MÄK Plan View Display 2.9 Release Notes

This release note contains the following release-specific information for Plan View Display Release 2.9:

- Systems Supported 3
  - Patching the Microsoft Visual C++ 6.0 Compiler 3
  - Building on Linux 4
  - Disk Space Requirements 4
  - Compiler Compatibility on Windows 4
  - MÄK Product Compatibility 4
  - Qt Release Compatibility 4
  - Third-Party Library Requirements 5
- Network Compatibility 5
- RPR FOM Versions Supported 6
- New Features and Updates 7
- Changes to the Plan View Display Application 8
  - Startup and Configuration Changes 8
    - A Default Map is Opened if None is Specified 8
    - Changes to Configuration Files 9
    - New Parameter in symbolmap-ttf.mtl 9
    - International Language Support in Map Area, Menus and List Boxes 9
    - New Startup (Command Line) Options 10
    - Font Size Setting for Linux 10
    - Ability to Change Time Stamp to Absolute from Relative 11
  - Display and Display Options 11
    - Toolbars and Menu Options have been Reorganized 11
    - Iterating through the Entity and Object Lists 12
    - Show or Hide Stealth Icon 12
    - Zoom to Extents 12
    - Overlay Label 12

Document ID: PVD-2.9-2-060222

New Emitter Beam Option 12
Displaying Entity Information 12
Processing Signal Interactions 13
Overlay and Terrain Database Changes 13
Undo/Redo Support 13
Geotiff Support 13
Save Overlays in Geocentric Coordinates 13
Multiresolution Paper Map Addition 13
GUI API Changes 14
Showing and Hiding Symbols 14
Overlay View Manager 15
View Menu 15
Help Menu 15
Reworked Echelon Panel 15
Tdb and Papermap Drawers 17
DtTerrainLocationEntryWidget 17
Setting Font Size on Linux 17
The Entity List and Environmental List are Toolbars 17
DtNameLabelSymbol Adds Layer Member 18
dropEvent() Renamed 18
Terrain Selection Changed 18
Changes to DtTerrainLineConfigWidget 18
Terrain API 19
Openflight Reader 19
MetaFlight Reader 19
Changes to Shapefile Support 20
Changes to GDB 20
FltMetafileRecord and DfdDfadMetafileRecord Additions 20
DtExtentInitializer Update 21
GDB File Format Changes 21
DtGdbNode Class Hierarchy Interface Changes 21
DtTerrainDatabase Interface Changes 22
Coordinate System Class Hierarchy Changes 22
Library Changes 22
Documentation Updates 23
Bug Fixes 28
Known Problems 28

Systems Supported

Plan View Display 2.9 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 Plan View Display 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 .NET 7.1

If you are building Plan View Display with .NET on Windows 2000, you must install the latest version of the directX SDK (obtainable from Microsoft).

Patching the Microsoft Visual C++ 6.0 Compiler

The Plan View Display 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 Plan View Display 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 Plan View Display 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 Plan View Display requires approximately 446 MB of disk space. Terrain databases use approximately 137 MB and documentation (primarily the class reference) uses 89 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, for example, the Plan View Display 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 the Plan View Display, you must link with VR-Link 3.9.6. The Plan View Display does not include a VR-Link developer's license. You must purchase it separately. If you are building for HLA and want to link with the MÄK RTI, use MÄK RTI 2.x.

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

Qt Release Compatibility

If you want to do development using the Plan View Display API, you need to use Qt, a cross-platform GUI toolkit from Trolltech. The Plan View Display GUI was built with Qt release 3.3.5. The Plan View Display does not include a Qt license. You must purchase a license from Trolltech.

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

Plan View Display 2.9 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 the Plan View Display with the DMSO RTI, contact MÄK technical support.

Plan View Display 2.9 is compatible with applications that use earlier versions of VR-Link if they support versions of the RPR FOM listed above.

DIS only

Plan View Display 2.9 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.

RPR FOM Versions Supported

Plan View Display 2.9 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, Plan View Display 2.9 uses RPR FOM 1.0.

