelsToKeep** – the number of external levels to maintain as separate, referenced GDB files when the OpenFlight data is imported. Default: 0.
- **LodDistance** – Default: 0.
- **LowLodDistance** – the low level of detail that is used to generate alternate representations for DtFileNodes. Set to less than 0 to disable. Default: -1.0.
- **DefaultSwitchNodeChildIndex** – the index of the switch nodes to import. Default: 1 (first child).

GdbProcessingMetafileRecord

This record type makes most tdbtool options available for metafile records. This is used as a post-processing step when all imported data is in GDB format, or when a native GDB file has been loaded.

The parameters are as follows:

- **SpatialSubdivision** – specifies whether or not to have a spatial subdivision. Default: True.
- **X_Resolution** – specifies the x resolution of the spatial subdivision. Default: 100.
- **Y_Resolution** – specifies the y resolution of the spatial subdivision. Default: 100.
- **Z_Resolution** – specifies the z resolution of the spatial subdivision. Default: 0.
- **BalanceTerrain** – specifies whether or not to balance the terrain node tree. Default: false. Should almost never be true.
- **BalanceLeafing** – specifies the leafing factor to use when balancing the terrain. Default: 4.
- **BalanceBranching** – specifies the branching factor to use when balancing the terrain. Default: 4.
- **ReduceTerrain** – specifies whether or not to eliminate duplicate vertices and surfaces and remove other redundant or unnecessary terrain information. Default: True.
- **ReductionTolerance** – used when eliminating duplicate vertices. Default: 0.0001.

- **ApplyCoordSysNodes** – specifies whether or not coordinate system nodes in the terrain database should be applied
- **ConvertToGeocentric** - Specifies whether or not to convert the terrain to geocentric coordinate system. Default: false.
- **ClipVectorData** – Specifies whether or not to planarize the vector data in the imported or loaded file or files. Default: true.
- **PlanarizeVectorData** – Specifies whether or not to planarize the vector data in the imported or loaded file or files. Default: false.
- **PlanarizeExcessiveEdgeCount** – the planarization excessive edge count. Default: 50000.
- **PlanarizeTolerance** – the planarization tolerance. Default: 0.0001.

This metafile record is used to initialize a *DtTerrainDatabaseToolConfiguration*, which is then passed to the terrain database tool when post-processing the imported data.

Section 3.1.7 Extent Records

The records can have a non-zero altitude as the z parameter., but if they specify different values, the resulting terrain will be malformed (polygons will not have adjoining vertices.)

You can now specify color and roughness using the Color and Roughness keywords.

Developers' Guide Changes:

Any class or file type in the Terrain API that is documented as beginning with Pv, should begin with Dt.

Section 12.2.2 Terrain Intersection Tests

The sentence, "Terrain intersection tests are also made in the target detection sensor, to determine if an entity has line-of-sight to a target." should say: "The PhysicalWorld also performs terrain intersection queries, if configured to, when performing such operations as line-of-sight determination and determining the terrain height at a specified location. Nothing really should be directly accessing the terrain database directly, but instead should be going through the physical world.

We strongly recommend that you not access the terrain database directly.

Bug Fixes

This release fixes the following problems:

- Previous versions of VR-Forces did not automatically read nested *DtReaderWriter* files correctly, if filenames were being specified using relative paths. When traversing a nested structure of files for reading or writing, it now correctly determines the relative path at each level.
- Dismounted infantry no longer rapidly toggle between a static posture (kneeling) and a moving posture (running) at the end of a movement task. Dismounted infantry also no longer toggle back and forth between a small non-zero and zero magnitude rotational velocity. At the end of a movement task, the entity now comes to a complete stop and stays in the static posture.
- Aggregates are now able to finish a wait duration task.
- VR-Forces entities now interact properly with non-VR-Forces entities after a scenario rewind.
- The back-end no longer crashes when an aggregate subordinate is destroyed. This crash was noticed specifically when an aggregate was tasked to patrol a route.
- DI no longer automatically try to remap from one static posture to another while a scenario is playing. Previously, the lifeform model would automatically remap certain static postures while in play mode. Now the remapping between postures only occurs when the entity transitions from a static state to a moving state or vice versa.
- DtAggregateStateRepository::findPendingSubtaskIds() no longer returns true when a subtask ID is not in the component's list of pending subtask Ids.
- VR-Forces now correctly commands the turret to be slewed to face the target direction when firing.
- Subordinates that are independently tasked, and are subsequently sent a 'set tasked by superior' message, now respond properly to other Set Data Requests without first receiving a new task via the aggregate. Now subordinates resume being tasked by superior by default at the end of their task, so there is no need to automatically call set tasked by superior to resume their participation in aggregate tasks and set data requests.
- The damage adjudication actuator, and individual damage adjudication actuator, were not correctly reregistering for detonation notifications when the entity was set restored directly, instead of through a setRestored message. This would cause entities restored directly to become "invincible." This has been fixed.
- Fixed wing entities no longer jump to zero altitude when they are destroyed and crash into the ground.
- The function stripBaseName() no longer crashes if given a filename as a parameter that doesn't contain any path information (just a filename). It now handles both cases.

