TerraLens

Product Overview

KONGSBERG

Kongsberg Geospatial

Table of Contents

Table of Figures

Preface

About the TerraLens® Platform

The TerraLens platform, formerly InterMAPhics®, is made available as a family of related products, which include the following:

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

TerraLens UI replaces the InterView development foundation.

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 this URL: https://www.kongsberggeospatial.com/products.

About this Document

This Product Overview provides a brief introduction to the key features and functions of TerraLens, the advanced Human Systems Interface (HSI) software-development solution from Kongsberg Geospatial. It includes a description of the product’s technical differentiators, its architecture and required specifications, as well as a glossary and other useful appendices.

While this document refers to “Human Systems Interfaces” (HMI), the phrases “Human Machine Interface” (HMI), “Computer-Human Interface” (CHI), or simply “User Interface” (UI) can be understood as interchangeable.

The TerraLens platform offers multiple programming interfaces for developers: C++, Java, and CLI (.NET). All interfaces provide similar core functionality, although some differences exist due to various programming language environments and platform differences.

For the purpose of this document, TerraLens can be understood to refer to the TerraLens Core SDK, unless otherwise stated.

This guide presents the key features of TerraLens at a high level, and is relevant to all programming interfaces.

For detailed technical information, see the TerraLens Programmer’s Guide, or turn to Section 5.0, ‘Where to Find More Information’, which contains a full list of related documents and publications.

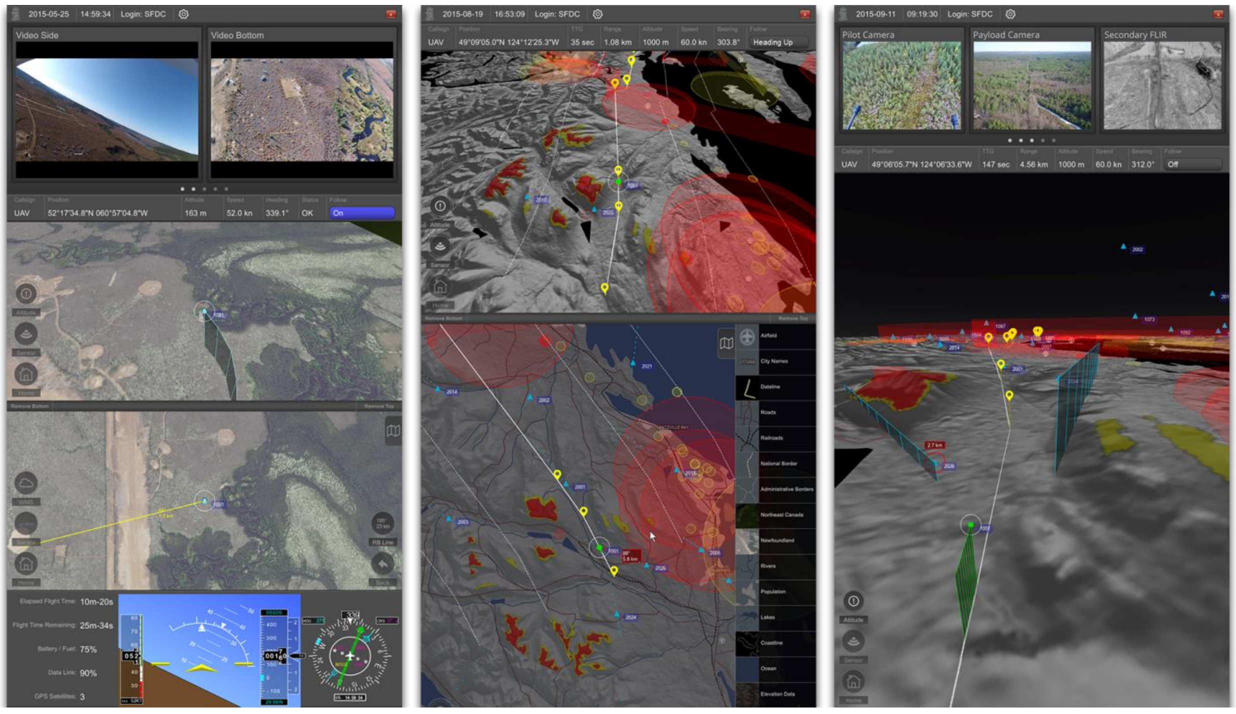

2015-05-25 14:59:34 Login: SFDC
Video Side
Video Bottom
2015-08-19 16:53:09 Login: SFDC
2015-09-11 09:19:30 Login: SFDC
Pilot Camera
Payload Camera
Secondary FLIR

