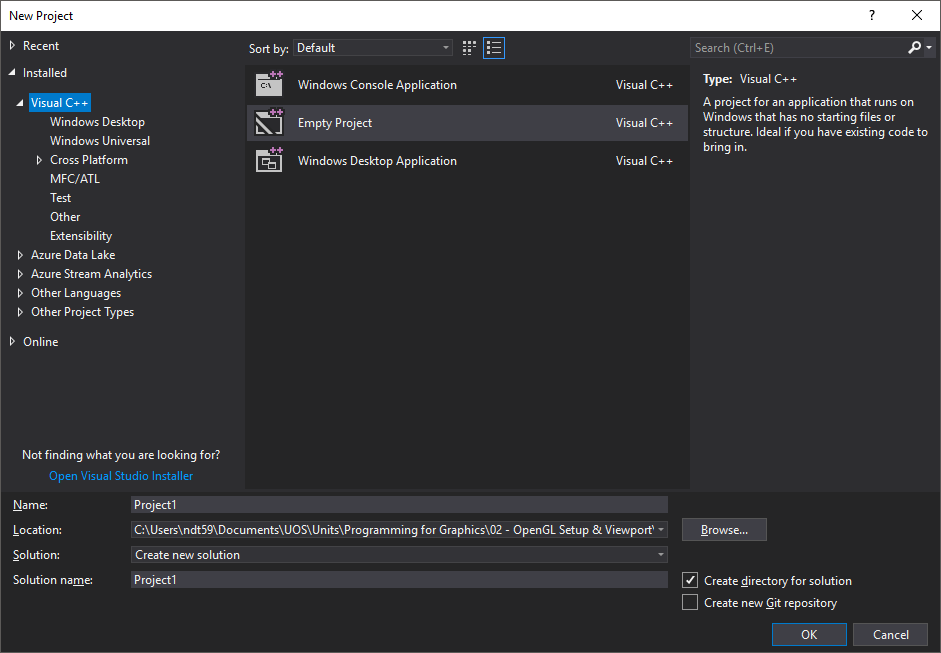
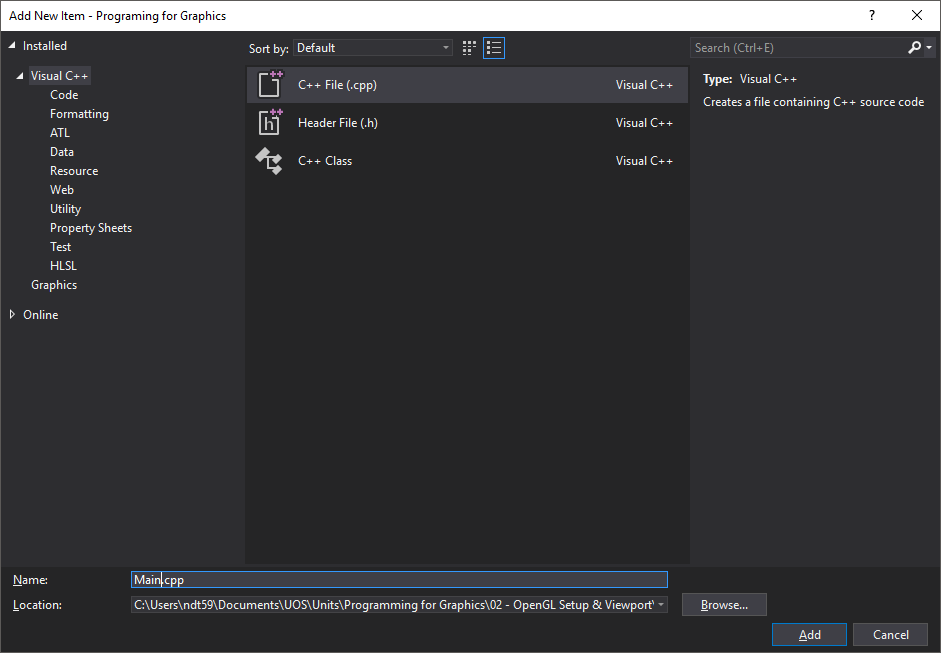
Be sure to down load the required library’s

* SDL: <https://www.libsdl.org/download-2.0.php>
  + Download the development libraries
  + (SDL2-devel-2.0.X-VC.zip)
* GLFW: <http://glew.sourceforge.net/>
  + Binaries for Windows 32-bit and 64-bit
* GLM: <https://glm.g-truc.net/0.9.9/index.html>

Create a new, empty c++ project.