- For entities with multiple weapon systems (such as an M1A2), an entity's rules of engagement state is now restored to the value it had prior to executing a fire-now request. Previously, it would always be set to "fire-now" after executing the fire-now request.
- Entities that have multiple weapon systems (such as the M1A2, that has a main gun and a secondary gun), will now execute fire-now requests that require the use of anything other than the first weapon system that appears in its state repository. Previously, only the first weapons system would successfully execute a fire now request. If a target of a type handled by a secondary gun was specified, that fire now request would always time-out. For example, a fire-now request issued to an M1A2 to fire on a dismounted infantry will now work correctly. It will use its secondary gun, the machine gun, to fire on the entity.
- Non-projected DTED terrains are no longer processed twice.
- Surface entities can now handle higher tick intervals without becoming unstable. In previous versions, a tick interval of 3 seconds was enough to make the guided missile destroyer surface entity unstable (it would rotate wildly when executing movement tasks). The surface entity model has been modified to clamp rotational yaw acceleration such that it never results in a yaw velocity that exceeds the maximum yaw velocity for the entity in a given tick.
- Control objects are now created with a correct 'closed-figure' flag. In previous versions of VR-Forces, all control objects created through the GUI were created specifying the 'closed-figure' parameter as true. Now only circle, area, point, and obstacle control objects are created with this flag set to true. All other types of objects (such as routes and phase lines) are created with this flag set to false. The resulting value for this flag can be viewed for a given object in the order of battle file.
- Turrets that are mounted on other turrets are now aimed correctly by the turret-slew and turret-scan controllers.
- Aggregates no longer publish updates too often as a result of padding data for the variable datum entries being uninitialized. Padding bits are now zero, and updates are sent only when actual data changes.
- Emitter systems now stop publishing if their host entity gets destroyed in the simulation.
- Whether a detonation is from a missile or not is now based on munition type, and is not dependent on the presence of a missile object existing in the simulation.
- Remote non-VR-Forces entities no longer appear to lag behind the position where they are expected to be (HLA only). This problem could be seen when tasking a VR-Forces entity to follow a remote non-VR-Forces entity. The VR-Forces entity would follow at a distance farther than the one specified in the Follow Entity task.
- All bodies of water in *berkeley.gdb* are now mapped to their proper soil type. Ocean and lakes are now Ocean and Lake soil types, not unspecifiedBodyOfWater types Vehicles will no longer drive onto bodies of water.

- The Detonation Manager now correctly calculates and fills in the bounding volume face and the angle of incidence of indirect fire detonations in comparison to target entities. Previously, it always defaulted to front-face and 0 angle of incidence.
- Initialization and accessor bugs in the myCapabilities member in *DtVrfObjectStateRepository* have been fixed.
- Ordered speed is now saved as part of an aggregate's state.
- When an aggregate completes a Move Along task, DtAGgregateMoveAlongController now sets the ordered speed for subordinates to the ordered speed of the aggregate. Previously, if a subordinate's speed was changed during execution of the Move Along task to maintain formation, it wasn't reset at the end of the task.

Known Problems

VR-Forces Release 3.9 has the following known problems:

- If you are using VR-Forces in a DIS exercise, you may experience problems with dropped packets and entities may time out. To avoid this problem, you can use asynchronous I/O by starting with the `-I` parameter.
- Aggregates composed of fixed-wing entities can only perform limited behaviors as an aggregate.
- When you aggregate a set of aggregates into a higher-echelon aggregate using the Aggregate As menu item, make sure that the aggregates being combined have been collapsed to their highest level. Failure to do so will aggregate the entities at the level they are currently displayed. For example, suppose that you want to aggregate two teams of two tanks each into a platoon. If at the time of the Aggregate As operation, one of the teams is expanded, displaying its constituent tanks, while the other team is collapsed, displaying a single team icon, the resulting platoon will be composed of two individual tanks and a team, instead of two teams.
- Fixed-wing take-off, landing, and taxiing behaviors may be difficult to configure through the front-end when simulating on geocentric terrain databases. This is due to the fact that it uses a different projected terrain database in the front-end when simulating on a geocentric terrain database in the back-end. Correlation errors in terrain elevation between the two databases must be small (on the order of meters) when placing waypoints or routes on the ground to support fixed-wing ground behaviors. Either use high resolution terrain databases (minimizing the elevation error between the front-end and back-end databases) when configuring these behaviors, or restrict your scenarios to fixed-wing flying behaviors.
- Using the *tdbtool* 's balancing option (`-b br,lf`) to balance certain terrains may cause the generated *.gdb* file to display incorrectly. This has to do with a draw ordering bug that is not currently corrected for all terrains. The resulting GDB file will still give correct intersection information, so it will still be suitable for simulation purposes.
- Vector clipping does not work properly with areal features.

