Delta3D Simulation Core & Stealth Viewer build process

Version 1.0 -3/11/2014

I.Introduction

This document covers downloading, building, and running the Delta3D Simulation Core & Stealth Viewer suite of applications. These are the modules involved in the process: Delta3D, and SimulationCore.

- Delta3D is an open source simulation gaming engine available from SourceForge.
- SimulationCore contains both the SimulationCore libraries and the StealthViewer. Some art assets and data files that could be released to open source are also included with it; however, in general, you will need to provide your own 3D assets as well as your own HLA configuration files and RTI. SimulationCore is part of delta3d-extras on SourceForge.net.

*** **NOTE** - There is also a 5th repository of material called, DVTE-SimViewer, that is based on the Simulation Core and works with this system. This repository contains optional material that is not publically accessible unless you have access to US GOTS material. It contains the HMMWV driving simulator, night vision goggle support, a large set of art assets (such as 3D models) and configuration data, and road-based dead-reckoning support. If you believe your organization/company/project qualifies to have access to this material, you will need to contact Curtiss Murphy at Alion Science and Technology, cmmurphy@alionscience.com. As part of that request, you will need to have an official government sponsor involved.

The Simulation Core and Stealth Viewer suite is supported on both Linux x86 and Windows XP 32bit. It has also been known to work on Mac OS X and Linux x86-64.

II.Build Environment

All of the sub-projects that make up the Simulation Core and Stealth Viewer are built using CMake. This process typically supports both version 2.4.8 and 2.6.x. CMake is a powerful tool that allows you to define a single build process that can support builds on a wide variety of build environments. For this document, we will focus on Makefiles for Linux and Visual Studio 2005 (ver 8) for Windows. Visual Studio 2003 and 2008 should work, but all external dependencies will need to be rebuilt with the appropriate compiler. CMake is available at (http://www.cmake.org) – Downloads.

III.External Dependencies

The first process for building delta3d is getting the external dependencies.

a. Windows

Go to www.delta3d.org, then downloads, then Delta3D external dependencies. You want the 2.8 package for Visual Studio 2008, 2010, or 2012.

b. Linux

Depending on the distribution used, different methods exist for installing the dependencies. It has a combined build and install script that will generate an ext directory like the one for Win32. For many distributions, such as Fedora and Ubuntu, most of the dependencies are available as prebuilt packages. The rest can be built from the source package. An easy way to do this is to edit the main script and comment out the scripts for the packages that are available on your distribution, then run it normally. The source package is also available from delta3d.org->Downloads->Delta3D External Dependencies.

Some Linux distributions have the environment variable QTDIR set by default. Make sure it is set to point to the 4.8.x version of QT. The value is /usr/lib/qt4 on Fedora 6 through 10.

IV.Checking Out the Source Repositories

The Delta3D 2.8 release includes

Recent Release Branch

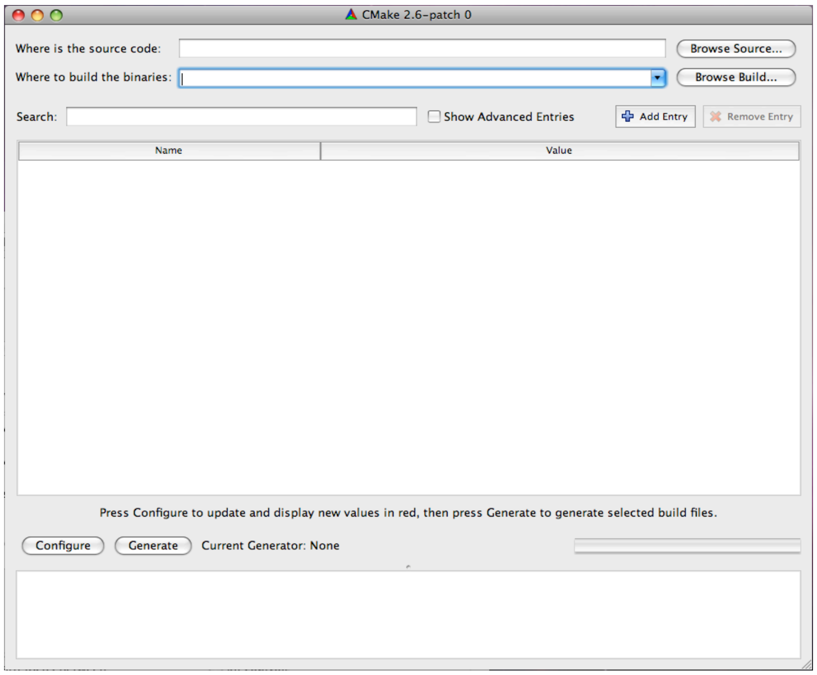
```
https://delta3d.svn.sourceforge.net/svnroot/delta3d/branches/RB-2.8
```

```
https://delta3d-extras.svn.sourceforge.net/svnroot/delta3d-
extras/SimulationCore/branches/RB-2.8
```

