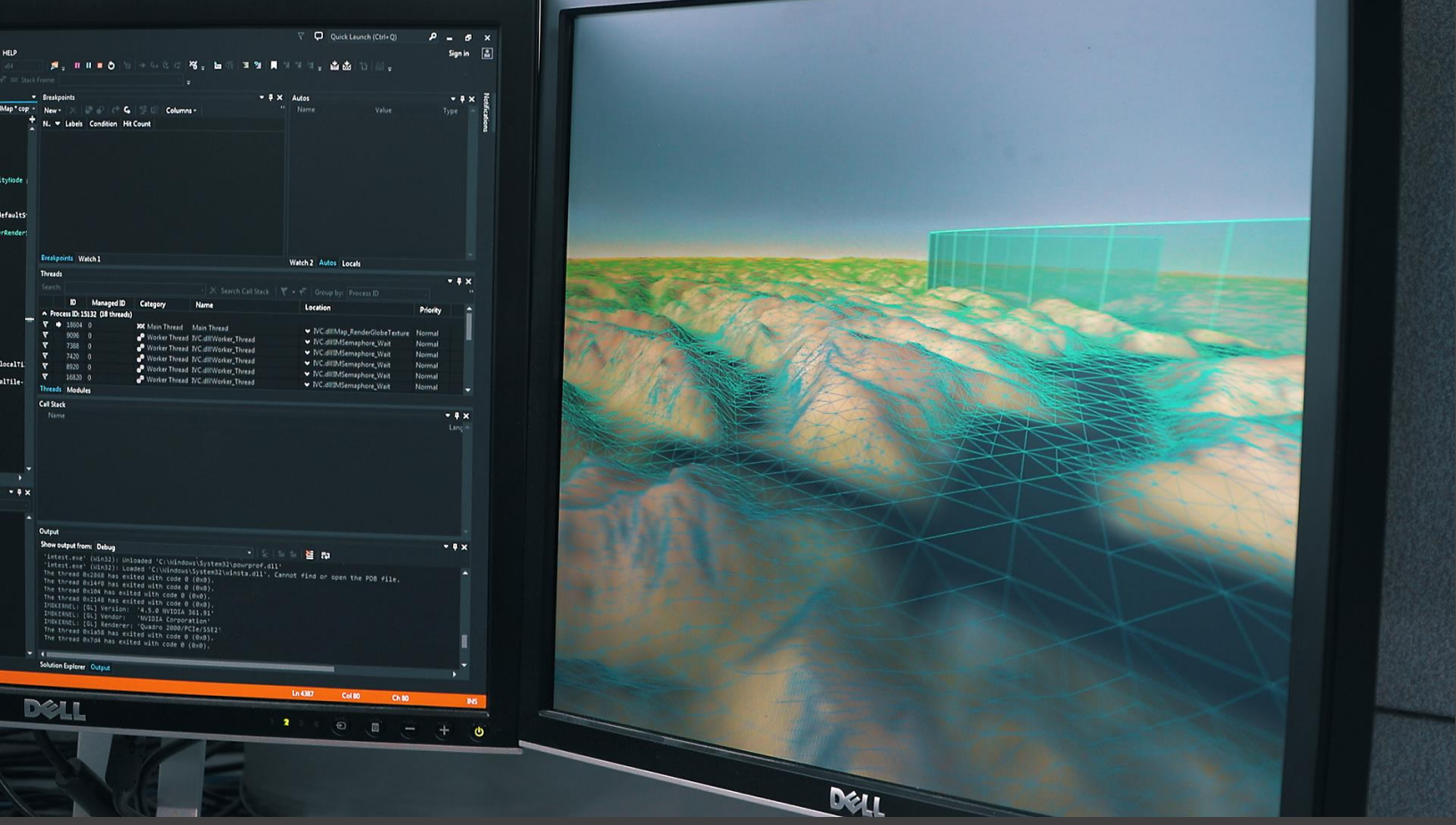

TerraLens

Release Information Manual

Version 9.3

Kongsberg Geospatial

Table of Contents

Table of Figures

Preface

About the TerraLens® Platform

TerraLens, formerly sold as InterMAPhics®, is available as a family of products:

TerraLens Core — Core SDK and developer libraries

TerraLens Server — OGC Map tile server, optimized for performance

TerraLens UI — HMI / UI development foundation with multi-touch support

TerraLens Mobile — Mobile SDK for Android apps

TerraLens Web — WebGL developer framework

TerraLens Creator — Advanced map styling and packaging toolkit

While the product names and packaging have changed, the core functionality has not. The version numbering is maintained from the former InterMAPhics product library, and many of the internal libraries and namespaces still reference InterMAPhics.

You can view a full breakdown and description of the TerraLens geospatial platform and products on the Kongsberg Geospatial website at: https://www.kongsberggeospatial.com/products.

In This Manual

This manual is provided with each release of TerraLens and provides information to enable users to identify information relevant to TerraLens releases. This includes:

- Components – Major components in TerraLens
- Supported Platforms – A list of currently supported TerraLens platforms
- New Features - Details on the new features that were added in this release of TerraLens
- Problems Addressed - Information on problems that were addressed in this release of TerraLens
- Limitations - Information on known limitations in TerraLens
- Known Problems - Provides information on known problems in TerraLens

Intended Audience

This document is intended for use by a TerraLens application developer with access to TerraLens software and the corresponding document set.

The information is intended for users who are upgrading from previous versions of TerraLens, and is also useful for new users.

Note About Namespaces: While InterMAPhics® has been rebranded as TerraLens®, the TerraLens source code maintains the original **InterMAPhics** class names and namespaces for backward compatibility and consistency with older documentation. Wherever the word "InterMAPhics" occurs in this document it is referring to the proper name of a class or class namespace.

1 Product Components

The major components contained in TerraLens are as follows:

Component	Description
Licensing	RLM components that provide licensing services to the TerraLens API.
TerraLens Creator	An application that provides map data management, definition of visual attributes, preview, and preparation for use in TerraLens applications.
Documentation	All non-API TerraLens documentation (Release Information Manual, Product Overview, Programmer's Guide).
TerraLens API	Components for development, including header files, libraries, and class library documentation.
Reference Application	A reference application including code, header files, executables, and documentation.
Reference Application Resources	Resources for the reference application including map data.
Sample Applications	Application modules and accompanying documentation highlighting key TerraLens concepts.

Table 1: Major components contained in TerraLens

2 Supported Platforms

Operating Systems, Compilers, and Programming Languages

TerraLens supports a variety of operating systems, compilers, and programming languages as shown in the table below.

Kongsberg Geospatial regularly adds support for additional configurations. Please contact Kongsberg Geospatial Customer/Technical Support for information on any configuration not shown.

Operating System	Arch	Compiler	API
Microsoft Windows 10	x86 x86_64	Microsoft Visual Studio vc14.2 (2019) Microsoft Visual Studio vc14.1 (2017) Microsoft Visual Studio vc14 (2015) Microsoft Visual Studio vc12 (2013) 32 bit/64 bit versions	C++
		Sun JDK 1.8; 32 bit/64 bit versions	Java
		Microsoft Visual Studio vc14.2 (2019), .NET 4.7.2 Microsoft Visual Studio vc14.1 (2017), .NET 4.6 Microsoft Visual Studio vc14 (2015), .NET 4.5.2, 4.6 Microsoft Visual Studio vc12 (2013), .NET 4.0, 4.5 32 bit/64 bit versions	CLI
Microsoft Windows 7	x86 x86_64	Microsoft Visual Studio vc14.1 (2017) Microsoft Visual Studio vc14 (2015) Microsoft Visual Studio vc12 (2013) 32 bit/64 bit versions	C++
		Sun JDK 1.8; 32 bit/64 bit versions	Java
		Microsoft Visual Studio vc14.1 (2017), .NET 4.6 Microsoft Visual Studio vc14 (2015), .NET 4.5.2, 4.6 Microsoft Visual Studio vc12 (2013), .NET 4.0, 4.5 32 bit/64 bit versions	CLI
Red Hat Enterprise Linux WS8	x86 x86_64	GNU GCC 8.2 32bit/64bit versions	C++
		Java JDK 1.8; 32bit/64bit versions	Java

Operating System	Arch	Compiler	API
Red Hat Enterprise Linux WS7	x86 x86_64	GNU GCC 4.8.2 GNU GCC 7.2 32bit/64bit versions	C++
		Java JDK 1.8; 32bit/64bit versions	Java
Red Hat Enterprise Linux WS6	x86 x86_64	GNU GCC 4.4.5; 32bit/64bit versions	C++
		Java JDK 1.7, Java JDK 1.6; 32bit/64bit versions	Java
Ubuntu Linux 18.04	x86-64	GNU GCC 7.3	C++
Android	ARM	Android 7.0	Java

Table 2: Supported Operating Systems, Compilers, and Programming Languages

The following table shows the operating systems and compilers maintained for legacy InterMAPhics versions; TerraLens 9 is not available on these systems.

Microsoft Windows 7 Microsoft Windows 8 Microsoft Windows 10	x86 x86_64	Microsoft Visual Studio vc11 (2012) Microsoft Visual Studio vc10 (2010)	C++
		Sun Sun JDK 1.7, Sun JDK 1.6	Java
		Microsoft Visual Studio vc11 (2012), .NET 4.0, 4.5 Microsoft Visual Studio vc10 (2010), .NET 4.0	CLI
Microsoft Windows XP	x86	Microsoft Visual Studio vc8 (2005)	C++
		Sun JDK 1.6	Java
		Microsoft Visual Studio vc8 (2005) – SP1	CLI
Microsoft Windows Vista	x86	Microsoft Visual Studio vc9	C++
		Microsoft Visual Studio vc8 (2005) – SP1	CLI
Red Hat Enterprise Linux WS5	x86 x86_64	GNU GCC 4.2, GCC 4.1.2	C++
		Java JDK 1.6	Java

Android	ARM x86	Android 4.0.3 Java JDK 1.7	Java
	ARMv7	i686-linux-android-c++ (gcc) 4.8	C++
Sun Solaris 2.9	SPARC	GNU GCC 4.3.3, Sun Studio	C++
Sun Solaris 2.10	x86	GNU GCC 4.3.3	C++
		Sun JDK 1.6	Java
HP-UX 11.23	Itanium	HP aC++ A.05.05	C++

Table 3: Legacy Operating Systems and Compilers

Graphics Hardware and Drivers

The 2D/3D displays in TerraLens are rendered using OpenGL and as such are hardware dependent. The graphics hardware and drivers may have different display and performance characteristics. Using the most up-to-date drivers for your graphics hardware will minimize the potential for graphics card problems.

For reference, TerraLens has been tested with the video cards listed in the following table.

Operating System	Video Cards
Microsoft Windows 10	NVidia GTX 660
	AMD Radeon RX 580
	AMD Radeon Vega 8
	Intel IRIS Plus 655
	NVidia Quadro P1000
Microsoft Windows 8	NVidia GTX 660
Microsoft Windows 7	ATI Radeon HD 5800
	NVidia GeForce 8800 Ultra
	NVidia Quadro 2000
Microsoft Windows XP	ATI Radeon 9700 Pro
	NVidia GeForce 7900 GS
Red Hat Enterprise Linux WS 7	NVidia GT 720
	NVidia GTX 1060
	NVidia GeForce GT 640
	Intel IRIS Plus 655
Red Hat Enterprise Linux WS 6	ATI Radeon XT 850 XT
	NVidia GeForce GTS 450
	NVidia GeForce GTX 280
	NVidia Quadro 600

Table 4: Graphics Hardware and Drivers

3 New Features

TerraLens 9.3

TerraLens 9.3 focuses on performance improvements, enhancements to WMTS and new presentations to enhance situational awareness. Key new features of the 9.3 release include:

- Presentations for Heat Maps, Cesium3DTiles, and Altitude Indication
- Enhanced support for web services
- MIL-STD-2525 D symbology
- Support for Microsoft Windows Visual Studio 2019 and Linux RedHat 8

TerraLens Visualizations

Heat Map Density Display

Heat maps provide a visual indication of the density of points in an area using a colored rasterization. The new `GeoHeatMap` and `WorldHeatMap` primitives provide application control over the color and intensity used for a density value.

World Width Lines for Primitives

The thickness of lines and polygons specified in geographic coordinates can now be set with a World Width value, for instance to represent the real-world dimension of an overlay. This cannot be applied to maps, and will override the outline settings if it is mismatched with the coordinate type used for the line width.

TerraLens Mapping Updates

Map Format Updates

Support for DAFIF map format has been extended to version 8.1. S-57 IENC (inland features) navigation levels 7-9 have been added to the existing support for levels 1-6. This additional support is seamless through both the MapSource and the legacy Mapping APIs.

Updates for Web Maps

Improved Handling of Server Capabilities

Improved handling of URLs for WMS (Web Map Service) and WMTS (Web Map Tile Service) is provided with specialized configuration classes, **WMSSourceConfig** and **WMTSSourceConfig**. Given a URL which resolves to a capabilities XML file appropriate for the format (WMS or WMTS), the class will discover the available map layers. The configuration objects can then be used to construct a MapSource hierarchy. The following example is for WMTS maps:

```
WMTSSourceConfig config(<URL_WMTS_CAPABILITIES.xml>);
MapSource * source = new MapSource(config);
std::vector<const MapSource*> features = source->getFeatureSources();
```

Improved Reference Systems and Projection Handling

When multiple reference systems are available, TerraLens will prefer CRS:84 as this most closely reflects our internal system and is therefore most performant.