Figure 1: Three applications created using the TerraLens UI foundation. All three are powered by the TerraLens Core SDK

1.0 Introduction

TerraLens® Core: A Trusted Geospatial Engine for Mission-Critical Applications

Refined and enhanced over more than two decades, TerraLens is the world's most-used HSI solution for operational Command and Control systems. It provides developers with every necessary feature to build mission-critical display systems for defense and air traffic control applications, as proven through more than two decades of successful deployments. TerraLens is at the heart of critical infrastructure in over 50 countries, key applications developed for:

- Air defense
- Missile defense
- Air surveillance
- Naval command and control
- Homeland security
- Air traffic control
- Vessel traffic management

TerraLens is Versatile

A high-performance, geographically based dynamic data-visualization product, TerraLens has a wide range of applications. It can be used to build the HSI of a performance-critical system at every stage of its lifecycle: prototyping, requirements definition, and development and maintenance applications. TerraLens supports a variety of Windows and Linux platforms, in addition to Android. And it enables Object-Oriented development in C++, Java, and CLI (.NET).

Flexibility for Developers

TerraLens is not a predefined HSI; it's a toolkit for building the HSI you require, with a platform-independent API that enables greater portability of programming. TerraLens is unconstrained, allowing developers to build any style of user interface to any standard required—and it permits deployments in both 2D and 3D.

Power and Performance

TerraLens delivers consistently fast refresh rates and low CPU utilization levels, and has been engineered to deliver high performance on low-power systems like ARM. Its unique capabilities allow it to process more than 30 military and commercial aviation map projections on the fly.

Where TerraLens Works Best

TerraLens display applications include Command and Control (C2, C3, C3I, C4ISR), ATC and others that present moving objects in a geographical context.

Specifically, TerraLens is ideal for developing applications that require:

- a geographical context for the main display
- display of geographical navigation overlays

- geographically positioned objects (including tracks) with frequently updated or real-time data (including position)
- tabular data displayed to the operator
- operator interaction to update the geographic context presentation and to modify display object characteristics through menu buttons and data input

TerraLens simplifies the transition between 2D and 3D displays—allowing views of common data to be presented in 2D or 3D windows for maximum clarity and visual impact. It updates thousands of tracks and plots per second with responsive operator interaction and minimal CPU utilization, and maximizes display performance through multiple redrawing strategies.

Benefits TerraLens Provides

TerraLens has the advantage of being a uniquely focused product. It visualizes data provided by input systems rather than serving as a data source itself, displaying fused and correlated tracks without having also to serve as the track correlator or fusion engine. TerraLens similarly provides the HSI for a GIS.

Most importantly, TerraLens is not a constraining or pre-defined HSI. Instead, it is a toolkit for building the HSI your particular system needs.

Combined, the benefits of TerraLens reduce your development risks and enable rapid deployment of applications—by reducing the amount of new code written (bringing down testing, documentation, and maintenance costs), and by providing a runtime graphics display engine that has been fine-tuned and field-proven in many programs over many years.

TerraLens permits genuinely collaborative development while allowing you to control project costs. And it is backed by Kongsberg Geospatial's longitudinal lifecycle support, with technology refreshes and legacy upgrades provided as required.

For all its sophistication and breadth of capabilities, TerraLens is surprisingly easy to learn, requiring just a few days of familiarization for a developer to get up to speed. That means you're able to move from planning to implementation and deployment faster—and accelerate your return on investment.

2.0 Technical Differentiators

What Sets TerraLens® Apart

TerraLens includes a full set of Object-Oriented components designed to accelerate and simplify the development of mission-critical situation-visualization systems, achieving greater reliability and performance than competing solutions—at significantly lower rates of resource utilization.

Multi-dimensional Display Capability

TerraLens allows developers to create 2D and 3D displays easily, and to build applications that include both 2D and 3D windows. TerraLens also affords the flexibility to add 3D windows to an existing 2D application. TerraLens capabilities are provided through a standard class library and integrate seamlessly with standard development tools.

Object Oriented API

From graphics primitives and mapping components to geographic and screen units and cameras for manipulating geographic contexts, TerraLens provides all the Object-Oriented components developers need to build the front end of a system efficiently and effectively.

Being fully object oriented, the API will feel immediately familiar to any experienced object-oriented programmer. The class and method names are clear and meaningful to avoid the need of referring to documentation. When appropriate, class interfaces mimic standard ones, such as the Standard Template Library interface for containers.