- Surface entities are not configured to fire weapons. You can customize the object parameters database entries to enable this.
- The Qt translation files (in *./translations*) for built-in Qt-level GUI items do not work properly. The translation files for VR-Forces work correctly.
- If you are running VR-Forces on Linux with the DMSO RTI, you cannot run in combined mode. Start the GUI and back-end separately.
- The order in which you create plans for aggregate units and individual members of aggregate units affects how the plans are implemented. If you create a plan for an aggregate member and then create a plan for the aggregate, the aggregate plan overrides the individual plan. If you create a plan for an aggregate and then create a plan for a member of the aggregate, the member plan is executed.
- If you run a MÄK Logger recording of a VR-Forces simulation and the site:application number for the back-end is the same as the site:application number of the back-end used to create the recording, VR-Forces may crash. To avoid this problem make sure that you run VR-Forces with different site:application numbers, or just run the front-end and use it as a plan view display to view the recording.
- The component descriptions in Appendix C of *VR-Forces Developer's Guide* have not been updated to reflect changes made in recent releases of VR-Forces. In particular references to port interfaces are obsolete and the component descriptions do not reflect the use of process state repositories. Fixed-wing and rotary-wing components may be significantly incorrect.
- Using large filled polygons, such as minefields, in overlays can degrade performance.
- You cannot load vector data with geocentric databases. Vector data works only with geodetic or UTM databases.
- You cannot give a helicopter a Set Altitude request that places it at less than the starting height value.
- Plans stop executing if they encounter a task that they cannot execute. For example, if the plan for a rotary-wing entity include Turn To Heading, which is not a valid task for rotary-wing entities, the plan stops executing.
- If you delete an entity that has a plan and create a new entity with the same name as the deleted entity, the old plan is displayed in the new entity's Edit Plan window. However, the new entity cannot execute that plan.
- Missiles get initialized at their maximum velocity. They do not accelerate to maximum velocity.
- Torpedoes ignore a polygon with soil type water, but the intervisibility fan does not. So it does not show an accurate representation of who can see who.
- When a subsurface entity gets destroyed it jumps to altitude 0.
- It you task a rotary-wing entity to fly to a waypoint on the ground, it will land when it gets to the waypoint. If opposing force trucks get within range of the rotary wing it will fire. Since the rotary-
- Intervisibility works with linear and areal features, but not with point features.

- Occasionally, VR-Forces is unable to load a scenario. When this happens, delete the contents of the *./data/temp* directory and try loading again.
- Setting the heading of a rotary wing entity whose altitude is less than the **starting-height** parameter causes it to rise up to the **starting-height** value. The workaround is to set a very small (non-zero) **starting-height**.
- You cannot use translation files to translate entity labels.
- You cannot use translation files to translate labels in the Create Entity menu (*vrf.ent*).
- The TdbTool does not convert images to Lambert Conformal Conic correctly.
- If the Stealth is attached to an entity in tether, tether track, or mimic track mode, you cannot drag the Stealth icon to a new location.
- Rotary-wing entities cannot embark on surface entities that are moving.
- Someitmes rotary-wing entities crash when embarking on aircraft carriers.
- VR-Forces may consider an entity to have arrived at a location, or to be embarked before the entity actually stops moving. Therefore, if you put embark or disembark tasks in a plan and these tasks are contingent on the parent entity arriving at a particular location or embarking on another entity, you must insert a wait duration statement before the embark or disembark statement to ensure that the parent entity has come to a stop before the task is implemented. The amount of waiting time required will vary, so you will have to experiment. See the embarkdemo scenario for an example of using a wait statement to ensure that a parent entity has come to a complete stop before the embarked entity disembarks.
- If you drag a toolbar onto the Utility Toolbar, you will not be able to dock that toolbar anyplace else in the VR-Forces window.