TerraLens support is extended to the default projections provided by GeoServer, **EPSG:4326/EPSG:900913.**

Improved Logging

A **MapSourceMonitor** object can be added to WMTS or WMS MapSources to monitor and diagnose issues with HTTP requests. WMTS and WMS MapSources also feature improved logging for several common issues.

TerraLens Rendering

Monitoring the Viewport Update

The new method **Viewport::updateWithStatus()** indicates whether new data was actually rendered along with the update, or the viewport was merely updated i.e. because the data to be drawn had not changed. This can be helpful for applications that are rendering additional graphics on top of the TerraLens display, to only render their custom graphics if necessary.

Map data is loaded in a background thread during a viewport update call, so all map data is not necessarily displayed by the time the update returns. While this is not noticeable to most applications that are updating on a regular basis, sometimes an application needs to know whether the map loading is complete i.e. for drawing custom graphics on top of the TerraLens display. If the method **Viewport::setMapsLoadingTracking()** has been set to true, the application can query the current state using **Viewport::isMapLoadingComplete()**.

Performance

TerraLens performance has benefited from increased multithreading in the rendering kernel, and the batching of calls to the GPU. This will be especially noticeable in cases where 3D primitives render both fill and line styles.

Drawing in 3D will also be noticeably faster with refined selection of the mapping and terrain tiles required for rendering.

High Resolution 3D Views

As 3D views become more common in situation awareness applications, TerraLens has new methods to not only enhance 3D visualization options but also balance visual resolution with performance.

High Resolution Cesium3DTiles

TerraLens now supports Cesium3DTiles, a format used to display large datasets of 3D objects such as cities. The **Cesium3DTile** class can be added to a MapLayer for visualization in a 3D viewport, with methods to control the level of detail as required to meet display performance objectives, as well as the material applied to the object model to control visual aspects such as brightness and contrast. Like other visualizations within the MapLayer, the amount of detail used

in drawing the 3DTiles changes in response to the distance of the object from the camera i.e. higher resolution for the areas close to the camera, less for the areas further away, with the detail displayed changing as the display is panned and zoomed. Since elevation is included in the 3DTiles, terrain has no effect of the 3DTiles display. This is not a map format, and as such is not automatically discovered via the MapSource API and must instead be loaded explicitly via the Cesium3DTile class.

Elevation Visualizations

Visualizations are available to highlight altitude in a 3D display, providing additional situational awareness for height indication.

Elevation Warning

A configurable colorized elevation ramp visualization can be applied to primitives, terrain, and the Cesium3DTiles. Please see each of these classes for the appropriate API.

Increased Terrain Resolution

TerraLens uses elevation data from maps to build a terrain mesh in 3D, over which maps are draped. For performance reasons, the terrain mesh contains tiles of multiple resolution levels so that more detail is used closer to the camera and less for the area further away. For high resolution imagery such as that used to depict cities, this can result in a perceived loss of crispness as the image pixels are stretched between points in the terrain mesh. For applications in which precision is required over runtime performance, the terrain mesh size can be adjusted using:

```
Viewport3D::setTerrainMeshSize(int size)
```

The mesh size indicates the number of pixels across the tile, with a default value of 256. Higher values (up to 512) will improve the visualization of highly detailed terrain but will decrease the performance.

MIL-STD-2525D Symbology

The MIL-STD-2525 specifications define a common language of symbology for use across systems deployed by different manufactures. Each version of the MIL-STD-2525 documentation builds upon the previous version adding more symbols and features and, in some cases, changing or deprecating older symbols.

Previous versions of the MIL-STD-2525 specification relied upon ‘SIDC’ (symbol identification code) values defined by 15 alpha-numeric values. With the change to the MIL-STD-2525D specification the method to define symbology has changed to use 16 groups of numeric values that can range in size from one to three digits. Please see the MIL-STD-2525D documentation section A.5 for more information.

TerraLens supports the MIL-STD-2525 specifications by allowing applications to generate symbology defined in the documentation primarily by entering a ‘SIDC’ value and requesting the created symbol. This functionality is extended to support the MIL-STD-2525D specification by entering the ‘SIDC’ value as a string containing the numeric values. The relative position of the characters in the string are used to parse each group of numeric values. Note that the graphic libraries loaded are specific to the standard they were created for and the appropriate format of ‘SIDC’ value should be used for a given graphic library.

For example, to generate a friendly civil fixed wing aircraft using the MIL-STD-2525C graphic libraries a SIDC value of “SFAPCH” would be used. When using the MIL-STD-2525D graphic libraries a value “1003010000120100” would be used to generate the same symbol.

In some cases, it is desirable to omit specific fields. Some applications require tactical (dynamic) graphics not to have an identity or color applied to the graphic. To allow for this fields can be left as a dash (‘-’) to indicate no option while maintain the relative positions. When creating the dynamic graphic for ‘assembly area’ using the MIL-STD-2525C graphic libraries a SIDC value of “G-GPGAA” would be used while “10--250000150200” would be used for the MIL-STD-2525D graphic libraries.

Platform Support

TerraLens is available for Linux RedHat8 with gcc8.2, and Microsoft Visual Studio 2019 with vc14.2.

TerraLens 9.2

TerraLens 9.2 focuses on performance improvements, and new visualizations for 3D displays. Key new features of the 9.2 release include:

- Support for Underwater visualizations
- Visualizations that highlight terrain altitude
- Enhanced support for web services

TerraLens Visualizations

4K Resolution and DPI Scaling

Some GUI toolkits adjust to 4K and other high Dots per Inch (DPI) resolutions by scaling the display. For viewports that are not explicitly initialized with a GUI window, such as those using offscreen rendering or rendering to a texture, this automatic scaling can result in fuzzy text and screen symbols. Applications can now provide the DPI scale factor to TerraLens in the method:

```
Viewport::onSize (width, height, scaleFactor)
```

This will allow TerraLens to adjust the detail of its screen based primitives to provide crisper looking graphics. The DPI can generally be retrieved from the GUI toolkit API.

Globe Color for 2D

The method `Viewport2D::setGlobeColor(`*`color`*`)` defines a color for the 2D globe display, which is visible with Orthographic projections. Note that this is different from the viewport background.

Elevation Model Clarification

The interpretation of elevation values (vertical datum) is dependent on the geoid model in use; this applies to both values in Elevation Map Data and the altitude provided for geographic primitives. Using a mix of datums can result in a visual discrepancy when the 3D camera is close to the ground.

TerraLens 9.2 applies the following criteria with respect to elevation:

- Map data is always loaded as EGM96
- Primitive altitude is always relative to the user-specified geoid model set for the Viewport
- Queried elevation values are always relative to the user-specified geoid model set for the Viewport
- A new method provides conversion of a GeoPoint3D from a specified GeoidModel returning a GeoPoint3D using the Viewport's current GeoidModel

  `GeoPoint3D Viewport3D::getGeoPoint(`*`geoPoint3D`*`, `*`geoidModel`*`)`

Support for Underwater Visualization

The method `Viewport3D::setWaterVisibility(visibility)` controls the display of a water surface in the Viewport3D. The default RGBA color of [0,0,255,128] can be modified using `Viewport3D::setWaterColor(color)`. The elevation of the water surface, by default at 0 meters, can be modified with `Viewport3D::setWaterLevel(WorldDistance level)`.

Sunshading can now be applied to subsea elevations, where the elevation is below 0. The new attribute MinElevation must be set for the sunshaded map, or in the stylesheet, to indicate the lowest value to which sunshading will be applied; by default this is 0 meters.

Line of Sight and Ground Proximity Visualization

Support has been added to provide visualizations that support Line of Sight/Viewshed and Ground Proximity areas, using the loaded terrain data.

The new ViewShed class highlights an area on the terrain showing what is visible from a given 3D location. It highlights the area that is within line-of-sight of that viewpoint, and shadows the area that is obstructed by terrain. The colors used to show visible/not visible, the geographic location and Field-of-View of the viewpoint, and the range of the viewshed can all be defined via the API. The Field-of-View can be a narrow band, or a full 360-degree circle. Note that this is a visual approximation only, and accuracy is restricted to the resolution of the terrain data.

Ground Proximity visualization highlights an area primitive, coloring it based on the altitude of the terrain within that area. This could, for instance, be used as a visual warning system around an aircraft. Multiple instances of this presentation can be displayed concurrently, at different locations. Again, this a visual approximation only, and accuracy is restricted to the resolution of the elevation data attached to the viewport. The new primitive fill type `ElevationFill` takes in an `ElevationFillColorList` which is a set of elevation/color pairs in the same format as the Elevation Map. A similar visualization can be applied to a line primitive using the line style `ElevationLine`.

TerraLens Web Services

TL Server Enhancements

In 9.0, we introduced **WebServer** and **WebServices** so that a single server instance can support a greater selection of functionality. KML and GeoJSON are now provided via their own service (not via WMTS as before). Vector maps that are to be served out can be added using the classes WebVectorService and WebVectorLayer.

The full set of WebServices are:

- WMTSService
- WebVectorService
- WebTerrainService
- SymbolService

WebTerrainService produces terrain tiles for use with TerraLens Core and TerraLens Web3D.

SymbolService can serve the MilGraphics (Mil2525B, Mil2525C, etc) symbology as bitmaps.

MapSource and URL Enhancements

Support has been added to load a URL for a symbol primitive, including if it's used in a KML map source.

MapSource loading for WMS and WMTS has been extended to support KVP string encoding in addition to RESTful interfaces.

TerraLens Mapping

MapSource Management

Loading time has been reduced for Web MapSources (WMS, WMTS) and S-57 via the MapSource API.

A new method MapSource::getParent() will return the higher level container MapSource object for a MapSource as found during discovery, allowing the application to traverse the MapSource hierarchy.

Tracking Map and S-57 Presentation Loading

A new interface has been added allowing applications to test whether the viewport has loaded all the required map tiles. For instance, if the application wants to ensure all maps are done being displayed before continuing execution, the method Viewport::isMapLoadingComplete() can be polled until it returns True.

As this adds overhead to the TerraLens engine, the method Viewport::setMapLoadingTracked() can be used to turn this tracking on/off at runtime.

If the S-57 Presentation classes from TerraLens 7x are being used to display S-57 map data, similar functionality is available via S57Presentation::isLoadingComplete().

Support for MapBox Tile format

The MapSource class has been extended to support reading and displaying MapBox vector and raster maps. This support is added through an optional CustomSource plug-in provided with the TerraLens libraries; the desired map plug-ins must either be explicitly linked with the application or loaded at runtime using InterMAPhics::LoadPlugin(). Once the plug-ins are loaded, they can be accessed via MapSource methods just like any natively supported map.

Map Rasterizer

Vector maps that contain large numbers of features can be time consuming to load and render, even when pre-processed into TerraLens tile (sdt) format. This is especially true for high resolution data, and maps that apply complex stylesheets. If the ability to interactively pick individual map features is not required, an option to improve runtime performance is to rasterize the vector maps. The Rasterizer system introduces two new classes: Rasterizer and RasterizerLayer. As this conversion can be time consuming, these classes would typically be used in an offline application to combine and convert vector maps to a rasterized format. If this is of interest, please contact Kongsberg Geospatial Technical Support for more information.

> The **RasterizerLayer** is essentially a MapLayer. Maps can be added to the RasterizerLayer.

The **Rasterizer** class acts as the main interaction point. RasterizerLayers are added to the Rasterizer to provide it with data. For producing the rasterized images, the Rasterizer has two methods: render and renderToGeoTIFF.

Rasterizer::render takes in a GeoExtent and the information for a bitmap (ex. a buffer, width, height, pixel format). It will render out the map data contained in the GeoExtent into the provided buffer.

Rasterizer::renderToGeoTIFF takes in a GeoExtent, a name, a width, and a height. This method produces a GeoTIFF with the rasterized data for the provided GeoExtent. The GeoTIFF will be named using the provided name and its geo extent, and its size will be the provided width and height.

Performance Optimizations

Frame Rate

The following features have been analyzed and optimized to improve rendering speed, without compromising visual resolution or accuracy.

- Vector map displays
- S-57 Presentation classes
- Curved text labels on maps i.e. names that follow the line of a road. Duplicate text labels have been removed.
- Atmosphere in the 3D viewport

Platform Support

Android release using OpenGL Shaders

TerraLens 9.2 is available for Java on Android 7.0 or newer. OpenGL-ES2 is required.

TerraLens 9.1

TerraLens 9.1 continues the work from the previous release leveraging the Graphics Processing Unit (GPU) through the use of OpenGL shaders to achieve greater performance at runtime.

Key features of the 9.1 release include:

- Improved 3D terrain rendering performance and functionality, including support for multiple terrain sources
- Support for terrain output from the TerraLens Server
- Loading resources from a URL
- Display of KML and GeoJSON vector data
- LIDAR data visualization
- Performance optimization for visualization of vector maps

Terrain Enhancements

The TerraLens Terrain provides the elevation model of the earth surface, over which maps and ground-based primitives are draped. The management of terrain in a 3D Viewport is now similar to map layers in a viewport. Multiple Terrain objects, each associated with their own MapSource data, can be added to a 3D Viewport. The order in which the Terrain objects are added to the viewport determines their visual hierarchy, with the later additions visually on top of the earlier ones in areas where the data overlaps.