High-performance Runtime Graphics Display Engine

The real-time high-performance runtime graphics display engine that underlies TerraLens delivers the maximum possible information updates to your display using a minimum of system resources. It also synchronizes and registers track and map data for the application—and has demonstrated its reliability in a variety of operational situations over many years.

TerraLens has been engineered to facilitate geospatial visualization and real-time geospatial data in particular. This includes optimizations for the display of 3D terrain data, and for fast tile loading, caching, and culling as the camera view changes.

Integrated math libraries provide capabilities specific to geospatial visualization, such as dynamic line-of-sight and proximity calculations.

TerraLens also guarantees a high degree of precision in the display of high-resolution terrain, and the placement of objects relative to Terrain. The capabilities exhibited by a particular application built with TerraLens will necessarily depend on the quality and resolution of the Terrain data provided to the application, but TerraLens demonstrably provides resolution to within less than 2.5 cm per pixel.

TerraLens can consume a wide variety of map formats without pre-processing, and can display layered maps from multiple sources and in multiple formats at run-time. This is particularly useful in quickly developing applications that fuse data from a variety of sources or services. For example, an application might include a map layer with high resolution satellite imagery of the ground, S-57 charts of maritime features, and an overlay showing real-time weather. With TerraLens, these map layers can all be sourced from different data providers, and in different formats.

TerraLens also makes it easy to include drawing tools, and 2D and 3D geometries in your applications. TerraLens provides a library of primitives objects, but will also allow the application to load and display 3D models in a variety of formats, including 3D Studio Max (3DS).

The built-in object library also makes it easy to provide rich interactive functionality, including individual object picking – with the ability to retrieve and display a wide range of data associated with map objects. For example, this gives users the ability to interact with maritime navigation entities like buoys and channel markers in S-57 charts.

The same powerful, high performance of the TerraLens Core engine is the foundation for the TerraLens Server, which serves out the OGC compliant Web Map Tiles.

Advanced Threading

TerraLens is designed specifically to achieve maximum performance across varying throughput and responsiveness profiles in multi-threaded environments. Using TerraLens, experienced threading programmers can implement the following threaded architectures:

- Separate processing threads for each external data source, responsible for updating the model classes that represent external, real-world data objects. This supports the de-coupling of external data interfacing and the efficient processing of multiple external inputs.
- Separate processing threads for each external data source, responsible for updating both the model classes and their visualizations. This supports the need to reduce display latency or time-to-glass.
- Separate processing threads for each viewport, responsible for all processing associated with managing, updating and interacting with presentations contained in that viewport. This supports the need to maximize throughput performance.
- Separate processing threads for user input, responsible for all processing associated with user interactions. This supports the need to reduce input latency (responsiveness).

Combined, these technical differentiators help provide the many benefits of TerraLens as described in Section 1.0, namely:

- Reduced program risk and increased reliability through use of a field-proven runtime graphics display engine;
- Accelerated software development through provision of a familiar object-oriented programming environment;
- Improved application programmer efficiency and productivity through code reuse and technical flexibility; and
- Enhanced runtime performance of finished applications due to field-proven optimized use of system resources.

