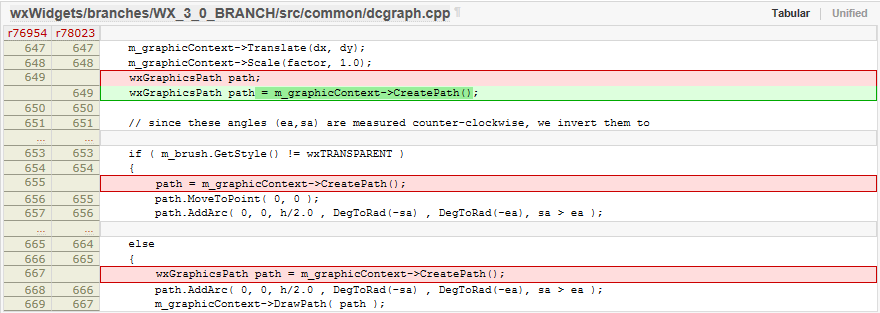
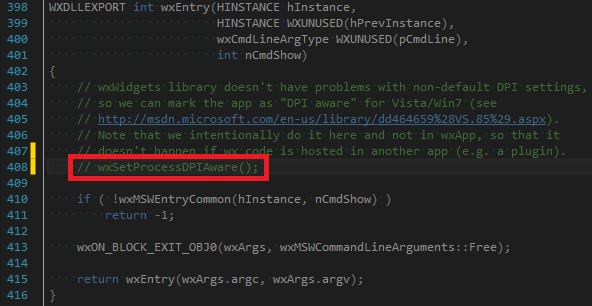
SAM Development with Visual Studio 2013 and wxWidgets 3.0.2

Aron P. Dobos 8 December 2014

1. Download Visual Studio Express 2013 for Windows Desktop from <http://www.visualstudio.com/en-US/products/visual-studio-express-vs>. Usually it is more reliable to download the .iso disk image, and then use a utility like WinCDEmu to mount the .iso image to a virtual drive. You can download WinCDEmu from <http://wincdemu.sysprogs.org/>
2. Install VS 2013. Run VS 2013 and select Help->About->Product license information. Sign in or create a Microsoft.com account to obtain a free license.
3. Download wxWidgets 3.0.2 (wxWidgets-3.0.2.zip) from [http://sourceforge.net/projects/wxwindows/files/3.0.2/wxWidgets-3.0.2.zip/download](http://sourceforge.net/projects/wxwindows/files/3.0.1/wxWidgets-3.0.1.zip/download)
4. Extract to c:\wxWidgets-3.0.2
5. Start VS 2013, and open c:\wxWidgets-3.0.2\build\msw\wx\_vc12.sln
6. Implement a fix for the wxGraphicsPath issue. See related discussion at <http://trac.wxwidgets.org/changeset/78023>
   1. Open c:\wxWidgets-3.0.2\src\common\dcgraph.cpp
   2. Delete lines 655 and 667, and initialize the ‘path’ variable on line 649, according to the image below:



1. Disable process DPI awareness for applications so that they are virtualized on high-DPI screens. See related discussion at <http://trac.wxwidgets.org/ticket/16116>
   1. Open C:\wxWidgets-3.0.2\src\msw\main.cpp
   2. Comment out line 408 that calls wxSetProcessDPIAware().



1. Build Debug and Release configurations for both Win32 and x64 platforms. Note: wxWidgets 3.0.2 does not require a separate folder for the 64-bit build anymore. The libraries for 32-bit go into lib\vc\_lib and 64-bit into lib\vc\_x64\_lib.
2. Setup the WXMSW3 environment variable to point to c:\wxWidgets-3.0.2 (Control Panel->System->Advanced system settings->Environment variables). Also, delete your old WXMSW3x64 environment variable if you have it – it’s no longer needed.
3. Check out the LK script engine from <https://efmsvn.nrel.gov/lk/svn/trunk> There’s a folder called vc2013\_wx3 with VS 2013 project files. Build the Debug/Release configurations for both win32 and x64. If successful, the libraries lkvc13wx3.lib, lkvc13wx3d.lib, lkvc13wx3x64.lib, and lkvc13wx3x64d.lib will be created in the “lk” folder. Create a new environment variable called LKDIR that points to your local LK folder.
4. Check out the wxWidgets Extensions (wex) library from <https://efmsvn.nrel.gov/wex/svn/trunk>. There’s a folder called vc2013\_wx3 with VS 2013 project files. Build the Debug/Release configurations for both win32 and x64. If successful, the libraries wexvc13wx3.lib, wexvc13wx3d.lib, wexvc13wx3x64.lib, and wexvc13wx3x64d.lib will be created in the wex folder, and dview{x64}.exe and sandbox{x64}.exe will exist in the project solution folder. Both of the executables should run fine. Create a new environment variable called WEXDIR that points to your local WEX folder.
5. Check out SAMnt from <https://efmsvn.nrel.gov/SAMnt/svn/trunk>. Before compiling, create a new environment variable SAMNTDIR that points to the local SAMnt folder.
6. Check out the SSC SDK from <https://efmsvn.nrel.gov/ssc/svn/trunk>. Use the build\_vc2013\ssc\_vc2013.sln project file and build Debug/Release for both win32/x64. After compiling, the project file automatically copies the Release ssc.dll to the SAMNTDIR/deploy/win32 or x64 folder, so that SAM always is using the latest simulation engine library. Create a new environment variable SSCDIR that points to the local SSC folder.
7. Open the SAMNTDIR/vc2013\_wx3/SAMnt\_vc2013.sln project file, and build both Debug/Release and win32/x64. The sam.exe binaries should appear in SAMNTDIR/deploy/win32/ and SAMNTDIR/deploy/x64.
8. The SAMnt SVN contains precompiled binaries for libcurl which SAM uses to access the internet. Simply copy everything in SAMNTDIR\vc2013\_wx3\libcurl\_ssl\_win32\bin to SAMNTDIR\deploy\win32, and similarly for the x64 folders.

SAM should now run properly!