Terrain objects can be instantiated from a wide range of MapSource formats, including native elevation data (DTED, SRTM, DEM GeoTIFF), TerraLens extracted elevation data in tile format (files with the ".sdt" extension), or TerraLens extracted terrain tiles (files with the ".dtt" extension). Additionally, a Terrain can be associated with the proprietary terrain format served by the TerraLens Server product.

The maps draped over the terrain are no longer limited by the resolution of the terrain, and will be rendered up to the maximum detail level supported by the map data.

The runtime management of the underlying Terrain model has been reworked to provide better performance (i.e. frame rate) especially when zooming and panning the camera in the 3D Viewport.

Graphics Resource Styles loaded from a URL

Graphics resources used to define symbol, fill, or line resource styles can now be loaded from an HTTP or HTTPS address. The file will download asynchronously and will render when ready, so the application is not blocked. The completion of the download will not automatically cause a cached layer to redraw, maintaining the application's control over the redraw strategy. This capability integrates the FileSymbol object with the symbol service provided by the TerraLens Server for MIL-STD symbology.

KML and GeoJSON Vector Maps

The MapSource class has been extended to support reading and displaying KML and GeoJSON vector maps. This support is added through optional CustomSource plug-ins provided with the TerraLens libraries; the desired map plug-ins must either be explicitly linked with the application or loaded at runtime using InterMAPhics::LoadPlugin(). Once the plug-ins are loaded, they can be accessed via MapSource methods just like any natively supported map.

Point Cloud and LIDAR Data Visualization

The new WorldPointCloud class is a 3D world primitive which allows for the efficient display of a large number of particles. The particles are displayed with either an assigned RGB value or an intensity value mapped to a color ramp.

The WorldLidar class is a specialized WorldPointCloud which is instantiated with a LIDAR data file; both las and laz formats are supported through the use of the 3rd party `laslib` library. This library is packaged separately within the TerraLens product distribution and must be explicitly linked by applications using this class. Alternatively applications can process the LIDAR data independently, and use the WorldPointCloud class to display it. For this release, WorldLidar is available for C++.

Performance Optimizations for Visualization

Methods that are CPU intensive have been updated to leverage the GPU to improve runtime performance. This includes:

- Methods that modify visual effects of the Map class, such as color, alpha, saturation, intensity and contrast
- Application of text outlines

Updated 3D Model Support

TerraLens now supports loading models in GLTF2 format.

Extended support for Querying Elevation

Support for querying MapSource elevations has been extended to all elevation formats: DTED, SRTM, DEM GeoTIFF and Elevation SDT.

Text Following Lines on Maps

Previously, the entire text label on vector line maps rotated to follow the direction of the associated line. This has been updated so that the text label will follow the curve of the road, so each character is positioned independently. Within a StyleSheet, the Labels option “LabelHorizontal” can be set to change this default behavior.

TerraLens 9.0

TerraLens 9 delivers a unified rendering pipeline for your desktop, mobile and web geospatial applications. TerraLens 9 leverages the Graphics Processing Unit (GPU) through the use of OpenGL shaders to achieve greater performance at runtime. For browser based applications, WebGL is used.

The API remains the same, so switching to the new release should simply be a matter of recompiling with the new libraries.

Note: TerraLens 9 requires OpenGL 3.3 or newer. Older graphics cards are supported by TerraLens 8.

Key features of the 9.0 release include:

- Interoperability with web applications
- Enhanced mapping support via the common MapSource class
- Improved logging support for real-time systems

WMTS/WMS MapSource Improvements

Improvements have been made to the loading of WMTS and WMS map sources.

Improvements include:

- Users can now specify the TileMatrixSet to be used for retrieving map data by including the TileMatrixSet in the MapSource constructor.
- The display of images from WMTS sources has been improved. For example, text will display far more crisply.
- The loading of WMTS and WMS data conforms to OGC standards.

S63 Through MapSource

S63 can now be loaded through the MapSource API by providing the appropriate authentication information when loading the data set.

S102 Support

Support for the map format S102 has been added. This map data is a bathymetry/elevation data format, so it can be treated in the same way as a DTED, SRTM, or DEM GeoTIFF.

In addition, MapSource::queryDataRecords() can be called to retrieve the S102's elevation and uncertainty values.

Greater Contrast Elevation Style

The default elevation map style has been changed to provide greater contrast between elevations.

Terrain Engine Enhancements

The terrain engine has been enhanced to be more scalable to better support hi-resolution terrain including up to 50cm resolution data.

Raster Extraction Options

The extraction parameter MapFormatType has been expanded to include two additional types: **Dynamic Tiles** and **Smallest Tiles**. Dynamic Tiles will create tiles that try to match the native data's aspect ratio and resolution to improve display quality. These tiles can be larger or smaller than the standard 256x256 tile. Smallest Tiles is similar to Dynamic Tiles, but will only allow the tile's size to be less than or equal to a 256x256 tile.

Video Draping on Terrain

GeoVideo can now be used to project an image on to the terrain from an aerial perspective.

Enhanced Material Support

We have introduced two new Materials classes for 3D Meshes. The High Visibility material is designed to make 3D meshes stand out more in a wider range of conditions. The Physical Material gives more control over how the mesh should be rendered by using Physically Based Rendering (PBR) techniques.

Enhanced Logging Functionality

More logging has been introduced in the TerraLens kernel, to improve debugging experiences. Errors now include a source, a severity, and type. This allows the user to control what errors they would like to see. The user can control how the error messages should be displayed through the Log Output class.

Custom Projections

We have introduced the **CustomProjection** class for use in 2D viewports. The class allows applications to use projections or variations outside of the standard projections we provide.

TL Server Enhancements

In 8.3, creating a WMTS server was created by a MapServer class. In 9.0, we have introduced **WebServer** and **WebServices** so that a single server instance can support a greater selection of functionality.

The WebServices are:

- WMTSService
- TerrainService
- SymbolService

The **WMTSService** produces WMTS tiles, as well as providing JSON or KML map data if vector is included in the service's map data. The WMTS Service supports custom tile matrix sets and projections. The serving of WMTS data conforms to OGC standards.

TerrainService produces terrain tiles for use with TerraLens Web Core 3D.

SymbolService can serve the MilGraphics (Mil2525B, Mil2525C, etc) symbology as bitmaps.

TerraLens 8.3.0

The 8.3.0 release focuses on changes in these areas:

- Improved support for web services
- 3D features
- Issue resolution as listed in Section 5 “Problems Addressed”

DEM GeoTIFFs

Digital Elevation Model (DEM) GeoTIFFs are now supported. These maps operate like DTED map data. DEM GeoTIFFs can be styled at runtime with elevation styles.

Similar to DTED, DEM GeoTIFFs can be tiled into the tiled terrain format (DTT) for use as terrain.

WMTS MapServer

TerraLens can be used to create a WMTS map server. This is accomplished through the **MapServer** and **MapServerLayer** classes.

The **MapServerLayer** acts similarly to a **MapLayer**. The **MapServerLayer** contains **Map**, **MapGroups**, or **MapPresentations**. While all supported map formats are usable by the **MapServerLayer**, the best performance will be achieved with TerraLens tiled map data (SDT format). Each **MapSource** added to the layer should call `setWaitForBestResolution(true)` to ensure the produced tiles are the full detail level.

One or more MapServerLayers need to be added to a MapServer object for the MapServer to serve out map tiles.

WMTS MapSource

Display of maps from a Web Map Tile Service (WMTS) is supported via the **MapSource** class. The usage is the same as loading WMS through the MapSource API.

WMS MapSource

The WMS MapSource support has been enhanced. These limitations have been removed:

- *The WMS data coverage should be defined in the "bbox" parameter of the URL string in geo degree format.*
- *The screen pixel size of maximum resolution for the entire WMS data area should be defined in the "width" and "height" parameters of the URL string. A square area is expected. If the size not defined, wgm level 18 will be used as default.*
- *The single layer in the URL string should be used before it is set into a **MapSource** object.*

Integration of Custom Map Formats

The CustomSource class can be used to feed custom map data into the TerraLens mapping engine. This allows integration of map formats not supported natively by TerraLens. More documentation and examples will be available in the next release; please contact Kongsberg Geospatial Technical Support if you require more information.

Querying the Map Source to Retrieve Map Data Records

In Version 8.1, support was added to query a vector based map data source (VPF, DNC) in order to return records matching a user-specified set of criteria. This capability has been extended to ESRI shapefiles and S-57 map data.

Text Following Lines in Maps

By default, text labels on vector line maps will now rotate to follow the direction of the associated line, such as for road features. The labels will only display if there is enough room along a line segment to display the label. Within a StyleSheet, the Labels option "LabelHorizontal" can be set to true to return to the previous behavior.

3D Model Formats

The **WorldModel** and **WorldModelData** classes have been enhanced to support a large variety of 3D model formats. The list of supported file formats are: 3DS, BLEND (Blender), DAE/Collada, FBX, IFC-STEP, ASE, DXF, HMP, MD2, MD3, MD5, MDC, MDL, NFF, PLY, STL, X, OBJ, OpenGEX, SMD, LWO, LXO, LWS, TER, AC3D, MS3D, COB, Q3BSP, XGL, CSM, BVH, B3D, NDO, Ogre Binary, Ogre XML, Q3D, ASSBIN, glTF, and 3MF.

Field of View (FOV) Control

In 3D cameras, the X and Y Field of View (FOV) can be set separately. If only one value is set, then the other value is controlled by the viewport's aspect ratio. This feature can be used to more closely represent the FOV values of a physical camera.

TerraLens 8.2.1

The 8.2.1 release focuses primarily on changes in the following areas:

- Improved thread safety and memory cleanup when deleting viewports
- Issue resolution as listed in Section 5 “Problems Addressed”

New functionality is described below, with details available in the Reference Manual.

Map Format Support

TerraLens now supports Shuttle Radar Topography Mission (SRTM) and GeoTIFF Digital Elevation Map (DEM) data for elevation maps. As with DTED, these formats can be tiled for display with color ramps or sunshading in 2D, and further tiled to “dtt” format to provide a terrain model in 3D.

Raster support has been extended for CIB/CADRG VFR Sectional and VFR Sectional Inset raster maps.

TerraLens can process JPEG2000 files that provide geographic projection and coordinate system information in an external world file (.jgw), in addition to the previously supported embedded projection information.

Support is extended to ESRI maps using “proj4” projection strings, in addition to the previously supported UTM projection and geographic coordinate definition.

Support for .prj projection files has been enhanced for the image map formats. The "Projected Coordinate Systems" defined in the prj file will be used to search a collection of known projections. Latitude and longitude projections defined in the prj file will be handled by TIFF map files. Support is added for loading UTM projections through prj files for PNG, BMP, and JPG map files.

WMS Display

Display of maps from a Web Map Service (WMS) is supported via the MapSource class. For better performance, retrieved WMS images can be cached locally for reuse, as shown in this example:

```
// Create a MapSource from the fully defined URL string, including
// the coverage area.
MapSource* ms = new
MapSource("http://ows.terrestris.de/osm/service?REQUEST=GetMap&VERSION=1.1.1&
FORMAT=image/png&SRS=EPSG:4326&BBOX=-83.000000000000000,27.000000000000000,-
82.000000000000000,28.000000000000000&TRANSPARENT=TRUE&SERVICE=WMS&LAYERS=OS
M-WMS&STYLES=default");

// Set the cache location for storing the WMS images locally for reuse
MapSourceInfo info = ms->getSourceInformation();
info.createLocalCache("D:\\temp\\mycache\\");
ms->setSourceInformation(info);
```

For this release, WMS support requires the following:

- The full URL string should be valid and verified by user.
- The WMS data coverage should be defined in the "bbox" parameter of the URL string in geo degree format.

- The screen pixel size of maximum resolution for the entire WMS data area should be defined in the "width" and "height" parameters of the URL string. A square area is expected. If the size not defined, wgm level 18 will be used as default.
- The single layer in the URL string should be always used before it is set into a MapSource object.

Military Graphics

Three new symbols have been added to the NTDS military graphics library: missile, sonobuoy reference center, shore bombardment point.

Air corridors were added for the Mil2525 graphical engine. Air corridors are made up of control points and segments. A corridor is made up of two or more geographic points each representing a control point. Segments connect the control points.

Text blocks in the Mil2525 graphical engine can now be aligned in the center or along either X or Y axis.

Querying TerraLens License Availability

A static interface `InterMAPhics::checkBasicLicenseAvailability()` allows the application to query whether a runtime license is available. This interface is available through **object.h** .

Not having a license available will cause an application to exit immediately at the next call to the TerraLens API. Checking the license availability before initializing TerraLens allows the application to intercept this behaviour and provide appropriate feedback to the user.

By default, if the operating system allows, TerraLens will pop up a message box indicating failure to the user before the application is terminated.

Interpreting and Displaying Numerical Map Data Values

New enumerated types have been added to the InterMAPhics::Object class as "`UnitsOfMeasure`" and "`MeasurementTag`". These can be used to change how numerical values in the map data are interpreted and displayed. Currently, support is limited to Depth values in DNC and S-57 maps. By default, values are assumed to be in Meters.