If you are building the Plan View Display and you want to specify a version of the RPR FOM through code, pass the version number (for example, 2.0006) to the *DtRprFom-Mapper* 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 Plan View Display 2.9 relies on the LgrControl and ViewControl custom MÄK extensions.

New Features and Updates

This release of MÄK Plan View Display has the following updates and new features:

- Startup and configuration changes
  - A default map is opened if none is specified
  - Changes to configuration files
  - New parameter in *symbolmap-ttf.mtl*
  - International language support in map area, menus and list boxes
- New startup (command line) options
  - Font size setting for linux
  - Batch Scenario Files
  - Ability to change time stamp to absolute from relative
- Display and Display Options
  - The Plan View Display toolbars and menu options have been reorganized
  - Iterating through the entity and object lists
  - Show or hide stealth icon
  - Zoom to extents
  - Overlay label
  - New emitter beam option
  - Displaying entity information
  - Processing signal interactions
- Overlay and terrain database changes
  - Undo/redo Support
  - Geotiff support
  - Save overlays in geocentric
  - Multiresolution paper map addition

GUI API Changes

- Overlay view manager
- View menu
- Help menu
- Reworked echelon panel
- Tdb and papermap drawers
- DtTerrainLocationEntryWidget
- Object mapping classes
- Setting font size on linux

- Echelon panel is now a docked toolbar
- Entity list and environmental list are now a docked toolbar
- Overlay objects that are considered control objects
- DtNameLabelSymbol adds layer member

Terrain Toolkit Changes

- Openflight reader
- MetaFlight reader
- Changes to shapefile support
- Changes to GDB
- FltMetafileRecord and DfdDfadMetafileRecord additions
- DtExtentInitializer update
- GDB changes
- DtGdbNode class hierarchy interface changes
- DtTerrainDatabase interface changes
- Coordinate system class hierarchy changes
- Library changes

All new GUI and configuration features are fully documented in *MÄK Plan View Display User's Guide*.

Changes to the Plan View Display Application

This section describes changes to the Plan View Display GUI and configuration. These changes are documented in *MÄK Plan View Display User's Guide*.

Startup and Configuration Changes

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

A Default Map is Opened if None is Specified

If you do not specify a scenario or database when Plan View Display starts up, it loads a default database, *default.map*.

Changes to Configuration Files

- The parameters in *pvd.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 prefix has 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 the Plan View Display to release 2.9.
- **hideOffScreenSymbols** is now set to true by default for performance reasons.
- You can specify the FED file to use with the **fedFile** parameter in *pvd.mtl*.
- A new variable, **processSignalInteractions**, in *pvd.mtl*, enables or disables processing of signal interactions for transmitters.

New Parameter in symbolmap-ttf.mtl

There are currently two optional parameters that can be specified after the font key used to display on the map area. One is the optional background character and the last is a color and transparency list for the font. A new optional parameter has been added before those two parameters which is 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). If there are any entries in *symbolmap-ttf.mtl* that you have modified to include a background character, make sure to put 1.0 before this number to keep the scale correct

International Language Support in Map Area, Menus and List Boxes

Plan View Display now supports international language for text items that can be entered via dialog boxes (names, labels, overlay tab names) as well as items found in files such as 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.

New Startup (Command Line) Options

Plan View Display has the following new startup options:

- The `-F ttl` option is no longer supported. To specify mcast time-to-live, use the the `--mcastTtl` option.
- The Plan View Display 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.

Ability to Change Time Stamp to Absolute from Relative

Added **useAbsoluteTimeStamps** variable (DIS only) to the *pvd.mtl* to allow user to change from relative to absolute timestamps

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 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 the Embarkation View and Selection Groups. 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 shown, the items in the overlay tab layer are visible. If the lock is opened, 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 Terrain Database," on page 14-2, in *MÄK Plan View Display User's Guide*.
- The Object Count toolbar is a new toolbar that displays the number of entities, aggregates, and control objects in the exercise.
- 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 Point," on page 14-3, in *MÄK Plan View Display 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 (<) .