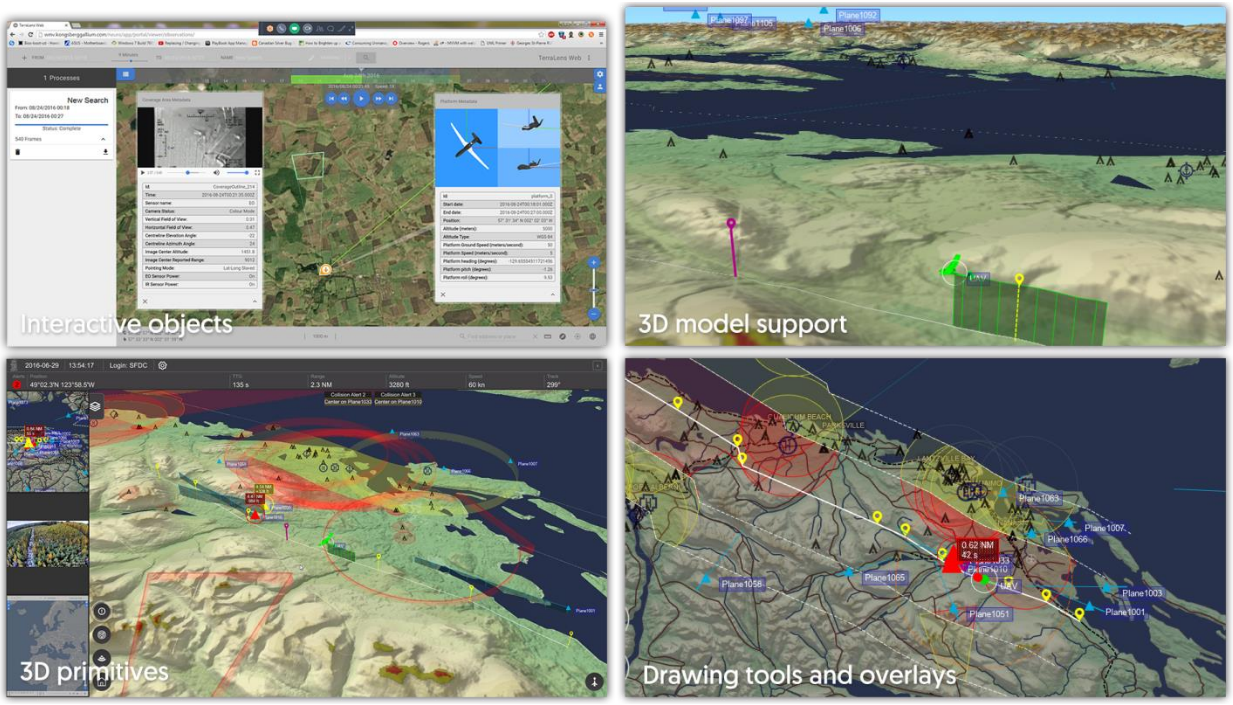


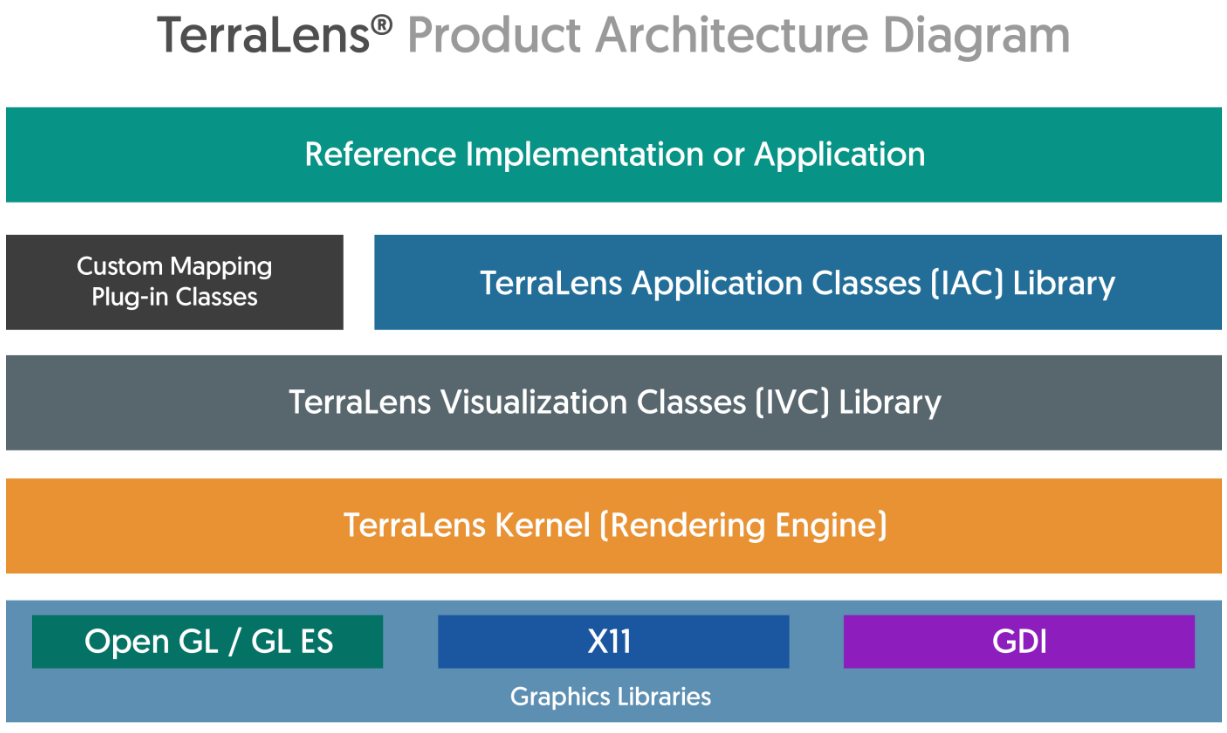
Figure 2: Screenshots showing a variety of the functionality provided by TerraLens Core, and TerraLens Web, including interactive objects, support for 3D models and primitives, and drawing tools and map layers and overlays