These types can be used with **MapSourceInfo** to change how numerical values representing depth in the map data are interpreted.

```
// Change the base unit of a MapSource
MapSource* mapSource = new MapSource("C:\\Data\\Maps\\DNC\\dnc17");
MapSourceInfo info = mapSource->getSourceInformation();
info.setUnitsOfMeasure(Object::Unit_SI_Meters,Object::Measurement_Depth);
mapSource->setSourceInformation(info);
```

These types can also be used with the **MapStyleSheet** to change how numerical data values in the map are displayed.

```
// Set the display unit of measurement
MapStyleSheet sheet;
MapStyle mapStyle = MapStyle();
mapStyle.setUnitsOfMeasure(Object::Unit_Fathom,Object::Measurement_Depth);
sheet.add(mapStyle);
sheet.apply(*map);
```

GeoVideo Display

The **GeoVideo** class has been extended with a constructor and a `setCorners()` method allowing the application to provide 4 geographic corners in which to display the video frames, in addition to the existing 2-point extent display. The video is stretched, shrunk or rotated as needed to match the given polygon area.

TerraLens 8.2.0

The 8.2.0 release focuses primarily on changes in the following areas:

- rendering to an OpenGL texture
- improved memory management in Java and CLI
- StyleSheet performance and functionality
- 3D features and improved 3D rendering performance

New functionality is described below, with details available in the Reference Manual.

Mapping

StyleSheet Application

The `MapStyleSheet::apply()` method has been modified to take an optional `StyleSheet::ApplyMode`, which indicates whether the new style sheet is to be appended to the current set of style sheets, or replace the current set of style sheets (including any default styling). By default, the style sheet will be appended. To remove all styles, a new (empty) style sheet can be applied with the mode `ApplyMode_Replace`.

Control of S-57 and DNC features in StyleSheets

The default style sheet settings for S-57 and DNC maps can be overridden on an individual basis, including the symbol applied to a particular feature, the size of the symbol, and the default range levels.

The **StyleSheet** class has also been added to support S-52 color palettes. `S52Style::colorTableType()` allows ColorTableType values of **DAY**, **NIGHT**, or **DUSK**.

The current color table settings can be retrieved from `S52Style::exportS52ColorTables()`, and new color tables loaded with `S52Style::loadS52ColorTables()`.

DTED Multi-Level Support

In the OpenGL version only, the **MapSource** class will now support loading mixed detail levels of DTED data from the same folder, creating a hierarchy with a separate MapSource child for each level of DTED found. Note that this will slow down performance during the discovery of map data, compared to having separate folders for each DTED level in the original map data.

TerraLens 5x Map Archives

The CopyDirective enumeration used by **MapSet** to determine the outputted map tile format has been extended with a value of “copyIM5” to produce the TerraLens 5x sdl archive.

OpenGL Features

Visual effects are possible in the OpenGL version of TerraLens 2D/3D that are not available in the X11/GDI version. However, the API remains consistent between the X11/GDI and OpenGL versions of TerraLens; discrepancies between graphics drivers are identified for each affected method in the Class Reference Documentation.

OpenGL Render Strategies

TerraLens now provides an API for setting OpenGL specific options and selecting between multiple rendering methods, such as:

- **OpenGLOffscreenRenderer** – a windowless approach for rendering, typically used for the snapshot.
- **OpenGLRegionRenderer** – specifies an area within a GL context into which the TerraLens viewport will be rendered, allowing it to exist within the same window as other GL graphics but not blend with them.
- **OpenGLTextureRenderer**– specifies a GL texture into which the TerraLens viewport will be rendered. This texture is controlled by the application which can create composite images in the same texture.
- **OpenGLWindowRenderer** – specifies the OpenGL window/widget for drawing; TerraLens will control the contents of this window and no other graphics can be drawn to it.

Anti-aliasing and Anisotropic Filtering Control

The `OpenGLRenderer:: setFiltering()` and `OpenGLRenderer::setAntiAliasing()` methods can be used to select the desired filtering and anti-aliasing modes. If the requested mode is not supported by the graphics card, TerraLens will select the next available mode.

Image Primitive Performance

To improve rendering performance of the image primitives, it is possible to enable OpenGL pre-multiplication of the images color values, rather than manually calculating and setting the value of individual pixels. This is available via the method `setPremultiplyAlpha()` for Screen, World and Geo types of Image classes, as well as **GeoVideo**. This feature is only available for the OpenGL version.

3D Features

Atmosphere Rendering

An atmosphere visual effect can be enabled for 3D via `Viewport3D::setAtmosphereVisibility()` - by default, the atmosphere is **off**. The look of the atmosphere, including hue, saturation and brightness, can be controlled with `Viewport3D::setAtmosphereColorTransform()`.

GeoCurtain Visualization

The method `GeoCurtain3D::setEdgeLine()` has been modified from a simple on/off setting, to an enumeration specifying which edges are to be displayed (none, horizontal, vertical, or all).

The method `GeoCurtain::setColorRamp()` can be used to interpolate a set of colors across the length of the curtain. When a color ramp is applied, the fill and edge colors are multiplied against the ramp color to provide additional control over the final appearance.

Primitive Interpolation

The `set()` method for **GeoCurtain** and **GeoPolyline3D** now supports specifying the resolution used for interpolation of the data points along a great circle. The default resolution is 1 degree.

World Model

`WorldModel::setMaterial()` is used to set the model color, including its ambient, specular and emissive color characteristics.

The `getExtent()` method retrieves the minimum and maximum points of the model's bounding box.

BasicGroup3D

The **BasicGroup3D** class is a positionless container class, which allows primitives to be logically grouped together even though they do not share a common position. It can contain both 2D and 3D primitives.

TerraLens 8.1.0

MapSource and Stylesheet Support for GDI and X11 Graphics Version

The new map handling process introduced in TerraLens 8.0 for OpenGL has been extended to the GDI and X11 graphics driver versions.

This simplifies the application code required to load maps at runtime, in that TerraLens automatically creates the presentation hierarchy required for display of all maps found in a MapSource archive. Additionally, style sheets can be used to define the visual attributes of the maps, along with rules to control the runtime display including range-based filtering.

The full feature description can be found later in this section under TerraLens 8.0, "New Map Handling Process in OpenGL version", and in Chapter 4: "Mapping in TerraLens 8".

Retrieving Map Source Information

The **MapSourceInfo** class maintains metadata available from the native map source data, including the map data type, its geographic extent, the resolution and scale, and textual description, if available. The **MapSourceInfo** can be retrieved using `MapSource::getSourceInformation()`.

Querying the Map Source to Retrieve Map Data Records

Support has been added to query a map data source in order to return records matching a user-specified set of criteria. Currently, the functionality can be performed on geographic vector based data such as VPF or DNC, and is limited to the leaf **MapSource** object; that is, one that has no additional children.

The method `MapSource::queryDataRecords( <extraction_criteria> )` returns a list of **MapDataRecords** matching the provided criteria such as name, data value, etc. The criteria are the same as used to extract a subset of map data for display/tiling.

From the **MapDataRecord**, information can be retrieved about the feature or attribute, as well as the TerraLens Primitive data used to visualize the record. This includes the geographic data points that make up the primitive. An application can use these data points for its own analysis or calculations, or to provide an alternate visualization for an object. For example, a query could look for all polygons within a particular geographic extent, or for all "Buoy" objects.

Compositing TerraLens with other Graphics

A Viewport constructor has been added which takes a GL context and GL texture, into which TerraLens will render. This allows TerraLens graphics to be composited with other graphics in an application controlled GL component, for instance when integrating TerraLens into a QML framework or **QtOpenGLWidget**.

The existing `Viewport::snapShot()` method can also be used to retrieve an image of the TerraLens viewport which can then be composited with other application graphics.

Note: This viewport constructor has been superseded by use of the **OpenGLTextureRenderer** in TerraLens 8.2.

Converting Primitives to Map Sources

The PrimitiveSource class converts geographic primitives into maps. Once the data is available in the Primitive MapSource, it can be extracted into TerraLens tile (sdt) format, styled using **MapStyleSheets**, and otherwise treated as a map. This allows user-defined geographic primitives to be handled within the efficient map tiling system used to display all map data formats.

This class can be used by the application to support map vector data in formats not handled natively by TerraLens. For instance, parsed KML primitives can be loaded into a **PrimitiveSource** to be treated as map data rather than individual presentations, and user-defined annotations can become a map within the **MapPresentation** hierarchy.

Support for Raster Maps without a Table of Contents file

TerraLens can now process and display maps in Raster Product Format (RPF) that are provided without the accompanying Table of Contents file, typically named a.toc. This includes maps in ADRG, CADRG, and CIB formats.

If no table of contents file is found, each first-level subdirectory under the "rpf" folder will be treated as a separate RPF map. Please note that different types of RPF maps (i.e. CIB and CADRG) should not be mixed in the same first-level directory, but can be located in separate directories under the same "rpf" folder.

Support for GeoTiff, TIFF and ESRI Maps with External Projection Information

TerraLens can process GeoTiff and TIFF files that provide geographic projection and coordinate system information in external world file (.tfw) and projection (.prj) files, in addition to the previously supported .tif embedded projection information.

Non-geographic TIFF files can be displayed as a geo-rectified map if the geographic projection is defined as a "proj4" string in the external projection (.prj) files.

Similarly, there is support for ESRI shape files that provide geographic projection and coordinate system information in external projection (.prj) files.

Support for Planar Multi-spectral TIFF

TerraLens supports 16-bit planar strip-format TIFF files up to 4 bands. Support is completely automatic through the Image class; however the user can use the new ColorTransform style class to scale the individual RGB channels to adjust the image. Since the bands actually cover a range of spectrums, it is often necessary to reduce the brightness of some channels (typically green, but it varies based on the satellite).

The **ColorTransform** class provides a method to scale each color:

```
ColorTransform::setScaleRed/Green/Blue(float value)
```

The ColorTransform style is then added to a **MapStyleSheet**, with the scaling applied to the TIFF image map via `MapStyleSheet::apply()`.

Map Layer Cache Mode for Display

The **MapLayer** constructor now takes a cache mode (specified by the enumeration `MapLayer::CacheMode)` to indicate the type of caching that should be used when displaying the data. A "cache" is the bitmap stored in memory resulting from rendering map data. Caching reduces the amount of data that is rendered when the visualization of the maps in the layer has not changed; that is, the attributes of the map and the map data to be displayed based on the camera zoom/extent area are the same as from the last update. Reducing the amount of rendering improves application performance; however each cache requires additional memory.

By default, a **MapLayer** is cached at the layer level; the cache produced in rendering the layer is maintained and can be simply copied when updating the parent viewport, rather than requiring a re-render of the layer's map data. In general, this default behavior is sufficient to balance performance and memory. This mode is set using `MapLayer::CacheMode_LayerCache`.

Alternatively, caching can be specified at the tile level. Each map layer is composed of multiple tiles which are built up from the map data covering a geographic area. When the data in the layer changes i.e. when the map is panned or zoomed, the layer has to be re-rendered. With caching at a tile level, only those tiles that have been updated or that are new have to be rendered; previously displayed tiles can be reused from their cache. This can improve performance when high-resolution vector data is displayed, but it does require additional memory for the tile cache. This mode is set using `MapLayer::CacheMode_TileCache`.

The cache mode can also be turned off for a map layer, in cases where the map is constantly moving or updating and there is no benefit to be gained from maintaining a cached image. In this mode, all map data records in the view are always rendered when the view is redrawn.

Map Source Merging

The MapSource class has been extended allowing native map data sources to be merged, for instance to combine multiple small GeoTIFF images to a single GeoTIFF map. This is enabled via an optional flag on the **MapSource** constructor:

```
MapSource(const char * path, bool mergeMapSource = false)
```

Please note that this will only support merging when loading native data sources of the same map format located in the same folder.

Map Source Conversion: Extraction Type and Tile Size

Support has been added for an application that wishes to convert maps read in by TerraLens to a format suitable for consumption in another product. At this time, support is limited to conversion to GeoTIFF.

A new method has been added allowing the application to set the width of the tiles created when extracting maps, namely:

```
static void MapSource::setTileExtractionWidth( int tilePixelWidth );
```

As well, two new extraction options have been added to the **MapFormatType**: GeoTIFF8bit and GeoTIFFTrueColor, allowing a map read in by TerraLens to be output to these GeoTIFF formats.

Radial Grids

Radial Grids provide a high performing and memory efficient way to display a gridded color image, for instance, output from a Radar Scan Converter or Sonar data. The color of the radial grid cells can be individually colored, or blended.

Please note that the Radial Grids are currently available in Java only. C++ and CLI support will be added in a future release.

3D Performance

Optional Geoid Model

To improve performance, the 3D Geoid model is disabled by default. It can be enabled by calling `Viewport3D::setGeoidModel()`, and passing in a **GeoidModel** - either `EGM96()` to enable, or `GeoidModel()` to disable it.

In general, the visual effect of the Geoid model is imperceptible unless you need values that have already been geoid adjusted to geo register when zoomed in at high resolution.

It affects the elevation value when performing conversion of 3D points.

Terrain Detail Level

Performance in 3D is greatly impacted by the amount of detailed terrain data to be visualized. The level of detail available in the source map data may be more than is required for a particular application, or under specific circumstances. The detail level used to render terrain can be controlled at runtime using the method `Viewport3D::setTerrainDetail()` which takes a world distance indicating the maximum desired detail level. For instance, `setTerrainDetail( Meter(100) )` would not show any terrain detail smaller than 100 meters.

This method can be used to control how an application responds when its resources are being stretched. For instance, if the time required for a 3D viewport update is too long, the terrain detail level can be lowered to maintain performance with a controlled degradation to the visual accuracy.

