Building OmniMap

This document describes how to checkout, build binaries, and build the installer for the OmniMap library.

SVN Checkout

The OmniMap svn tree resides at http://wush.net/svn/elumenati/OmniMap. The *branch* folder contains three folders that have already been merged back into the *Trunk*. The Tag folder contains snapshots of the tree at various release times. The current release is tagged 2.0.3. If you want to build the current release use that tagged version. If you want to build the most recent version(Which is required for OmniConfig, and Omnity, use the trunk).

To build OmniMap it is advised that you start with an empty folder, and check the OmniMap tree into a folder within that folder. This is because when you build OmniConfig, Omnity, and MoviePlayerTools, you will need to check them out into this same folder, so that relative path references to OmniMap are correct. For instance, to check out the current working version, create a folder called "*Elumenati*". Then use your SVN client to check out the http://wush.net/svn/elumenati/OmniMap/Trunk/OmniMap source into a new folder called OmniMap within the *Elumenati* folder .

Building on Windows

The windows release requires that the following components be built after checkout from SVN:

1. Class library documentation
2. OmniMap.dll release and debug versions
3. OpenGL, D3D 9, and D3D 10 examples

Building Class Library Documentation

First, install *Doxygen* as it is used to build the Programmer's Reference. Once installed start the Doxygen Wizard from the Start menu (Start-programs-doxygen-Doxywizard). Open the file *OmniMap/Documentation/ClassDocs/OmniMap.doxy*. On the front panel, set the ***Working Directory*** to ".", or *OmniMap/Documentation/ClassDocs*, then press the Start button. This will build the documentation. Once the Working Directory has been reset, the documentation can be built by the Visual Studio Solution file if something in a header file changes.

Building the Library

After checking out the OmniMap source tree, using Visual Studio 2005, open the solution file : OmniMap/src/Win32_BuildD3D/omnimapdllD3D.sln. Build the solution, and the OmniMap library is built. The release requires both the debug and release versions be built. So build both of them.

Building the Internal Version of the Library

OmniConfig requires a special build (Internal build) of the *OmniMap* library. To build the internal version(OminMapINT.dll, and OmniMapINTD.dll). If you have recently built the external version of OmniMap, you need to "**Rebuild Solution**" so that all sources get recompiled with the "*ELUMENATI_INTERNAL*" preprocessor variable defined. If the *OmniConfig* source tree is installed per the instructions above, and in the "*Building OmniConfig*" document, OmniConfig will link with the internal library built per these instructions.

Building the Examples

The Windows release includes examples for OpenGL, D3D 9 and D3D 10. To build them, with Visual Studio 2005, open *OmniMap/examples/Win32_Build/OMExamples_2005.sln*. Set the configuration to "Release", and build.

Building Installer for Windows

Using Advanced Installer, open the Advanced Installer project file *OmniMap/Installation/OmniMapAPI.aip*. Build the installer. See the documentation for Advanced Installer for more information. Advanced Installer produces an Windows install file for OmniMap.

Building on OSX

This section describes how to build OmniMap on OSX.

Dependency on libGLEW

OmniMap uses libGLEW, the GL Extension Wrangler library. Install it using macports from http://www.macports.org/

Building the Lua Library

The first step is to build the liblua.a, which is linked with the OmniMap library. In a shell, simply go to :

OmniMap/src/cLua/LuaDist/lua-5.1/src

Then type : make macosx

This will build liblua.a, which is referenced from the xCode build file.

Building the OmniMapAPI Framework

Now, OmniMap can be built using the xCode project file. Open the xCode project file at:

OmniMap/src/OSX_Build/OmniMapAPI.xcodeproj

Select the OmniMapAPI target for build, and build it. It can be built in either Debug or Release mode. In Debug mode, the built framework will be in *OmniMap/src/OSX_Build/build/Debug/OmniMapAPI.framework*. In Release mode, the built framework will be in *OmniMap/src/OSX_Build/OmniMapAPI.dst/Library/Frameworks/OmniMapAPI.framework*. The release tree is used to build an installer for application developers who want to link the OmniMap API with their application.

Building the OmniMapUnityPI Bundle

The current source tree for the plugin wrapper can be checked out from :

http://wush.net/svn/elumenati/OmniMapUnityPI/Trunk/OmniMapUnityPI

Release 1.0.0 for the plugin source is tagged at :

http://wush.net/svn/elumenati/OmniMapUnityPI/Tags/OmniMapUnityPI_1_0_0

The checked out OmniMapUnityPI directory should be in the same directory as the OmniMap source tree root folder(OmniMap). For instance, if you created the “Elumenati” directory when you checked out OmniMap, then OmniMapUnityPI should be a sibling to OmniMap in the Elumenati directory.

The plugin is a bundle that includes a dynamic libarary that is essentially OmniMap plus the OmniMap wrapper code for Unity3D which is in OmniMapUnityPI/src/OmniMapUnitPI.cpp. The plugin bundle is built by simply setting the Active Target in the OmniMapAPI xCode project window to *OmniMapUnityPI*, and the Active Build Configuration to *ReleaseWithOmnity* or *DebugWithOmnity*, and then pressing the Build button. The resulting bundle will be in OmniMap/src/OSX_Build/build/DebugWithOmnity/OmniMapUnityPI.bundle for debug, and OmniMap/src/OSX_Build/build/ReleaseWithOmnity/OmniMapUnityPI.bundle for release. To use this bundle in a Unity3D project, you must copy *OmniMapUnityPI.bundle* into the Unity3D project's *Assets/Plugins* directory.

Building Installer for OSX