3.0 TerraLens Core Architecture

The Pieces - and How They Fit Together

TerraLens comprises a primary set of components that includes:

- TerraLens Visualization Classes
  (high performance graphics engine and display data management classes)
- TerraLens Application Classes
- Advanced Mapping Classes
- TerraLens Creator (GUI tool for styling and packaging map data from map databases)
- Reference Implementation


TerraLens® Product Architecture Diagram
Reference Implementation or Application
Custom Mapping Plug-in Classes
TerraLens Application Classes (IAC) Library
TerraLens Visualization Classes (IVC) Library
TerraLens Kernel (Rendering Engine)
Open GL / GL ES
X11
GDI
Graphics Libraries

Application Classes - provide higher-level display components built on top of the IVC lower-level classes

Visualization Classes - provide the core capabilities of the product. These classes offer all of the base support to display graphical objects in a geographic context and to process operator interactions with these objects

Custom Mapping Plug-in Classes - support integration of map data formats to supplement those provided in TerraLens

TerraLens Kernel - provides the underlying rendering functionality for higher-level libraries; The kernel performs all native graphics library calls to realize the display.

Figure 3.1: A simplified Product Architecture Diagram of the TerraLens Core SDK

TerraLens Core Visualization Classes

The TerraLens class libraries make it possible to visualize dynamic objects, such as tracks, moving over a map background. They provide the following graphics primitives and attributes as well as basic and advanced mapping components that allow mission-critical displays to be built quickly and easily:

- 2D and 3D graphic primitives
  Definable in Geographic, World, or Screen coordinates
- 2D and 3D view components
  For specifying window placement as well as window-specific pan, zoom, and projection parameters
- 2D and 3D mapping presentation components and containers
- Hierarchical container objects
  For organizing graphic primitives into groups and subgroups
- Graphic attributes
  For specifying the exact look of each graphic primitive
- Units of position, distance and orientation
  For specifying the spatial characteristics of a display object in the screen or real-world context—in either 2D or 3D

Class methods can be invoked by the application source code at runtime to perform a series of functions:

1. set the geographical context of the display view and control its orientation and scale;
2. control the position, visibility, and appearance of display objects; and
3. process inputs from the operator, other processes and external system sources.

The Visualization Classes by Category

The TerraLens visualization classes are typically separated into seven major categories:

Context Classes

These establish the context for your application. The camera provides pan and zoom capabilities; the spheroid model and projection establish the geographic context.

Container Classes

These create the containment hierarchy and provide control over the visual priority of objects contained within that hierarchy.

Mapping Classes

These are used to manage the geographical maps to be displayed in an application, including:

- Loading map data from external sources such as disk, CD-ROM or web services
- Pre-processing map data for faster runtime access
- Visualizing map data using attributes and rules according to user defined style sheets

Primitive Classes

These provide base-level graphical objects such as lines and rectangles within a TerraLens application. To be visualized, primitives must be added to a container class.

Attribute Classes

These define the graphical characteristics of primitives and maps: color, line style, font, symbol style and more.

Unit and Coordinate Classes

These make it possible to define the shape and position of graphical objects in different coordinates systems.

Error Handling Classes

These provide support for easy error reporting and logging.

For more detailed information on the class groups and how they are used, consult the TerraLens Programmer's Guide.

TerraLens Core Application Classes

The TerraLens application classes comprise a set of components that support your development process and enhance your ultimate application. They include:

Architectural components

To ease the implementation of a sound model-view-controller (MVC) design pattern in an application

Controller components

For platform independent user-input processing

High-level situation visualization components

Such as tracks, range and bearing lines and geo grids to accelerate prototype development, as well as support for Mil-Std symbology

Tracking cameras

For following moving targets in either 2D or 3D

Filter components

For classifying objects based on the value of a particular attribute

Network components

For implementing platform independent inter-process communication

Navigation utilities

For calculating geographic paths, estimated times of arrival, closest points of approach and other elements. To learn more about the TerraLens Application Classes, read the Programmer's Guide.