Version Compatibility

World Offsets in 3D

Any offset specified in world coordinates within a 3D viewport now represents a real world coordinate. Great circle computations are used to calculate the corresponding latitude/longitude position, and the altitude is calculated perpendicular to the plane of tangency at this newly calculated position.

World coordinates should be used as relative positions to a geographic position (the geographic parent's position).

Known Limitation in TerraLens 8.1.0

GDI/X11 Applications with 3D Views should upgrade to OpenGL

Customers who require a 3D view should use the OpenGL version of TerraLens, rather than the legacy GDI/X11 version. The 3D support in the fully OpenGL version has been optimized to leverage the driver capabilities. Using the 3D view in the GDI/X11 version of TerraLens is not fully supported.

TerraLens 8.0

New Map Handling Process in OpenGL version

An extensive description of the changes to TerraLens map handling is provided in Chapter 4: "Mapping in TerraLens 8". Please note that the new map handling functionality is currently only available in the OpenGL version of TerraLens; it will be added to the GDI/X11 version in an upcoming release.

Simplified Runtime Use

A new TerraLens API is available to simplify loading maps at runtime. This API connects a map presentation, or map layer, to a generic map source via the constructor:

```
public MapPresentation( mapSource ) or public MapLayer( mapSource )

where mapSource = new MapSource( mapSourceLocation );
```

The map source represents the data for an individual map, or alternatively an archive containing data for a number of maps. A containment hierarchy will be automatically created under the map presentation or map layer to accommodate the number of maps found at the archive source location.

The map source location can be a folder containing map data in a mixture of formats. Each individual map source object is either in a native map format (such as VPF, CADRG, MrSID, S-57 etc.), an TerraLens pre-processed tile format (i.e. the "sdt" files), or another source of map data such as a URL for a Web Mapping Services (WMS) feed. TerraLens determines the appropriate way of displaying the different map formats with no specialized application code.

Map Style Sheets

To further generalize the application code, the visual characteristics of the maps are specified outside of the application using a style sheet. The style sheet is used to set typical graphics attributes such as color, line style and width, fill pattern, font type and size. The style sheet also defines custom rules to control the runtime display of maps, such as range based filtering. A style sheet can be applied to an individual map, or to a set of maps. Different style sheets can be applied at runtime to change the look of the maps, for instance to support the concept of "Day" and "Night" palettes.

TerraLens provides default styling for all map formats. For S-57 and DNC, the style uses the standard S-52 visualization.

The TerraLens style sheet can be built up in memory using an API to set the attributes and rules; please refer to the Class Reference documentation for the classes under "_Style", including Expression, Label, **MapStyleSheet**. The style sheet can also be saved to and loaded from a file. The style sheet is saved in JSON format.

The **MapSource** will automatically apply a user specified style sheet named “mapStyleSheet.json” for a map, if located in the same folder as the root of the map data.

New MapPresentation Class

As mentioned above, a new **MapPresentation** container class has been added to the map presentation hierarchy, providing consistency with the `InterMAPhics::Presentation` containment structure. A **MapLayer** object may contain one or more **MapPresentation** objects. A **MapPresentation** object may contain **MapGroup** and **Map** objects.

When a **MapSource** object is associated with a **MapPresentation** object using the **MapPresentation** constructor, an appropriate map presentation hierarchy will be generated to emulate the directory structure found. For each folder that is an identifiable map data source, such as DTED or a VPF feature, a **Map** object will be created. For each folder that contains map data sources but is not itself the root of the map data source, a **MapGroup** will be created; this may be used for logical organization within the map archive. Please note that this convenience function assumes that the created map presentation hierarchy will not be modified by the application – if the organization of the **MapSource** changes, a new **MapPresentation** object should be created for it.

Alternatively, the application can explicitly manage the **MapPresentation**, adding **MapGroups** and **Maps** as needed.

Extended MapGroup Class

The **MapGroup** class now can contain other **MapGroup** objects, in addition to **Map** objects.

Extended MapLayer Class

A **MapLayer** object can now contain other **MapLayer** objects in addition to **MapPresentation** objects.

When a **MapSource** object is associated with a **MapLayer** object using the **MapLayer** constructor, an appropriate map presentation hierarchy will be generated to emulate the directory structure found. For each folder containing an identifiable map data source, such as DTED or a VPF feature, a **MapPresentation** object will be created. For each folder that contains map data sources but is not itself the root of the map data source, a **MapLayer** will be created; this may be used for logical organization within the map archive. Please note that this convenience function assumes that the created map presentation hierarchy will not be modified by the application – if the organization of the **MapSource** changes, a new **MapLayer** object should be created for it.

Managing Map Data

Map data may be managed in one of four ways; loaded from a given location, copied to a new location, extracted to the TerraLens tile format, and stored as a link to some other location.

The **MapSource** constructor requires a path or link to the location of the map data; the copy, extract, and link methods can then be used to manipulate the map data.

Access to Map Feature Data

Under TerraLens 8, feature data associated with individual map sources is maintained whether displayed natively or pre-processed. For instance, picking on a vector map can now return not just a pick of the whole map presentation (i.e. “River”), but the ability to retrieve the individual map record

(i.e. “Nile”) using `Viewport::multiPickMapRecords()` or `MapLayer::multiPickMapRecords()`.

The feature data can also be used to define rules for styling features using style sheets.

Improved Performance when Adding and Removing Presentations

The time required for adding and removing presentations from the viewport has been reduced, resulting in improved runtime performance.

Determining the TerraLens Release at Runtime

A static function `getInterMAPhicsInformation()` accepts a query string to return information about the current library including “Product”, “Copyright”, “Release” and “BuildDate”. Other values identify the Kongsberg Geospatial configuration such as “CommitID”, “CommitDate”, “JavaVersion”, “C_Compiler”, “CPP_Compiler”.

The static function `getKernelType()` returns an enumerated type indicating the type of underlying graphics supported by the TerraLens library. The possible values are “KernelX11” for Linux/Unix, “KernelGDI” for Windows, and “KernelOpenGL” for all.

GeoTiff projection support

If projection information is available in the GeoTiff data (i.e. PROJ.4 values) it will be used rather than converting the points to UTM.

BigTIFF support

TerraLens now supports BigTIFF map format.

Retrieving Map Resolution of Tiled Source

The resolution of a map (meters/pixel) can be retrieved from a map using `TiledMapSource::getResolution()`.

The functionality of the method has been extended to return either the native source resolution that existed in the original map data, or the resolution resulting from selecting a tiling detail level (low, medium, etc.). For a pre-processed map (i.e. one that has already been tiled) these two values will be the same.

RPF Map Data Boundaries

The geographic extents of the individual map data areas contained in a RPF map (i.e. CIB, CADRG) can be retrieved using the new method `RPFMapSource::getBoundaries()`.

Setting Altitude Relative to the Terrain in 3D

The new method `setTerrainRelative()` in the classes GeoPresentation3D and GeoGroup3D determines whether the altitude set via the position property is relative to the altitude and orientation of the terrain, or whether it is at sea level (default behavior).

Controlling Map Loading

The new method `MapSource::setDefaultDataLoadingMode()` allows the application to control the strategy used to load map tiles at runtime. This flag can be applied individually to each **MapSource**, allowing different strategies to be used for different map types or map instances.

The default option (**DefaultLoadingMode**) is the historic method of loading tiles. It will load the levels starting at the lowest resolution up to the current view range. This is typically the desired option, because while it requires more processing time initially, runtime performance will be improved when changing the view range.

The new option (**DirectLoad**) will load only the tiles required for the current view range; tiles from lower resolution view ranges will not be loaded. This minimizes the time required to display a map initially, however, there could be instances when a map is not displayed immediately after zooming to a new range as new tiles are loaded. This option is preferred when large amounts of high-resolution data are being displayed and they will be displayed only at a low view range (i.e. you have no need for them to be displayed at higher view ranges).

Retrieving a ScreenGroupSymbol as a Bitmap

The method `ScreenGroupSymbol::getBitmap()` returns a bitmap containing the screen group symbol, including for **MIL-STD-2525** symbol. This bitmap can be used on buttons, toolbars, etc.

MIL-STD-2525C Symbology Update

The MIL-STD-2525 API has been extended to provide greater control over Tactical Graphics. This includes the following:

New Mil2525ScreenGroup

A new class Mil2525ScreenGroup which inherits from ScreenGroup has been added to simplify control of text and direction of travel indicator features decorating static Mil2525 symbols.

Accessed via `Mil2525SIDCCode.getMil2525ScreenGroup()`.

Extended symbology support

Support has been added for displaying the following status of MIL-STD symbols: Capable, damaged, destroyed and full capacity, and alternate symbols. The alternate status is specified using `Mil2525SIDCCode.setUseAlternateStatusSymbols()`

Symbol Pickability

A hollow fill is now used for unfilled frames and icons in the MIL-STD symbology, allowing mouse events on this area to result in a “pick”.

Color support

Mil-STD symbol color scheme support has been added to support ‘light’, ‘medium’, ‘dark’, ‘dimmed’ and custom color schemes for use when generating symbology.

`Mil2525SIDCCode.setColorScheme()` sets one of the default or custom schemes.

`Mil2525SIDCCode.setColorTable()` allows applications to change existing or add new color schemes.

Line pattern support for tactical graphics

Tactical graphics with complex line patterns have been converted to use textures and bitmaps to improve visual appearance.

The use of a line pattern increases rendering speed and improves the visual look of patterns

Text updates

The **Mil2525ScreenGroup** and **TacticalGraphic** classes have extended `setText()` allowing the font and size to be overridden instead of using default settings.

The default font can be defined with `Mil2525SIDCCode.setDefaultFont()`.

Scaling of a font can be defined with `Mil2525SIDCCode.setFontScaling()`.

To improve legibility, a number of symbols support displaying text components as a font rather than a bitmap by default. This functionality can be disabled using `Mil2525SIDCCode.setUseBitmappedStrings()`.

Tactical Graphics Updates

Tactical graphics symbols are defined as screen elements with a geo-anchor using `Mil2525SIDCCode.setGeoBasedTacticalSymbols()`.

New **SymbolGraphic** class provides custom control for tactical graphic symbols

Range tactical graphics updated to provide more presentation control

New Compound tactical graphic concept e.g. G*M*NR is a compound graphic with G*M*NZ as the child graphic. Allows children of a compound graphic to be set and manipulated separately.

OpenGL Enhancements

Visual effects are possible in the OpenGL version of TerraLens 2D/3D that are not available in the X11/GDI version. However, the API remains consistent between the X11/GDI and OpenGL versions of TerraLens; discrepancies between graphics drivers are identified for each affected method in the Class Reference Documentation.

GeoString Class

The new **GeoString** class defines text that is displayed as geographically projected on the display. Its position and direction can be defined based on origin, bearing and axis.

Note that currently the font system requires font sizes to be specified explicitly. Selecting a small font size for a **GeoString** can result in pixilated output at high zoom, while large font sizes can increase memory use significantly. Users should select the smallest acceptable font size for their desired zoom range.

This class is supported only on the OpenGL version; in X11/GDI currently nothing will be displayed for a **GeoString**.

OpenGL Texture Compression

The new method `Map::setRasterCompression()` allows users to enable compression (if available in the graphics card), reducing texture memory consumption by 75% at the cost of some image quality. Texture memory usage in TerraLens is largely determined by the size of the Map Set.

Direct Color and Dynamic Color

The OpenGL version of the **Color** class no longer creates internal table entries tracking each color created. This color value will be used directly in the OpenGL rendering and will not be managed within TerraLens, thus reducing memory usage and increasing performance.

To allow setting basic colors without a color object the primitive and area objects have new color methods `setColor(color32bit, format)/setFillColor(color32bit, format)` to support setting color directly via an unsigned integer color value and specifying the format. There is a complementary call to retrieve the current `color: getColor(format)`.

Conversely, the **DynamicColor** class will be managed within TerraLens. Changing the value of a **DynamicColor** will change all objects that use that **DynamicColor** automatically, without having to modify each primitive individually.

The behavior of colors in the GDI and X11 versions of TerraLens remains unchanged; table entries are created for colors and use of the new methods is equivalent to using the `InterMAPhics::Color` class.

Line Drawing Updates

TerraLens now supports join and cap styles when drawing primitives in the OpenGL version, which results in visibly smoother lines and edges for primitives using a line width greater than 3. Note that this feature requires more resources to draw and could negatively impact performance depending on the number of primitives. The default line style uses no cap or join style.

The **MIL-STD-2525** symbols have been updated to use the cap and join support by default.

Threading Control over Map Updates

The method previously known as `setEnhancedThreads()` has been deprecated and its default set to false. The new API that should be used is `setAutoMapUpdate()`. If set to *true*, TerraLens will automatically call an update after loading each map request list. A map request list is generated when the camera's position is modified and new map tiles must be loaded for the new view range; the update is required to visualize the new map tiles. A separate request list is generated for each map, so an update would be performed as the loading of tiles is completed in each thread potentially leading to "flashing" of the viewport. In applications that regularly update the viewport i.e. based on a timer or a "heartbeat", the automatic map update is not recommended as it will lead to redundant updates.

4 Mapping in TerraLens 8

Overview of TerraLens 8 Mapping Approach

TerraLens 8 presents a new approach to map handling which simplifies the application code required to handle different map data sources.