V.Building Delta3D

Building Delta3D first requires running CMake. CMake may be run on the command line or as an interactive UI. A `configure` shell script exist to make this process simpler on Linux. If you edit that script to match your settings, you can run it once and be finished. In either the command line or UI case, CMake attempts to figure out the values for variables needed to generate the build system. The variables can be options, library paths, include paths, etc. These values may be set directly on the cmake command line using -Dname=value or in the UI. Here are the specifics for the UI with explanations for command line usage.

1. Run CMake (most likely, this is C:\Program Files\CMake 2.8\bin\cmake-gui.exe or CMakeSetup.exe).



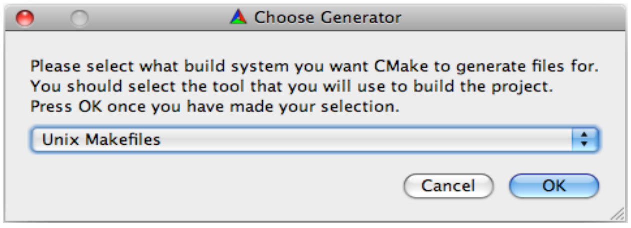
Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

2. Set the source path and the build binary path. Both should be the path to the root of your Delta3D project (ie, something like C:\Projects\Delta3D). CMake can support having the source and binary paths being different, but it is easiest to keep them the same.
3. Once that is done, hit the configure button. It will then ask what generator to use.
   a) Windows

      On Windows, pick Visual Studio and the version you use.
   b) Linux

      On Linux, pick Unix Makefiles. While other generators should work, they are not covered in this document. On the command line, the generator is selected using the -G option, e.g. `cmake . -G "Unix Makefiles"`. "Unix Makefiles" is the default on Linux.



Please select what build system you want CMake to generate files for. You should select the tool that you will use to build the project. Press OK once you have made your selection.

Unix Makefiles

4. After clicking okay, CMake will run a first round of configuration. It will show what paths it found, what it didn't find, and what options are available. Everything will be highlighted in red at this point because the values are not considered final. As part of the configure step, it makes some effort to resolve some of the settings. It will look for a DELTA_ROOT and other settings. Look at each of the settings and paths and verify that they are pointing to the correct directories.
5. In addition to the overall settings, there are a few special settings that you need to consider. Look for the "BUILD_*" options in the UI. Note - if you are using the command line, you may set them with the "-D" option.

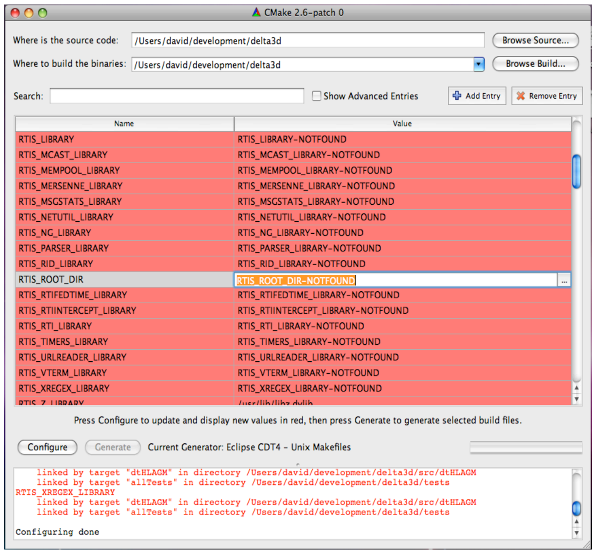
   5.1. BUILD_BINDINGS: Enable this, or set it to ON, if you want to build the python bindings. You don't need these for the Simulation Core or the StealthViewer, and they take a considerable amount of time to compile. We recommend you disable this.