TerraLens Core Custom Mapping Plugin Classes

In TerraLens 8, the mapping support has been incorporated into the Visualization Classes providing a simplified programming interface along with the use of style sheets to define the map presentation.

The optional Custom Mapping Classes allow for the integration of map formats not explicitly supported by TerraLens Core. The parsing and organization of the map data is handled by the application code, and passed to TerraLens Core for visualization. Samples of these classes are provided with the TerraLens product.

Please see the Release Information Manual for additional information.

TerraLens Creator Map Management Tool

TerraLens Creator is a specialized tool that allows you to manage map data and other image assets, and prepare map packages for use within your TerraLens application. It gives you the ability to discover map data from external map databases or archives, choose cartographic features of interest, establish display attributes and define rules for when to apply them to the map, and package the selected data for use by TerraLens Core.

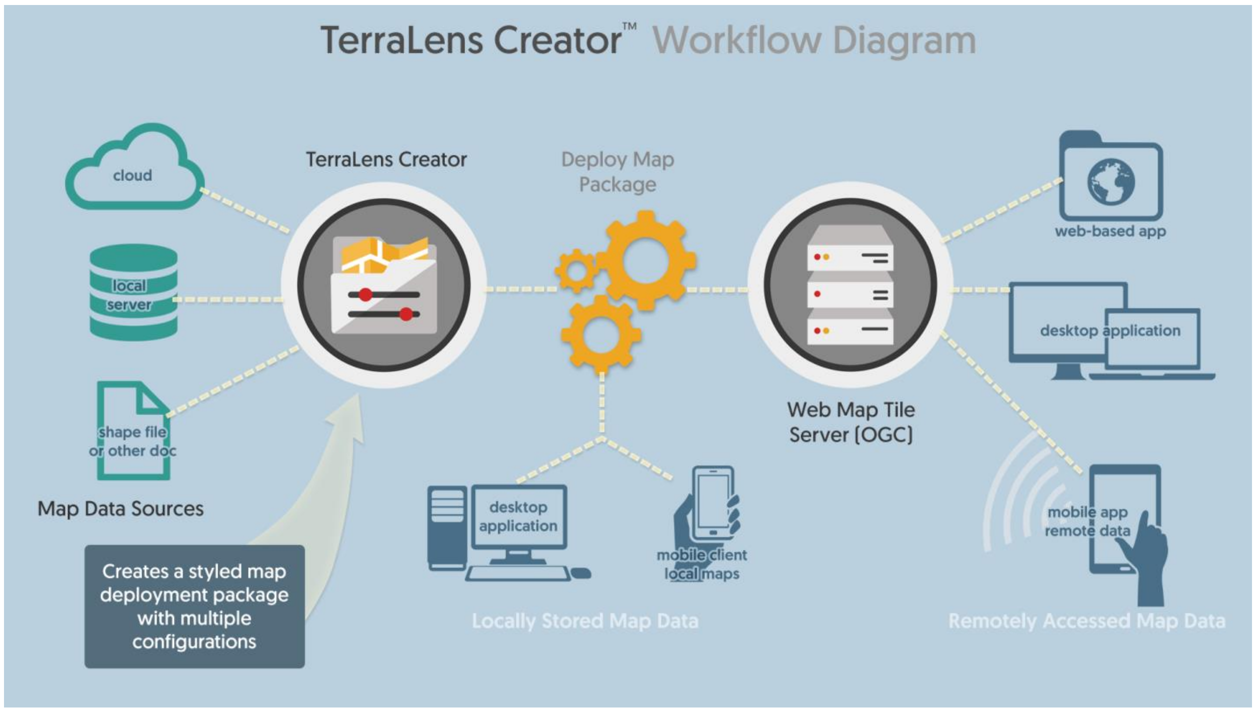


Figure 3.2: TerraLens Creator Workflow Diagram

The workflow diagram above demonstrates how TerraLens creator can be used to style and deploy map packages from a variety of data sources. The resulting packages can either be installed locally on a desktop or mobile client, or provided to multiple clients with an OGC standard Web Map Tile Server like TerraLens Server.

Reference Implementations

TerraLens includes an executable Reference Implementation: a clear, comprehensive basic TerraLens application that can serve as a model for your custom application. The source code of the Reference Implementation is fully accessible, open to modification or extension by your development team, or simply as a sample code reference.

Other Considerations

TerraLens also supports a number of third-party tools, and comes complete with an integrated documentation set.