Common API for all Map Data Formats

In previous versions of TerraLens, applications required specialized code to handle the various map formats before they could be displayed. Displaying a map from its native format (e.g. MrSID, CADRG) required different classes than displaying a pre-processed map (i.e. one cached to the TerraLens "sdt" tile format). S-57 maps utilized an entirely different set of presentation classes. WMS integration required specialized application code as well.

With TerraLens 8, the mapping engine determines the appropriate approach to interpret and display the map data, whether it is in a supported native format, TerraLens pre-processed format, or a WMS location. The API is simplified to associate a model (**MapSource**) with a presentation (**MapPresentation**), regardless of the data format of the model.

Simple API to specify Map Source

In previous versions of TerraLens, in order to access map source data the correct root folder level had to be specified; this could be confusing for users not familiar with the structure of map data in its native format. Each displayed map presentation had to be initialized with a unique map source.

With TerraLens 8, the mapping engine traverses the specified map source location to find all the individual maps that can be displayed. It also creates a map presentation containment hierarchy emulating the structure found at the map location. An application can therefore specify a single map archive location which can contain any mixture of native maps (e.g. VPF, Raster, Imagery, S-57) and pre-processed maps (i.e. TerraLens "sdt") of various data formats to visualize or manipulate as required.

External Specification of Map Presentation Characteristics

The visual appearance of the Map Presentation can be defined through a number of attributes, which differ depending on the type of map. For instance, a Vector Map (e.g. VPF) line feature, such as roads, will be displayed with a line style, width, color, whereas a Sunshaded Elevation Map will be displayed with a set of color ramp values.

In previous versions of TerraLens, the application had specialized code to deal with the visual attributes of each type of map.

With TerraLens 8 the visual attributes of a map are stored and manipulated using the new style sheet class MapStyleSheet. Applications are able to create new style sheets to modify visual behavior at runtime or edit style sheets saved with maps (in the JSON format) to modify how maps are loaded from map archives. Styles sheets contain all of the elements to control the visualization of a single map or a set of maps including newer features like text formatting and text outlines.

TerraLens 8 will automatically apply a user specified style sheet named "mapStyleSheet.json" for a map, if the style sheet located in the same folder as the root of the map data.

Sample use of Mapping Classes in TerraLens 8

Displaying a Map

The new TerraLens API simply connects a map display object (**MapPresentation** or **MapLayer**) to a generic map source. A map presentation containment hierarchy will be created under the display object to accommodate the number of maps and directories found. The application can determine how to logically structure the map presentation hierarchy i.e. in order to group maps together by usage, visual control, priority etc.

Map Source Example

The Map Source can be the data for an individual map, a folder containing multiple maps, or a WMS location. A customer could have maps organized by usage, as in the following picture:

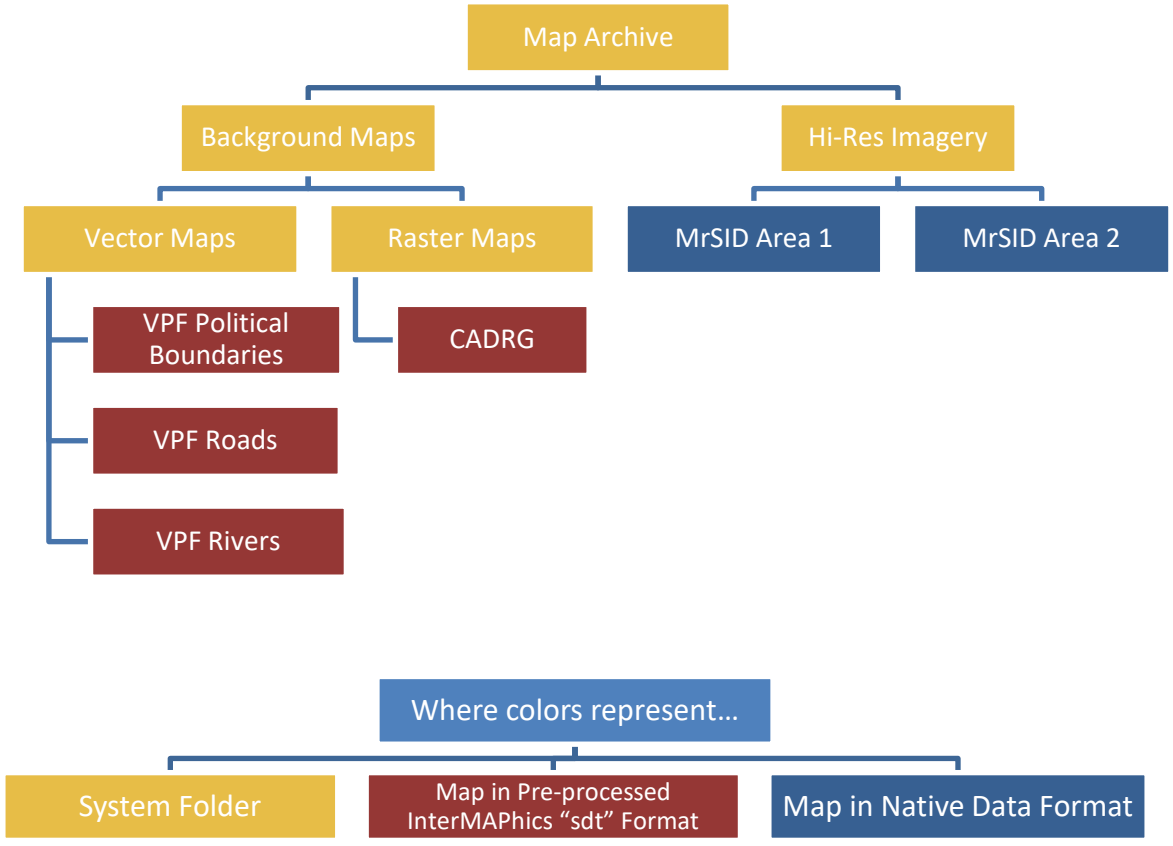

Figure 1: Displaying Maps Contained in an Archive

Map Display Example 1: Associating the map source with a MapPresentation object

```
// Map Source can be the data for an individual map,
// a folder containing multiple maps, or a WMS location.
// In this case 'mapSourceLocation' is a string defining the location of
// the directory 'Map Archive' in the previous picture.

MapSource* mapSource = new MapSource( mapSourceLocation );

// The containment hierarchy created under the MapPresentation
// will consist of a MapGroup object for each non-map folder found,
// and a Map object for each recognized map found.
//
// Referring to the diagram, objects will be created as following:
//    MapPresentation for: "Map Archive"
//          MapGroup for: "Background Maps", "Hi-Res Imagery"
//          MapGroup for: "Vector Maps", "Raster Maps"
//          Map for: "VPF Political Boundaries", "VPF Roads", "VPF Rivers"
//          Map for: "CADRG"
//          Map for: "MrSID Area 1", "MrSID Area 2"
//
// Note that no TerraLens object will be created for a folder that does
// notcontain map source data at some sublevel.

MapPresentation* mapPresentation = new MapPresentation( mapSource );
MapLayer* mapLayer = new MapLayer;
mapLayer->add(  *mapPresentation );
```

Map Display Example 2: Associating the map source with a MapLayer object

```
// Map Source can be the data for an individual map,
// a folder containing multiple maps, or a WMS location.
// In this case 'mapSourceLocation' is a string defining the location of
// the directory 'Map Archive' in the previous picture.

MapSource* mapSource = new MapSource( mapSourceLocation );

// The containment hierarchy created under the MapLayer
// will consist of a MapLayer for each non-map folder found, and a
// MapPresentation object for each folder containing maps.
//
// Referring to the diagram, objects will be created as following:
//    MapLayer for: "Map Archive",
//          MapLayer for: "Background Maps"
//          MapPresentation for: "Hi-Res Imagery"
//          MapPresentation for: "Vector Maps", "Raster Maps"
//          Map for: "VPF Political Boundaries", "VPF Roads", "VPF Rivers"
//          Map for: "CADRG",
//          Map for: "MrSID Area 1", "MrSID Area 2"
//
// Note that no TerraLens object will be created for a folder that does not
// contain map source data at some sublevel.

MapLayer* mapLayer = new MapLayer( mapSource );
```

Defining the Look of the Map with Style Sheets

Style sheets defined using the class **MapStyleSheet** fully support defining the visual presentation of a map when associated with a map at runtime. This includes typical attributes such as color, line style and width, fill style, text font and size, as well as more specialized control such as the text outline effect ("halo") and formatting of map text strings. Visibility of a map can also be controlled via style sheets (described in the next section Using Expressions in Style Sheets).

Using style sheets the runtime application is able to present any supported map visually even if the type of map was not known at development time – no specialized code needs to be written for each new map type. Allowing map visual attributes to be defined and stored outside of the runtime API allows the user to create a certain look for a map that can be encapsulated with the map data and distributed to deployed applications.

Default style sheets are used by TerraLens to provide standard visualizations, for instance the S-52 standard is applied to S-57 and DNC map formats. Users can override the defaults if desired, by applying a style sheet on top of the default.

Style sheets are applied in a cascading manner – that is, the new style will not undo those styles applied previously. For instance:

- Style A is applied, which sets vector maps "Road" feature to a black dashed line.
- Style B is then applied, which sets vector maps "Road" feature to yellow.
- The "Road" feature will now be shown as a yellow dashed line, keeping the dash style from previously applied Style A.

Runtime creation of new style sheets is supported via the **StyleSheet** and **Style** classes. If an application using TerraLens 8 allows a user to modify the attributes at runtime, it should be through the definition and application of a new style sheet. Style sheets can be saved to and loaded from a file, specified in JSON format.

Style sheets provide a convenient way of supporting multiple user configurations in an application. An example would be "Day" and "Night" color palettes. Or, a standard map configuration may be provided to all operators, but certain user roles may have a different map configuration (i.e. turning on a different set of features), and the application further allows each user to have an individual map configuration (i.e. only allowing color to be modified).

Using Expressions in Style Sheets

Expressions are used to refine the set of map data to which the style or rule is applied. They can include both map data values such as feature name or city population, as well as runtime viewport settings such as range. An expression can be used to implement range based filtering of maps or map features, in which the style to turn visibility on is applied only when the view range criteria is met. Alternatively, an expression can be used to change the look of the individual map feature records depending on their data so that a city with a large population is shown with a bigger icon than a city with a small population.

The expressions are very much like C expressions.

- You use '`&&`' instead of '`AND`', '`||`' instead of '`OR`', and almost all C operators.
- For string comparisons, '`^`' is 'begins with', '`$`' is 'ends with' and '`~`' is 'contains'.
- String comparison is always case insensitive.
- Quoting uses single quotes.
- String values should always be quotes, but attribute identifiers should never be.

The following example shows how to change the symbol style for a points vector map when the camera range is between 350-370 NM.

```
// Define the visual attributes you want to apply for the
// vector points in the map.
MapStyle mySymbolStyle;
mySymbolStyle.setSymbolColor(Color(255,0,0,255));
mySymbolStyle.setSymbolSize(12);
mySymbolStyle.setSymbolStyle(BasicSymbol(BasicSymbol::BasicSymbolDot));

      // Set the range and attribute filter.  We want to apply this style to
      // the map
      // titled "Island", when the range is between 350-370NM

Using Expression;
      mySymbolStyle.setFilter(
      both( compare(
            "Island", Exact, id(Title)),
            both( compare(id(Range), LessThanEqual, NauticalMile(370)),
                  compare(id(Range), GreaterThan, NauticalMile(350)))));

      // Add the new style to the map style sheet.
MapStyleSheet myStyleSheet;
myStyleSheet.add( mySymbolStyle );
```

Map Preparation for Runtime Display

The **MapSource** class has been extended to support the creation or loading of map archives, allowing grouping of maps that are to be visualized by an application at runtime. This archive can contain a mix of pre-processed maps, native maps, or links to map data.

To manage the map source data, you can copy (maintain the current data format), extract (process into TerraLens tile format), or link (create a link to a map source).

To refine the data extracted from the map source, a Filter expression can be defined prior to the extraction, as in the following example:

```
// Point the map source at the source data.
MapSource mapSource( nativeMapLocation );

// Indicate which features you want to pull out into a separate map.
// In this case, it is using the feature code from the VPF format.
```

```
mapSource.setFilter(Expression::fromString("F_CODE='BA040'"));

// The features matching the expression will be processed into TerraLens
// tile format and tiled to the specified destination.

mapSource.extract("/MyMaps/");
```

These filter expressions can also be used in style sheets and at runtime to set visibility or visual attributes on a per feature basis. Therefore, unlike previous versions of TerraLens, it is not necessary to extract each feature into its own map to control per feature display at runtime.

Control of Runtime Map Display

The new method `MapSource::setDefaultDataLoadingMode()` allows the application to control the strategy used to load map tiles at runtime.

The default option (DefaultLoadingMode) is the historic method of loading tiles. It will load the levels starting at the lowest resolution up to the current view range. This is typically the desired option, because while it requires more processing time initially, the runtime performance will be improved when changing the range of the map.

The new option (DirectLoad) will load only the tiles required for the current view range; tiles from lower resolution view ranges will not be loaded. This minimizes the time required to display a map initially, however, there could be instances when a map is not displayed immediately after zooming to a new range as new tiles are loaded. This option is preferred when large amounts of high-resolution data are being displayed and they will be displayed only at a low view range (i.e. you have no need for them to be displayed at higher view ranges).

This flag can be applied individually to each **MapSource**, allowing different strategies to be used for different map types or map instances.