The Plan View Display 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 *MÄK Plan View Display 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 group 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 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

Plan View Display supports Undo and Redo for changes to object location (movement, vertex movement, vertex addition or deletion). See "Undoing Object Vertex Edits," on page 9-21 in *MÄK Plan View Display 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 out 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 Plan View Display 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.

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 this symbol. The setKeepHidden() member function, introduced in release 2.9, 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 this release, 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().

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).

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.

View Menu

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 DtAppHelpMenu.

Reworked Echelon Panel

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

- The *DtPvdEchelonPanel* class has been removed and replaced with *DtPvdEchelonView*
- In the *DtPvdEchelonViewHandler*, 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 *DtVRfEchelonTree*, panel or view, the following changes need to be made. 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);
```

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().

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 10.

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.

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 *DtPvdGuiDragDropHandler* 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.

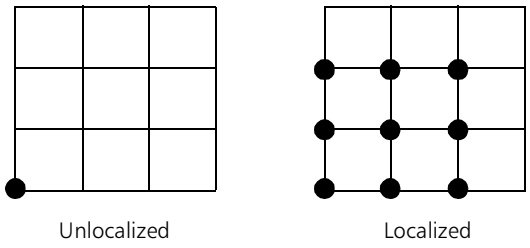

Figure 1. Unlocalized and localized terrains

MetaFlight Reader

Plan View Display 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.
- It solves filename patterns that contain DSNAME, DSTITLE, LEVEL, COL, and ROW entries in the MetaFlight files.

- 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 Update

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.

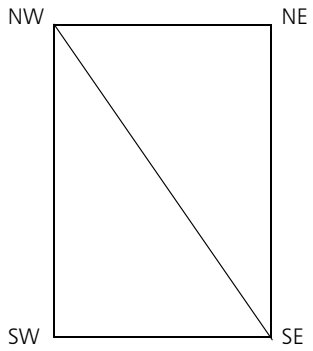

Figure 2. Extent polygons

You can now specify a maximum latitude or longitude that is smaller than the minimum latitude or longitude. Plan View Display 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

An important change to the terrain database interface, and indeed almost all of the terrain API, was the correction of const member functions that returned non-const references or pointers to internal members. These have been replaced with two functions, a const 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 Plan View Display 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.

Documentation Updates

The PDF version of *MÄK Plan View Display User's Guide* has been updated for this release.

- The appendix that listed all windows and dialog boxes has been removed from *MÄK Plan View Display User's Guide*. This information is still available in the Plan View Display 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 *MÄK Plan View Display 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 4.6.3 in *MÄK Plan View Display 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.

Changes to TDB Tool and Terrain Metafile Documentation

- The last bullet in the description of support for MetaFlight files in Section 14.5.6 of the PDF version of *MÄK Plan View Display User's 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 14.3.3 of the PDF version of *MÄK Plan View Display User's 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 14.5.1. Database Formats Supported by the TDB Tool

The TDB Tool supports Shape files.

Section 14.5.2 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 14.5.4 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 14.5.5 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 14.5.7 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 14.6. Terrain Databases Provided with VR-Forces

Makland-geocentric.gdb does not have vector data.

Section 15.1. 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 15.1.1. 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 15.1.2. 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 15.1.7 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.

Terrain API Changes

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

Bug Fixes

This release fixes the following problems:

- Non-projected DTED terrains are no longer processed twice.
- 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 unespecifiedBodyOfWater types.

Known Problems

Plan View Display Release 2.9 has the following known problems:

- If you are using the Plan View Display 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.
- 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.
- The Qt translation files (in *./translations*) for built-in Qt-level GUI items do not work properly. The translation files for the Plan View Display work correctly.
- 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.
- 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. 5335
- Intervisibility works with linear and areal features, but not with point features.
- You cannot use translation files to translate entity labels.
- If you drag a toolbar onto the Utility Toolbar, you will not be able to dock that toolbar anyplace else in the Plan View Display window. 5080