Third-party Tools

TerraLens can be readily used in conjunction with:

GUI Toolkits

- Qt, QML
- Java Swing API; Java Abstract Windowing Toolkit (AWT)
- .Net
- MFC
- Motif

Development Environments

- Microsoft Visual Studio
- Java Integrated Development Environments (IDEs) such as Eclipse or JBuilder
- Native compilers and debuggers
- Drawing Tools

Visual Studio Support

The Windows version of TerraLens allows developers to build native-mode applications using Visual C++, or the .NET languages (e.g. C#) in Microsoft Visual Studio.

Reference Documentation

The following documentation is provided in HTML format for the C++, Java and .NET interfaces:

- TerraLens Core Class Reference
- Programmer's Guide
- Sample Applications

Graphics Libraries

TerraLens Core uses OpenGL for rendering, and can draw into a GL texture for integration with other GL components. Both OpenGL shader and fixed function pipeline rendering is supported, depending on your driver capabilities.

For devices without OpenGL support, TerraLens is also available for X11 and GDI graphics drivers, using the same TerraLens API.

TerraLens at Work

Figure 3.3 below depicts a typical system in which a TerraLens-based application drives the operator interface. The TerraLens runtime engine on each operator workstation executes an application developed using the TerraLens Class Libraries.

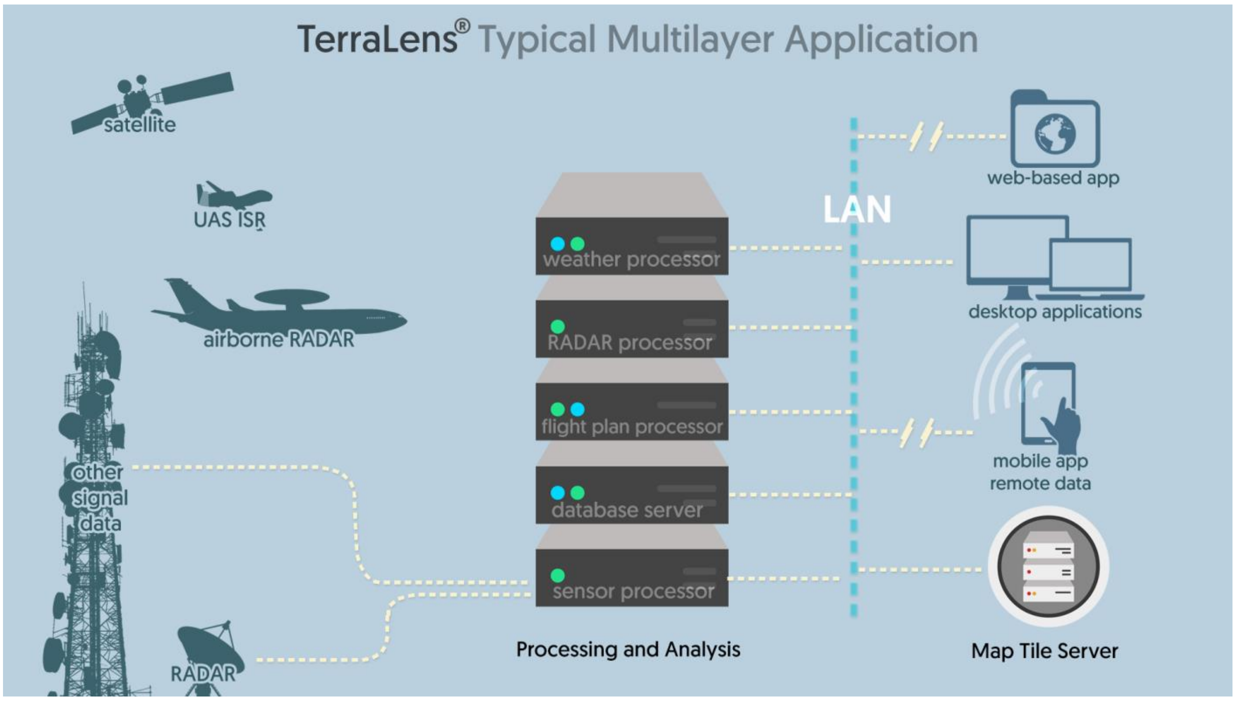


Figure 3.3: TerraLens-based display system in a typical multilayer application

In the Figure above, the Web Map Tile server enables networked map delivery. The TerraLens Server typically serves out map data to both thin and traditional thick clients. The TerraLens Server can provide data to multiple end users and is scalable.

4.0 TerraLens Core Specifications

Operating Systems and Compilers

At present TerraLens Core is available for use on the following operating systems and compilers. Other platforms are available on request.