Accessing Individual Map Features and Feature Data

Under TerraLens 8, associated data available for individual map objects is maintained whether displayed natively or pre-processed. For instance, picking on a vector map can now return not just a pick of the whole map presentation (i.e. the VPF map “River”), but the ability to retrieve the individual map record (i.e. “Nile”).

The set of picked map records is retrieved using `Viewport::multiPickMapRecords()`. From a **Record**, you can access associated map source data by specifying the **Field** to be retrieved i.e. specifying the name or index used in the map source data. The available field names can be retrieved from the **MapSource** object.

The association between the map presentation hierarchy and the map data objects is shown in the following figure:

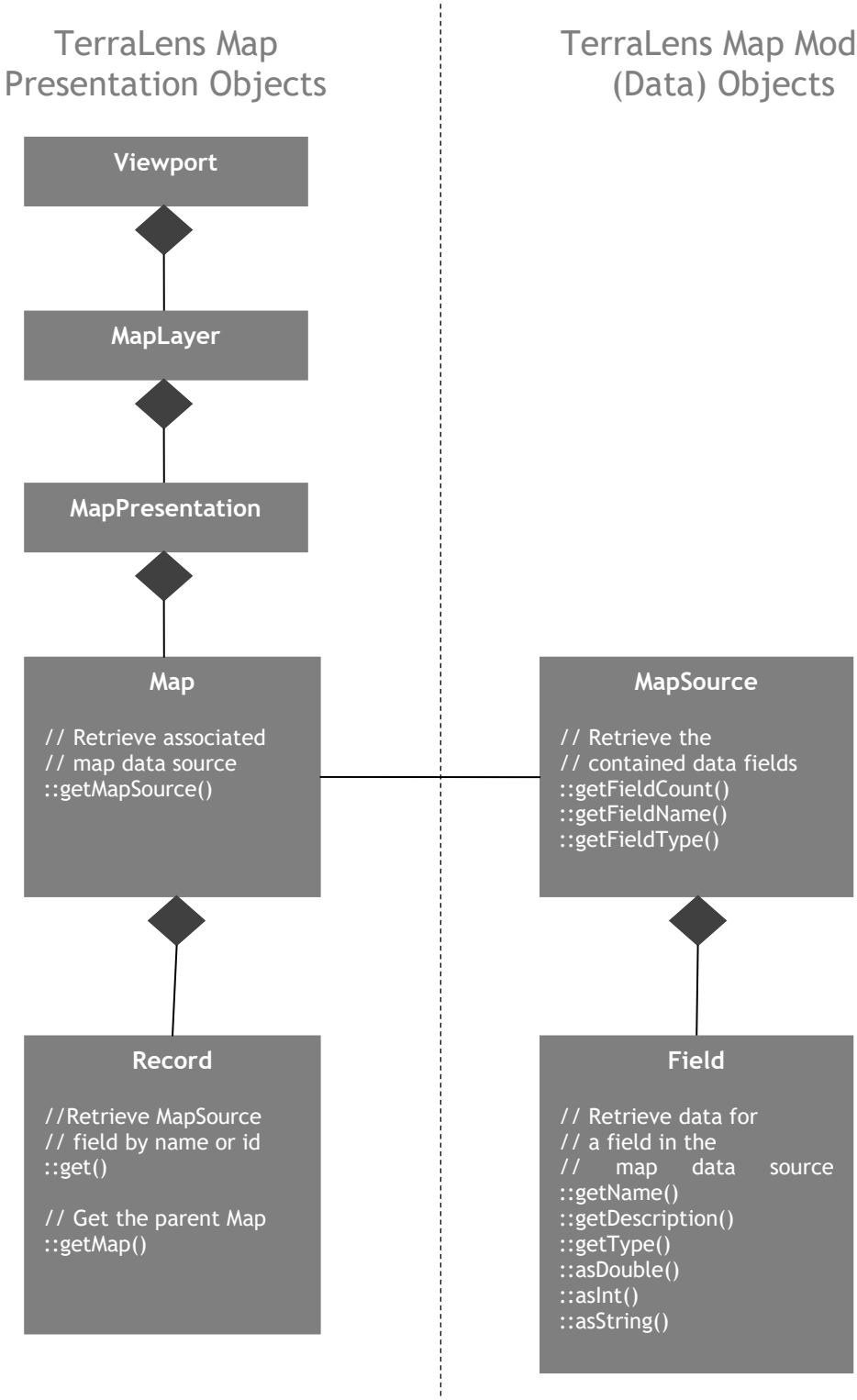

Figure 2: Retrieving Data from a Map Pick

Common Questions about TerraLens 8 Mapping

Do I need to rewrite my existing InterMAPhics 7.x or 8.x map handling code?

Not at all! TerraLens 8 still fully supports the previous API and mapping support through the map classes specialized for format (e.g. **TileSource**, **VectorMap** etc). Some methods were moved to their parent classes so be sure to recompile any components using TerraLens 8 as well as your main application.

Can I use mix the InterMAPhics 7 and 8 map approaches?

Yes, you can handle some maps using the InterMAPhics 7 approach, and others using the new TerraLens 8 classes. For instance, if your application has a well defined way of loading S57, you can keep using the S57PresentationView/S57Presentation classes and add the new generalized map support for any other format to easily extend the map display capabilities of your application.

When is it better to use the existing InterMAPhics 7 map classes?

Maintaining the data associated with the map features, and applying rules based on those data values, provides great flexibility to the application developer. However, it does add computational complexity and so impacts performance. The impact in most applications will be minimal. If you are building a new application, you should use the new mapping approach.

If however you have an existing application using the specialized S57 classes i.e. **S57PresentationView**, and are handling a lot of data either due to a large coverage area or high-resolution data, you should keep using your current approach for now.

A goal of our next release is to further improve the performance of the new mapping approach.

I have a set of maps pre-processed using InterMAPhics 7 – do I have to re-extract them under TerraLens 8?

Tiles created using InterMAPhics 7 are still usable with TerraLens 8. However the individual map feature data is not stored in the extracted tiles, so you can't pick individual records or apply rules based on their data values until you reprocess the maps using TerraLens 8.

I want to preview and pre-process maps using TerraLens 8, but have to support legacy applications that use InterMAPhics 7. Is this supported?

Using the new `MapSource::extract()` method you can specify the **MapFormatType** as either "Version8" or "Version7" – by default extraction will use the "LatestVersion" (currently Version8).

Is there an easy way to start defining map style sheets?

You can use the **MapStyleSheet** and **MapStyle** classes to define the visual attributes that should be applied to a map at runtime, and then use `MapStyleSheet::saveToFile()` to save the settings to external style sheet file. The style sheet is formatted in JSON format and can be modified manually, although it is safer to use the API to ensure proper formatting.

The next TerraLens release will include a new Map Preparation tool, providing a GUI based application in which to preview maps, define the look and feel in Style Sheets, and pre-process the maps into the TerraLens tile format.

5 Problems Addressed

The table below lists problems addressed since 9.2; many of the fixes are also backported to previous versions as appropriate.

The corresponding Geospatial Change Request (GCR) number is provided to facilitate communication with Kongsberg Geospatial Customer Support.

GCR No.	Problem Description
TLC-1497	New define allowing applications to disable std template exports from TerraLens
TLC-1644 TLC-1290	License changes - Disable new license manager feature that was performing a broadcast on the local network, even when a local file was available. - Add option to enter license by a string
TLC-1541	Support newline handling allowing multi-line strings to be displayed using one primitive.
TLC-1534 TLC-1576 TLC-1234	Improve legibility of fonts and glyph symbols.
TLC-1540 TLC-1516	Improved ScreenString and ScreenPolyline performance
TLC-1325 TLC-1293	ESRI Shapefile improvements including: - ESRI Shapefile point maps were not labelled correctly - Support ESRI UTM using the legacy Mapping API
TLC-1253	Handle WKT projections with an authority specified
TLC-1004	Fix for map data projection conversion using UTM system.
TLC-1419	DAFIF includes ARF_PNTs that are to be displayed as a polygon
TLC-1239	DAFIF extraction issues - added support for DAFIF version 8.1 - handle missing folders
TLC-1225	Handle folder inconsistencies in CADRG map sets
TLC-1492 TLC-1416	SetLocale() issues on Windows - Use consistent locale for writing/reading extracted maps - JSON Stylesheet parsing fix to support different locale

TLC-1559 TLC-1482 TLC-1445	DateLine improvements: - Fix for extents that cross the dateline - Fix for isIntersecting() calculations along dateline - DAFIF lines incorrectly wrap when crossing dateline
TLC-1505 TLC-1511	S-63 loading optimizations
TLC-1641	Fix over-release of critical section in S57 identified by analysis application.
TLC-1648 TLC-1213	Improved multi-threading when loading/closing S57 map sets
TLC-1645 TLC-1491 TLC-1477 TLC-1460 TLC-1332 TLC-1309 TLC-1384 TLC-1277 TLC-1065	Updates to S-57 / DNC symbology including: - Symbol fill is slightly off due to rendering extra outlines - Support for sizing symbols comprised of multiple parts - Depth symbol readout incorrect following resize - Added Ferry Sites and Buildings - Apply light rotation when zooming between nav levels, changing color table - Properly apply customized color tables with/without alpha values - Add BF010 DNC symbols for bottom characteristics
TLC-1514 TLC-1474 TLC-1464 TLC-1463 TLC-1462 TLC-1461 TLC-819 TLC-818	Updates to WMTS support via MapSource including: - WMTS MapSource disappears at high zoom levels - MapSourceConfig to improve URL parsing - Provide monitoring of tile loading errors - Long shutdown delays if waiting for tile requests - Some issues handling pure KVP WMTS servers - getCapabilities_XML() not working for parent source, just child - Robustness in querying specific WMTS servers
TLC-1542 TLC-1532 TLC-1531 TLC-1525 TLC-1518	Updates to WMS support via MapSource including: - Display correct resolution for server layers providing MaxScaleDenominator - Use the preferred CRS projection format when available - Add support for the MapSourceMonitor - Robustness in forming requests to the server - Support servers that redirect requests
TLC-1611 TLC-1629 TLC-1578	Updates to MilStd-2525 symbology including: - Add civilian color option - Add feature to disable text boundary boxes

TLC-1577 TLC-1567 TLC-1457 TLC-1446 TLC-1432 TLC-1425 TLC-1367	- Improve performance of Organisation Boundary polyline - Fix crash in release mode when generating bitmaps of Mil symbols - Support for App-6B symbology - Small icons missing edges - Unknown tracks displayed with flat edge - Long load times for symbols with large font sizes due to halo - Added missing signals intelligence symbol to Mil2525C
TLC-1422 TLC-1430 TLC-1442 TLC-1452	Updates to improve thread safety in autopositioned classes
TLC-1405 TLC-1400 TLC-1360	Clean up memory leaks: - S57 symbols - At shutdown in CLI - GL context - MapPresentation
TLC-1220	CLI exceptions during garbage collection
TLC-1447	Crash picking on 3D volume
TLC-1435	Restrict PresentationLayer::multiPick to return results only from its layer
TLC-1370	Fix Viewport::multiPickMaps crash due to DNC features with no default symbol.
TLC-1358	Cap interpolation values for GeoPoints
TLC-1242	Changes in atmosphere presentation from 8.3
TLC-1221	Improve thread safety for Intersection methods
TLC-955	Improve performance for HD3000 GPU on Windows10

Table 5: Problems Addressed in TerraLens 9.3

The table below lists problems addressed since 9.1

GCR No.	Problem Description
TLC-1004	Fix for map data projection conversion using UTM system.
TLC-967	Fix folder case sensitivity for DTED data on Linux (Lat and Long folders must match case)
TLC-809	3D models fail to display without an explicitly set Material
TLC-621	S57 maps should use LOWACC symbol rather than printing out a string
TLC-660 TLC-577	S57/S63 MapSource and Feature crash possible on deletion in multi-threaded environment
TLC-575	S57 symbols should use size defined in specification
TLC-543	Non-staggered fills should align for S52 areas
TLC-483	S57 Value Descriptions not correct S57 topmark symbols scaled improperly
TLC-470	S52 Color tables don't support alpha value
TLC-588	Memory leaks in Java and C# for extent rectangles in S57PresentationView
n/a	Transparent layers overlap along tile edges for extracted S57 area maps
TLC-544	WMTS request failures: Need to set a default CURLOPT_USERAGENT as some websites reject requests without one.
TLC-249	Don't save invalid WMTS tiles to the local cache
TLC-238	WMTS: Add support for KVP in addition to existing REST encoding
TLC-484 TLC-521	Better support for querying and specifying WMTS Layers
TLC-507 TLC-648	MapPresentation not displayed when range based filtering is in style sheet expression
TLC-571	Extraction of ESRI UTM takes a long time.
n/a	Crash loading point maps in 3D (in TerraLens 9.0 and 9.1 only)
n/a	Crash deleting filter expressions
TLC-678	Vector maps cause an assert when loaded in 3D
TLC-566	QueryDataRecords crashes on DTED if extent is too small

TLC-647	KML styles are not being applied
n/a	Support added for projection "CH1903+ /LV9"; added togws84 parameter to the prj string.
n/a	GeoSectorArc leaking memory if no inner radius is defined.
n/a	3D Geo Primitives are not pickable (in TerraLens 9.0 and 9.1 only)
TLC-579	Crash creating text and resources if onSize() is not yet called
TLC-527	Improve precision of GeoPrimitive intersection tests
TLC-550	Crash on PointCloud when changing resolutions
TLC-565	Added multi-threading safety to Radar Scan Converter
TLC-469	Allow ScreenString::getExtent with no active viewport

Table 6: Problems Addressed in TerraLens 9.2

The table below lists problems that were addressed since 9.0

GCR No.	Problem Description
TLC-203	Add Java interface for Atmosphere controls
n/a	S-57 – Resolve non-deterministic display of “Call-procedure” symbols
n/a	Sunshade copy constructor not copying all fields correctly
TLC-268 TLC-269	For Vector maps, pointers were being reused potentially leading to memory corruption and a crash on Windows10
44074	Enhanced the Expression class with comparison operators
TLC-339	S57 data update files may not be applied in correct order resulting in missing map areas. Note this problem was introduced in Version 8.3.0.45 and may only occur with some data sets.
44072	ESRI Shapefile prj file values not handled for edge cases, so the map is not displayed
44064	Setting and clearing a ScreenString multiple times causes a memory leak

Table 7: Problems Addressed in TerraLens 9.1

The table below lists problems that were addressed since TerraLens 8.3.

