OpenEXR x64 : Windows Build
How to build and link OpenEXR x64 with Gratin on Windows

Bourgeas Jean

INRIA

July 15, 2015

Sommaire

1. First of all, you need to download visual sudio 2010 SP1 if you don't have it (it is free).
2. find the zlib125dll.zip and zlib125.zip (this is an old version and isn't on the zlib website anymore).
3. find ilmbase-1.0.2.tar.gz and openexr-1.7.0.tar.gz on the OpenEXR website.
4. Let's assume we're in yourpath\ locaction. Create a Folder and extract zlib125.zip, ilmbase and openexr in it. Let's call it "Files".
5. Extract zlib125dll.zip in yourpath\Files\zlib\

1. Go to yourpath\Files\ilmbase-1.0.2\vc\vc8\IlmBase\ and launch IlmBase.sln with Visual Studio 2010.
2. In BUILD→Configuration Manager, change Active solution platform to x64 and Active solution configuration to Release.
3. Press F7 to build the solution.

1. Copy yourpath\Files\zlib\dllx64\zlibwapi.dll in yourpath\Deploy\bin\x64\Release\
2. Copy yourpath\Files\zlib\dllx64\zlibwapi.lib in yourpath\Deploy\lib\x64\Release\
3. Copy yourpath\Files\zlib-1.2.5\zlib.h in yourpath\Deploy\include\
4. Copy yourpath\Files\zlib-1.2.5\zconf.h in yourpath\Deploy\include\

1. Go to yourpath\Files\openexr-1.7.0\vc\vc8\OpenEXR\ and launch OpenEXR.sln with Visual Studio 2010.
2. In BUILD→Configuration Manager, change Active solution platform to x64 and Active solution configuration to Release.
3. Press F7 to build the solution.

1. Now, when you will compile Gratin, instead of typing : cmake -DEIGEN3_INCLUDE_DIR="path_to_eigen"
2. You will have to type (let's say libpath = yourpath\Deploy\lib\x64\Release) : cmake -DEIGEN3_INCLUDE_DIR="path_to_eigen" -DZLIB_INCLUDE_DIR="yourpath\Deploy\include" -DOPENEXR_INCLUDE_PATH="yourpath\Deploy\include" -DZLIB_LIBRARY="libpath\zlibwapi.lib" -DOPENEXR_IMATH_LIBRARY="libpath\imath.lib" -DOPENEXR_HALF_LIBRARY="libpath\Half.lib" -DOPENEXR_IEX_LIBRARY="libpath\iex.lib" -DOPENEXR_ILMIMF_LIBRARY="libpath\IlmImf.lib" -DOPENEXR_ILMTHREAD_LIBRARY="libpath\IlmThread.lib"
3. Do not forgot to copy all the dlls from yourpath\Deploy\bin\x64\Release\ in your Gratin build directory or it will fail at the execution

This was the easy way to use OpenEXR on Windows and I really hope that this version is still compatible.
If it is not, we found an hard way to make the last version of OpenEXR works on our machines.

Sommaire

1. First of all, you need to download visual sudio 2013 if you don't have it (the community version is free). We tested this manipulation with the 2008/2010/2012 version and it didn't work.
2. Go get zlib on this page.
3. We made this work for the 1.2.8 version and hardly recommend to do the same.
4. You need to get the source code version and the compiled dll version.
5. Let's assume we're in yourpath\ locaction. extract zlib128.zip here and zlib128-dll.zip in yourpath\zlib\.

1. Go to yourpath\zlib-1.2.8\contrib\vstudio\vc11 and open zlibvc.sln with visual studio 2013.
2. Accept the security warnings and the upgrade VC++ Compiler and Libraries.
3. In BUILD→Configuration Manager, change Active solution platform to x64 and Active solution configuration to Release.
4. In the solution Explorer go to zlibvc→zlibVC.def and change VERSION 1.2.8 to VERSION 1.28
5. You can now build by pressing F7.
6. You now have zlibwapi.dll and zlibwapi.lib in yourpath\zlib-1.2.8\contrib\vstudio\vc11\x64\ZlibDllRelease.

1. copy and rename zlibwapi.dll in zlib1.dll and zlibwapi.lib in zdll.lib
2. replace yourpath\zlib\zlib1.dll by the new one.
3. replace yourpath\zlib\lib\zdll.lib by the new one.
4. this manipulation will remove these zlib errors : (in case you're encountering them)
   error LNK2019: unresolved external symbol compress
   error LNK2019: unresolved external symbol compress2
   error LNK2019: unresolved external symbol uncompress

Sommaire

1. First of all, you need to download visual sudio 2010 SP1 if you don't have it (it is free). We tested this manipulation with the 2008/2010/2013 version and it didn't work.
2. Go get ilmbase on this page.
3. We made it work with the 2.2.0 version.
4. extract ilmbase-2.2.0.tar.gz in yourpath\

1. Open a command prompt : press windows + R, write cmd and press OK.
2. go to yourpath\ilmbase-2.2.0 and enter the following instructions :
3. setlocal
4. cmake -DCMAKE_INSTALL_PREFIX=..\build -G “Visual Studio 10 Win64” ..\ilmbase-2.2.0

- Go to yourpath\ilmbase-2.2.0 and open ilmbase.sln with Visual Studio 2010.
- In BUILD→Configuration Manager, change Active solution platform to x64 and Active solution configuration to Release.
- Press F7 to build the solution.
- Right click on the INSTALL project and choose Build.

- Go to yourpath\build\lib and copy the five dll files.
- Paste them where cmd.exe is installed on your machine (probably in C:\Windows\System32\).
- this manipulation will remove this error : (in case you're encountering it)
  error MSB6006: "cmd.exe" exited with code -1073741515.

Sommaire

1. Go get openexr on this page.
2. We made it work with the 2.2.0 version.
3. extract openexr-2.2.0.tar.gz in yourpath\

1. Open a command prompt : press windows + R, write cmd and press OK.
2. go to yourpath\openexr-2.2.0 and enter the following instructions :
3. setlocal
4. cmake -DZLIB_ROOT=..\zlib -DILMBASE_PACKAGE_PREFIX=..\build -DCMAKE_INSTALL_PREFIX=..\build -G "Visual Studio 10 Win64" ^ ..\openexr-2.2.0

- Go to yourpath\openexr-2.2.0 and open openexr.sln with Visual Studio 2010.
- In BUILD→Configuration Manager, change Active solution platform to x64 and Active solution configuration to Release.
- from the b44ExpLogTable to IlmImfUtilTest project, you will have to resolve linker errors. For each, right click on it and choose Properties. Go to Configuration_Properties→Linker→General and, in the Additional Library Directories, change the two occurencies of ../build in ../../build
- Press F7 to build the solution.
- Right click on the INSTALL project and choose Build.