For specific version information, please consult the Release Information Manual provided with the TerraLens Core release or contact Kongsberg Geospatial Customer Support.

Operating System	Arch	Compiler	API		
			C++	Java	CLI
Microsoft Windows	x86 x86_64	Microsoft Visual C++	✓		
		Microsoft Visual Studio .NET			✓
		Sun JDK		✓	
Red Hat Linux	x86 x86_64	GNU GCC	✓		
		Sun JDK		✓	
Android	ARM x86	Android SDK	✓	✓	
Sun Solaris[1]	x86	GNU GCC	✓		
		Sun JDK		✓	
Sun Solaris[1]	SPARC	GNU GCC	✓		
		Sun Studio	✓		
HP-UX[1]	Itanium	HP aC++	✓		

Figure 4: Table of available operating systems and compilers

1 Available for legacy InterMAPhics versions

Format Support

Map Format Support

Vector maps and overlays

- DNC (Digital Navigation Chart)
- DCW (Digital Chart of the World)
- NGA VPF: VMAP 0,1,2,3; WVS+/WVSPLUS (World Vector Shoreline)
- ESRI ShapeFiles
- S57/S52
- S63
- AML (Additional Military Layer)
- OGC GeoPackage
- KML
- GeoJSON
- DAFIF 6,7,8 (Digital Aeronautical Flight Information File)
- MBTiles
- NITF
- SVG
- CGM

Raster maps and images

- ARCS
- JPEG 2000
- MrSID
- GeoTIFF, BigTIFF, TIFF
- CIB 1,5,10
- BMP
- JPG
- PNG
- ADRG / CADRG
- OGC GeoPackage
- ECW
- MBTiles

Web Map Formats

- WMS
- WMTS

Elevation Formats

Supported elevation data display

- DTED Levels 0,1,2
- DEM as GeoTIFF
- OGC GeoPackage
- USGS SRTM
- S102

Graphics Rendering

Supported render pipelines

- OpenGL
- OpenGL ES
- GDI
- X11

Projections and Coordinate Systems

Supported Map Projections

Map projections can be dynamically updated at runtime

- Alaska Conformal
- Albers
- Azimuthal Equidistant
- Cartesian
- Equidistant Conic
- Equirectangular
- Gnomonic
- Goode
- GVNSP
- Hammer
- Interrupted Mollweide
- Lambert Azimuthal
- Lambert Conformal
- Mercator
- Miller
- Mollweide
- Orthographic
- Polar Stereographic
- Polyconic
- Robinson
- Sinusoidal
- Stereographic
- Transverse Mercator
- UTM
- VanDerGrinten
- WagnerIV
- Wagner VI
- WagnerVII

Supported Spheroid Projections

- Airy
- Australian National
- Bessel
- Clarke 1866
- Clarke 1880
- Everest
- GRS 1980
- Hough
- International 1909
- Krassovsky
- Mercury1960
- Modified Airy
- Modified Everest
- Modified Mercury 1968
- New International 1967
- Southeast Asia
- Sphere Model
- WGS66
- WGS72
- WGS84

Supported Coordinate Systems

- Lat/Long
- Georef
- UTM
- MGRS
- ECEF

Coordinate systems are fully extensible through application code

5.0 Where to Find More Information

The TerraLens Reference Family

All of the following documents are provided with the TerraLens family of products.

Document	Description	TerraLens
TerraLens Product Overview (TerraLens.PO.00)	Introduction to TerraLens (this document)	PDF
TerraLens Release Information Manual (TerraLens.RIM.00)	Record of updates, fixes and pertinent release information	PDF
TerraLens Class Libraries Reference	Documentation for every class and method offered by TerraLens in all 3 languages.	HTML
TerraLens Programmer's Guide	Programming notes and code samples for developers creating object-oriented display applications using TerraLens	HTML
TerraLens Samples	Documentation for the TerraLens Samples	HTML
Customer/Technical Support Handbook (TerraLens.TSH.00)	Description of customer support and training services provided by Kongsberg Geospatial	PDF

Figure 5: List of documentation and references for the TerraLens geospatial platform

Kongsberg Geospatial Contacts

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

1-800-267-2626 (North America)

+1 613-271-5500 (Outside North America)

You may also contact the Kongsberg Geospatial team at the following:

tech.support@kongsberggeospatial.com

sales@kongsberggeospatial.com

info@kongsberggeospatial.com

Further information can be found on our website www.kongsberggeospatial.com