5.2. BUILD_EXAMPLES: Enable this, or set it to ON, if you want to build the delta3d example apps. These take time to build, and are unnecessary, but are helpful for development and also can be run to quickly verify if Delta3D is configured properly.

5.3. BUILD_HLA: Enable this, or set it to ON, if you want to build the HLA components. This is required for the Simulation Core and the Stealth Viewer, but you will need to supply your own RTI. You will have to set a few paths in a future step. Several RTI's are supported in the delta3d build. It should default to RTIS. Options exist to switch to other RTIS. Theses options are USE_RTIS, USE_CERTI, USE_MAK, USE_HLA1516e. You should set only one to ON. On the command line do -DUSE_RTIS=ON, for example. If you want SimCore with no HLA support, you still have to turn on BUILD_HLA, but you can turn off all the plugins. The base dtHLAGM library does not depend on the RTI. It loads it as a plugin.

5.4. BUILD_ISENSE, BUILD_PLIB: These options deal with interfaces to joysticks and trackers. These are not used, so there is no need to enable them unless your custom application needs them.

6. Run configure again.



Where is the source code: /Users/david/development/delta3d

Where to build the binaries: /Users/david/development/delta3d

Search:

Name	Value
RTIS_LIBRARY	RTIS_LIBRARY-NOTFOUND
RTIS_MCAST_LIBRARY	RTIS_MCAST_LIBRARY-NOTFOUND
RTIS_MEMPOOL_LIBRARY	RTIS_MEMPOOL_LIBRARY-NOTFOUND
RTIS_MERSENNE_LIBRARY	RTIS_MERSENNE_LIBRARY-NOTFOUND
RTIS_MSGSTATS_LIBRARY	RTIS_MSGSTATS_LIBRARY-NOTFOUND
RTIS_NETUTIL_LIBRARY	RTIS_NETUTIL_LIBRARY-NOTFOUND
RTIS_NG_LIBRARY	RTIS_NG_LIBRARY-NOTFOUND
RTIS_PARSER_LIBRARY	RTIS_PARSER_LIBRARY-NOTFOUND
RTIS_RID_LIBRARY	RTIS_RID_LIBRARY-NOTFOUND
RTIS_ROOT_DIR	RTIS_ROOT_DIR-NOTFOUND
RTIS_RTIFEDTIME_LIBRARY	RTIS_RTIFEDTIME_LIBRARY-NOTFOUND
RTIS_RTIINTERCEPT_LIBRARY	RTIS_RTIINTERCEPT_LIBRARY-NOTFOUND
RTIS_RTI_LIBRARY	RTIS_RTI_LIBRARY-NOTFOUND
RTIS_TIMERS_LIBRARY	RTIS_TIMERS_LIBRARY-NOTFOUND
RTIS_URLREADER_LIBRARY	RTIS_URLREADER_LIBRARY-NOTFOUND
RTIS_VTERM_LIBRARY	RTIS_VTERM_LIBRARY-NOTFOUND
RTIS_XREGEX_LIBRARY	RTIS_XREGEX_LIBRARY-NOTFOUND
RTIS_Z_LIBRARY	/usr/lib/libz.dylib

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Eclipse CDT4 - Unix Makefiles

```
    linked by target "dtHLAGM" in directory /Users/david/development/delta3d/src/dtHLAGM
    linked by target "allTests" in directory /Users/david/development/delta3d/tests
RTIS_XREGEX_LIBRARY
    linked by target "dtHLAGM" in directory /Users/david/development/delta3d/src/dtHLAGM
    linked by target "allTests" in directory /Users/david/development/delta3d/tests

Configuring done
```

7. Once configure finishes, you will see new variables for RTIS build options. If you have RTI and RTI_HOME set in your environment variables, it will probably find your RTI files. If these values are not correct, try setting the RTIS_ROOT_DIR option in the UI and press Configure again. Note – if you are using the command line, simply add -DRTIS_ROOT_DIR=YourPathToRTI. Run configure again.

