OmniMap API
DirectX 9 and DirectX 10
Integration Guide

Version 2.0.0

Last updated: January 15, 2008

IMMERSIVE PROJECTION DESIGN
THE ELUMENATI
www.elumenati.com

Contents

Overview

This document is a guide for integrating the OmniMapD3D Geometry Correction Library API into existing DirectX9 applications. The document will provide an overview of the library's architecture, and how it fits into an existing DirectX9 application, followed by a case study of the steps required to integrate the OmniMapD3D API into a sample DirectX9 program.

For more information, see the *OmniMap API Configuration Guide* and the *OmniMap API Class Documentation* included with the OmniMapD3D installer.

Intended Audience

This document is written for programmers interested in implementing geometry correction within their real-time DirectX9 application for use with the Elumenati OmniFocus range of products.

Developer Requirements

- Microsoft WindowsXP
- GPU supporting DirectX9 and Pixel Shader 3.0
- DirectX SDK (December 2006 or later)
- Visual Studio 2005

Assumptions

This document assumes the developer has installed the OmniMapD3D API from the installer program *OmniMapD3D.msi*,and used the default location for the installation. The default installation location for the OmniMapD3D API is:

C:\Program Files\Elumenati\OmniMap

In this document, we will refer to this folder as <*InstallationDir*>. So if you have chosen to install the OmniMapD3D API in a location other than the default location, <*InstallationDir*> refers to that location.

OmniMapD3D API Components

The OmniMapD3D API is shipped with documentation, the example program referred to in this document, C++ header files, lib files and corresponding DLLs. The documentation includes this document, the API Configuration document, and detailed class documentation in HTML format.

The class documentation can be found in:

<InstallationDir>/docs/classdocs

The header files are found in:

<InstallationDir>/include

The library files can be found in:

<InstallationDir>/lib

The DLL's can be found in:

<InstallationDir>/bin

The example program can be found in:

<InstallationDir>/examples

OmniMapD3D API Architecture

The OmniMapD3D API is designed to be easily integrated into existing DirectX9 applications. The main class that provides the dome rendering functionality is called *OmniMapD3D*. The *OmniMapD3D object* is constructed with arguments for the width and height of the final output, and a pointer to the *IDirect3Ddevice9* that the application wants the final rendered output to be drawn to. Additional parameters specify the location of the startup OmniMap Lua configuration file, and the directory where the OmniMap Lua support files, and the OmniMap Effect file reside.

```
OmniMapD3D omniMapObj = OmniMapD3D::OmniMap(width, height,
d3dDevice, “OmniMapD3DConfig/omnimap_startup.lua”,
“C:\Program Files\Elumenati\OmniMap\Resources”);
```

To generate a scene for a dome, the OmniMap class needs to call the application's rendering function 3 or 4 times with a modified viewing frustum and viewing angle. The rotation and viewing angles of each channel are dependent upon the dome configuration specified in the Lua script files *omnimap_startup.lua*, and *omnimap_user_edit.lua*. Each of these render passes is managed by the *OmniMapChannelBase* class. In the application where the rendering of a single frame is done, a call is made to the *OmniMapD3D* object telling the *OmniMapD3D* object which function to call for rendering a single channel. That call looks like:

```
omniMapObj->ForEachChannel(renderFunction);
```

where *renderFunction* is a pointer to a function that takes a pointer to an *OmniMapChannelBase* object as an argument:

```
void renderFunction(OmniMapChannelBase *channel);
```

ForEachChannel will call the *renderFunction* once for each channel. The *renderFunction* must call *OmniMapChannelBase::beginRenderToChannel* before rendering the scene. This call will call ID3DXRENDERTOSURFACE::BeginScene to direct rendering to the dynamic texture for this channel. *beginRenderToChannel* will also set the DirectX9 projection matrix to contain the OmniMap projection matrix required to render the channel, and the correct rotation to render the channel. The application programmer can access these values through the *OmniMapChannelD3D* class interface. As each channel is rendered, the resulting image is rendered into a DirectX9 dynamic texture.

When all channels have been rendered, the application calls:

omniMapObj->PostRender()

This method will composite the channels and spherically project them onto the display.

Demo Program Case Study

This section describes what was done to create the project file for the Demo project located in:

<InstallationDir>\examples\Direct3D\Direct3D9\MultiAnimation

The same process applies to setting up the Direct3D 10 sample located in:

<InstallationDir>\examples\Direct3D\Direct3D10\BasicHLSL10

1. Open Visual Studio Workspace File

1.1 Open the Visual Studio Workspace for the dome enabled Demo program by double-clicking on “MutliAnimation_2005.sln” in this directory.

2. Add OmniMap include Directory.

2.1 Using the properties dialog for the MultiAnimation project, we first added the include path for the OmniMap API include files to the property:

Configuration Properties

C/C++

General

Additional Include Directories

That path is :

<InstallationDir>\include

In the project file, it is referenced relative to the Source directory. It is refererenced as ../../../../include.

3. Add OmniMap API Directory

3.1 Using the same properties dialog, we then added the library path for the OmniMap API library files to the property:

Configuration Properties

Linker

General

Additional Library Directories

That path is :

<InstallationDir>\lib

In the project file, it is referenced relative to the Source directory. It is referenced as ../../../../lib.

4. Add OmniMap library dependency.

4.1 Finally, using the same properties dialog, we added the OmniMap library to the list of libraries to link with the Demo application. That property is:

Configuration Properties

Linker

Input

Additional Dependencies

That library is *OmniMap.lib* for the Release configuration, and *OmniMapD_D3D.lib* for the Debug configuration.

5. Edit "MultiAnimation.cpp" or "BasicHLSL10.cpp"

5.1 All changes necessary for integrating OmniMapAPI D3D into MultiAnimation.cpp and BasicHLSL10.cpp are bracketed by #ifdef USE_OMNIMAP. Comments explaining each of the changes are included in the source code.

5.2 Parameters like dome style (horizontal, vertical), number of render channels and projector and audience position are set in the Lua (ASCII) file *omnimap_startup.lua* in the <*InstallationDir*>*/bin/OmniMapD3DConfig* directory for Direct3D 9 and the <*InstallationDir*>*/bin/OmniMapD3D10Config* directory for Direct3D 10.

5.3 The screen shape parameter that can be set to any of the Elumenati product screen shapes can be found in *omnimap_user_edit.lua*.

5.4 Run the program. Make sure that the directory containing the OmniMap DLLs is in your path. The directory is <*InstallationDir*>*\lib.* Make sure that you set the Configuration property :

Configuration Properties

Debugging

Working Directory

to:

<InstallationDir\bin>