GCR No.	Problem Description
n/a	Add BNG (British National Grid) support
44048	Crash when DAFIF7 features were included in DAFIF8 dataset
44046	AML features that are scaleless take too long to extract
44025	Default colour ramp for elevation maps displayed using Map is not the same as for legacy class ElevationMap
44021	A MrSID map exported with the newer MG4 format will appear black
44018	AML not loading specific cells from the catalog
44012	GeoArea primitives' edge line did not display anything on 3D
44010, 44006	Various crashes displaying S57 in TerraLens Creator when moving between features
43996	GeoPolygon not filled properly under certain projections
43990	GeoTIFF map tiling too slow (added extraction option to use Dynamic Tiling)
43980	Deleting a BasicMapSet immediately after changing the view range causes a crash
43975	S57 Presentation View Extents should use Rhumbline
43971	GeoSectorArc – setting delta angle and radius of 0 causes crash
43968	SDT extracted with setTitle parameter cannot be exported to DTT
43965	Areas with delta latitudes over 180 degrees will not fill
43962	System Styles should be applicable to MapServer/WMTSServices
43961	Area vector maps do not properly reproject when the view is panned, resulting in gaps
43953	Errors with Records returned by pickMapDataRecords
43954	Extraneous data seen in Contour Data display
43950	Support tiffs with less than 8 bits per color channel (Photometric Palette)
43948	Produce terrain DTTs from SDTs using higher detail level if available
43947	Attributes with Double/Real values are converted to Integers in Tiling
43946	RGBA TIFF with 16 bit color channels
43945	Viewport3D.getGeoPointAtTerrain(GeoPoint) Missing API in Java

43944	GeoImage.setTexture and OpenGLTexture APIs are missing in Java
43938	ESRI Shapefile with complex polygon doesn't show
43920	Query elevation from native S102 Data
43917	Return text string associated with Chart Feature Enumerated Types
43914	S102 supported via MapSource
43908	TransparentColors cannot be set through StyleSheets
40617	Handling space characters in paths on Linux and UNIX.

Table 8: Problems Addressed in TerraLens 9.0

The table below lists problems that were addressed since TerraLens 8.2.1.

GCR No.	Problem Description
43907	Viewport using a render strategy of OpenGLTextureRenderer throws GL errors after resizing
43901	S57 display missing features when feature was provided as two primitive types (i.e. area and label)
43899	Calling Radial Grids setRGBA mode caused the presentation to not be displayed
43850	Deleting a MapLayer which has never been added to a Viewport crashes
43821	Java only: MapSource crash on DNC map extraction with expression
43784	Visual seams when merging a large number of geotiffs
43781	Memory leak in map tiles; aggravated by 4K display and large collection of maps
43752	Displaying a GeoSectorArc with start angle and delta angle reset to 0 causes pick to crash
43750	Added robustness to RPF feature/file detection on Linux

Table 9: Problems Addressed in TerraLens 8.3

The table below lists problems that were addressed since TerraLens 8.2.0.

GCR No.	Problem Description
43729	Extract larger ESRI extent to ensure entire shape outline is included
43706	MapDataRecord does not handle Integer values correctly
43681	Text on Tactical graphics should be upright – controlled via new method
43675	Crash using 3D model
43674, 43651	Issues with MGRS Points in polar regions
43662	Memory Leak on closing viewport
43661	GeoLine setRhumblineInterpolation is doing nothing
43655	Allow the default S52 symbol (question mark) to be turned off
43650	CADRG Rotating 3D Viewport Memory Leak
43609	GeoImage display JPG's colors incorrectly
43608	GeoImage does not obey its corners when the image is upside down
43596	CompassRose labels color correction
43467	DtedAltitudeUtility throws exception with certain directory structure
43445	Apply world files to image map files when using MapSource
43377	Filtering by attribute name not applied when extracting or querying maps
37828	getProjectionType() returns wrong type for ""PolarStereographic"
n/a	Gaps in display of high resolution CIB maps
n/a	Rendering issue for S57 area data with holes
n/a	2D Primitives not picking in 3D
n/a	GeoSurfacePolygon should interpolate points to match GeoPolygon
n/a	Viewport3D::getGeoPointAtTerrain() crash when lon<-180
n/a	Elevation value can't be queried from an extracted DTED map data through the "getElevationAt" API of MapSource object

Table 10: Problems Addressed in TerraLens 8.2.1

The table below lists problems that were addressed since TerraLens 8.1.

GCR No.	Problem Description
43534	Geo3D to World3D to Geo3D with EGM96 provides inconsistent altitude
43532	GlyphSymbol vertical alignment should be based on individual character
43517	DTED mixed case extraction via BasicMapSet::copy on Linux
43504	multiPickPresentation does not work on Java
43476	Ability to load a dted subfolder
43472	PolarPoint3D negative elevation angle error
43463	Complex Polygon Inner Boundary issues
43454	GeoPoint Rhumbline calculations cannot handle Latitudes of >= 90 Degrees
43434	VectorMap setTextOffset does not appear to work for 3D
43435	VectorMap getTextOffsetX always returns 0
43432	MultiThreading Race Condition with two DNC MapSources
43431	Java: Core on clean up involving Map and MapStyleSheet
43430	S63 fails to load text files with /r/n combinations in Linux
43429	MapSet copy fails on certain ESRI data
43426	Java: MapStyle.getLabels() returns null
43425	Java: core on mapStyleSheet.clear()
43417	Java crash with mapSource.GetMapSourceInfo()
43416	Java map extraction with no specified extent extracts nothing, but should extract whole area
43390	TerraLens crash after removing map presentation
43389	GDI: FileSymbol does not support PNG
43386	GeoPoint3D.move doesn't work
43385	Large GeoRectangles do not display correctly in Orthographic projection
43374	Memory Leak: SetFixedProjection

43373	Memory Leak: MapSource getSourceInformation()
43371	S63 fails to load older format
43341	Memory Leak: GeoArc setCenter
43339	Fix the issue of PrimitiveIntersection::getIntersectingArea
43332	CIB Anomaly - bounds range fix
43322	MapSource does not collect Image (JPG, PNG, BMP) maps
43270	Crash after loading large amount of maps
43231	ScreenImage CLI Contructor with 4 ScreenPoints crashes program.
43194	WorldCircle in 3D not drawing in its proper visual order
43161	World Sphere projection not supported
43153	Mil2525 Symbols using the frame modifier not appearing at small sizes
43115	Outline clipping
43106	MGRS format in Polar Regions (leading zeros)
43101	ScreenPolygon object drawn with extra lines when 2 points are identical
43100 42758 42136	Screen Sector Arc fixes: - ScreenSectorArc with angle (0-360) doesn't render - ScreenSectorArc smoothness
43065	Clipping S57 features in rotated views
43023	Screen text characters move as the view is panned
42850	Connect 7x esri labels to 8x MapSource
42826	DTEDAltitudeUtility calculateMemoryRequirement always returns 0.0
42696	GeoComplexPolygon crash
42614	Viewport update fails to update layer after calling remove
42158	Intersection between some primitives did not return correct values
39292	ScreenImage displays white square
39029	BasicGroup visibility issue on 3D viewport

Table 11: Problems Addressed in TerraLens 8.2

The table below lists problems that were addressed since TerraLens 8.0.

GCR No.	Problem description
n/a	Java TerraLens applications may experience blanking and flashing as the viewport is updated. To prevent this, the following flag should be used at startup: '-Dsun.java2d.noddraw=true'
42089	Order of mixed Hebrew, English and numbers different in OpenGL version
42112	getColorRGB and getFillColorRGB return color as ARGB instead of RGBA values
42272	Windows icon files not selecting correct size based upon image size
42696	GeoComplexPolygon not supported in OpenGL Java
42819	S57/63 Traffic Separation Scheme arrows not rotating when view rotated
42937	Large range change causes an update delay when S57 is loaded
42970	Support for "full" level of detail extraction added to CLI and Java
42960	Allow applications to specify key and value types used by KeyedModelManager

Table 12: Problems Addressed in TerraLens 8.1

6 Limitations

Limitations are bounds set on the use of TerraLens. Known limitations are as follows:

API		Synopsis	Description
C++	**Java**		
	✔	Java does not support a windowless viewport constructor	
✔		Applications must use Multithreaded DLL C Runtime Library (Microsoft Windows Only)	An application should always be built using a C runtime multithreaded DLL (Non Debug) on windows. To set the multithreaded DLL setting in Visual Studio: 1) From the menu bar, select the “Settings…” menu item in the “Project” menu. 2) The “Project Settings” dialog will appear. In this dialog, select the “C/C++” tab. 3) In the “Category:” pull down menu, select the “Code Generation” category. 4) In the “Use run-time library:” pull down menu, select “Multithreaded DLL”.
	✔	Compiling and Running Reference Implementation	Two batch files: compile_and_jar.bat and run.bat, are included in ReferenceApplicationJava. The batch files require a JAVA_HOME environment variable to be set before compiling, archiving, and running the Reference Implementation The work around is to modify the batch files themselves to define the JAVA_HOME environment variable. You may need two variables, one for the location of javac and jar, the other for the location of java
✔	✔	Video Card Drivers for 3D (OpenGL) and OpenGL Driver	In order for the OpenGL version of TerraLens to function properly, it is essential that video card drivers be up to date with the latest OpenGL driver support.
		All machines running CLI applications require the .NET Framework	For a CLI application to run on a Windows machine, it needs an up-to-date version of the .NET Framework. This can be obtained by downloading Dotnetfx.exe from MSDN.
✔	✔	Fonts and bitmaps do not rotate on X11	Fonts and bitmaps cannot be rotated using TerraLens X11 graphics version. This is not a limitation with Windows, nor OpenGL versions.

Table 13: Known Limitations of TerraLens

7 Known Problems

Problems are considered to be outstanding issues with the usability of TerraLens where the intended or documented behavior may not be observed or where the expected results are not produced.

Multiple problems can apply to the same feature. The corresponding Geospatial Change Request (GCR) number is provided for ease of reference. For further information, refer to the Customer/Technical Support Handbook. Workaround instructions are provided where possible.

API	[Component] Feature	Synopsis	Description	GCR #
All	[IVC]	3D model formats not supported	The legacy formats b3d and hmp are not handled in TerraLens 9.1	TLC-345
All	[IAC]	InputEvent Handler not compatible with OpenGLRenderer	The InputEventHandler requires a system window handle, which is not used with the OpenGLRenderer. For the renderers, user input must be fully handled by the application.	43673
All	[IVC]	StyleSheet label needs color	If the text color is not set in a style sheet's label definition, that label will not be applied	43412
All	[IVC]	Halo not available in 3D in TerraLens 8	In TerraLens 8, the halo effect is disabled for primitives in the 3D display. It is enabled in TerraLens 9.	43338
ALL	[IVC]	Pixel width not applied in 3D in TerraLens 8	In TerraLens 8, the user specified pixel width is not applied to the lines in 3D, as TerraLens automatically applies a varying line width to create a 3D effect.	42459
All	[IVC, AM]	Extraction different from previous release	Some vector text feature extractions are not producing the same results as in Version 7.4, showing more data than previously.	42151
All	[IVC]	snapShot() captured wrong window	On Linux, Viewport::snapShot() captured the window in focus on top of the viewport rather than just the viewport.	42145
All	[IVC,AM]	Map Data Source problems handling mixed case files on Linux systems	For DNC data and Raster a.toc files, source data files are handled if they are all in upper case, or all in lower case.	41460 41918
C++	[AM]	Limited map and symbology support on HP-UX and vws7	Due to limited 3rd party library support for these compilers, the following map formats cannot be extracted: MrSID, JPEG2000, ARCS, AML, S63. A previously extracted	40456

API	[Component] Feature	Synopsis	Description	GCR #
			map of these types can however be displayed. The MIL-STD-2525 symbol library is not available on these platforms.	
All	[IVC] 3D Fill Style	Missing 3D fill styles for primitives	BasicFillHollow and BitmapFill are not supported for world and screen area primitives	38318

Table 14: Known Problems

8 Obtaining Support

If you have any questions or comments, do not hesitate to contact Kongsberg Geospatial Technical Support at:

tech.support@kongsberggeospatial.com

Further information can be found in the Customer/Technical Support Handbook, which is provided on each TerraLens release CD-ROM or DVD.

www.kongsberggeospatial.com

Printed in Canada