8. Hopefully, you now have no red values and no errors. Hit "Generate" or ‘OK’.

9. Build Delta3D

   a) Windows

      On Windows, a “Delta3D.sln” file should exist in the root of the delta3d directory. Open your Visual Studio in the normal way and then open the solution file and build it.

   b) Linux

On Linux, you should have a Makefile in the root. Running make (or make -j[x] where [x] is the number of concurrent threads), will build Delta3D. The default command line in the Linux `configure` script is:

```
cmake -debug-output . -DBUILD_HLA:BOOL=ON -DUSE_RTIS=ON -
DBULID_PLIB:BOOL=ON -DBULID_BINDINGS:BOOL=ON -
DBULID_ISENSE:BOOL=ON -DCMAKE_BUILD_TYPE=Release
```

You should change it to

```
cmake -debug-output . -DBUILD_HLA:BOOL=ON -DUSE_RTIS=ON -
DBULID_PLIB:BOOL=OFF -DBULID_BINDINGS:BOOL=OFF -
DBULID_ISENSE:BOOL=OFF -DCMAKE_BUILD_TYPE=Release -
DRTIS_ROOT_DIR=[Your rti-s path]
```

If all the dependencies are setup correctly, this command line should run and build the Makefiles with no errors.

Note, If you need to support an RTI other than one that is supported, you can it by looking at the ones in the CMakeModules folder and make your own. If you add support for another RTI and would like to submit it, send the changed files and instructions to info@delta3d.org. The same is true for any Delta3D submissions. SimulationCore and the optional DVTE-SimViewer projects will have to updated to allow building the other RTI versions.

VI. Building SimulationCore

Use CMake on SimulationCore the same way you did for Delta3D

SimulationCore looks for Delta3D in "../delta3d". If you have not put it there, you will need to set the DELTA_DIR variable in the UI or with the "-DDELTA_DIR=[my path]" command line option.
LWindows

On Windows, a "SimulationCore.sln" file should exist in the root of the SimulationCore directory. Open your Visual Studio in the normal way and then open the solution file and build it.

a) Linux

Assuming you have Delta3D

```
cmake -debug-output . -DCMAKE_BUILD_TYPE=Release
```

VII.Running Delta3D Applications

Once CMake is run on all the repositories, and built using CMake or Visual Studio, applications still will not run because all of the compiled libraries will need to be added to the library path.

To run the demos in SimCore, you need to make your current working directory the root dir of the demo project in the source tree. It will have a config.xml file. You may need to edit this file to give the proper path to the Pal plugins. They are typically in the delta3d/ext/PalPlugins directory.

a) Windows

The accepted solution on Windows has been to run Visual Studio from a batch file so that the path will be setup, but you also set the environment variables system-wide. Either way, you need to set DELTA_ROOT, DELTA_DATA, and PATH. Here is an example DEV environment variable with all the projects installed there:

```
set
PATH=%DEV%\SimulationCore\bin;%DEV%\delta3d\bin;%DEV%\delta3d\ext\bin;%DEV%\delta3d\ext\bin\osgPlugins;%DEV%\rtis20\bin\win32_msvc-10.0;C:\Qt\4.8.5\bin

export DELTA_ROOT=%DEV%\delta3d

export DELTA_DATA=%DELTA_ROOT%\data
```

b) Linux

On Linux LD_LIBRARY_PATH needs to be set, as well as DELTA_ROOT and DELTA_DATA for the Delta3D examples to work. Assuming the DEV environment variable is set, here is an example

```
export
LD_LIBRARY_PATH=${DEV}/SimulationCore/lib:${DEV}/delta3d/lib:${DEV}/delta3d/ext/lib:${DEV}/delta3d/ext/lib/osgPlugins:${DEV}/rtis20/lib/linux_g++-4.4

export DELTA_ROOT=${DEV}/delta3d

export DELTA_DATA=${DELTA_ROOT}/data
```