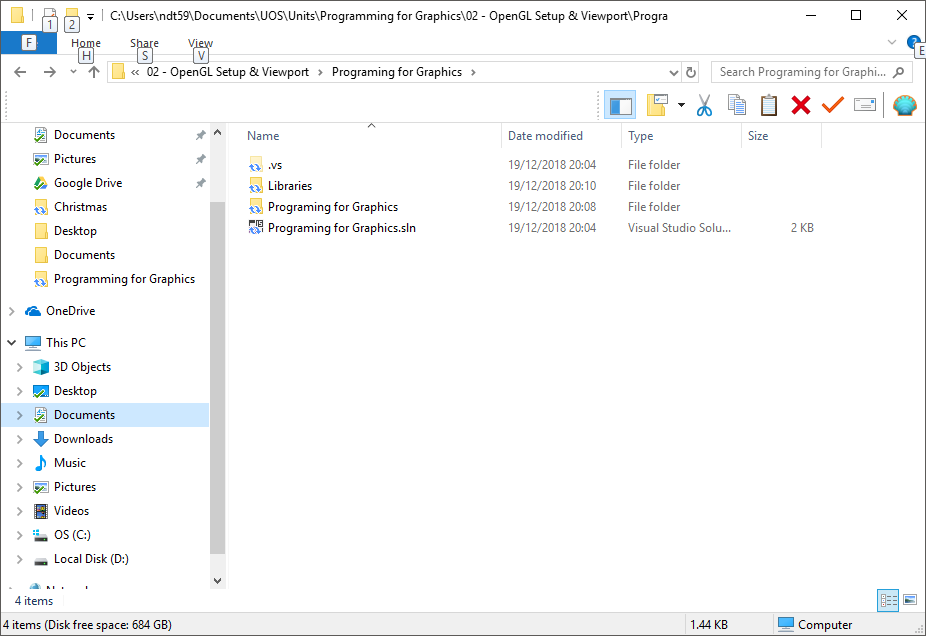


Add a new .cpp file called Main

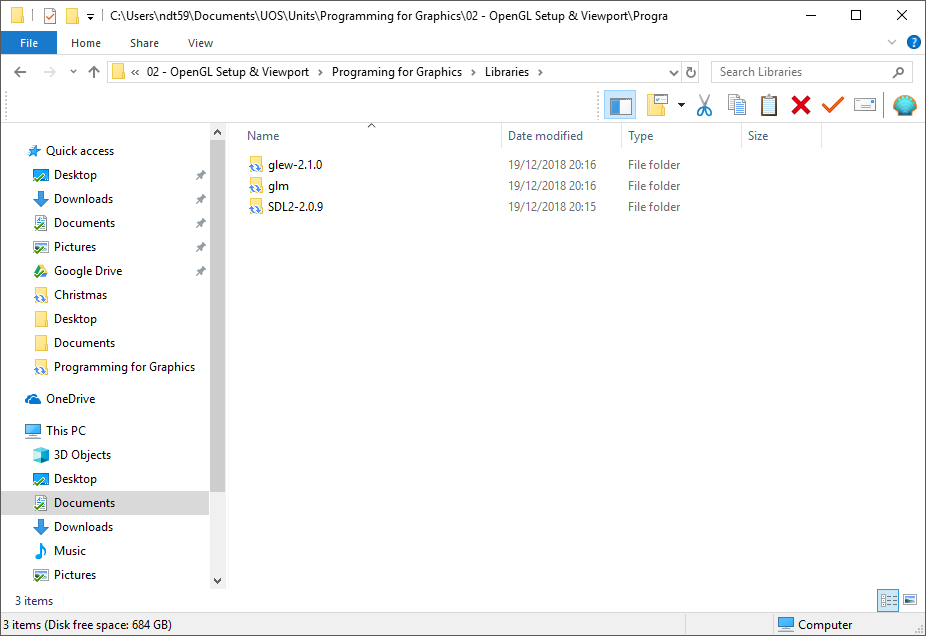


Open a file explorer window and navigate to you project.

Create folder in the project called Libraries.

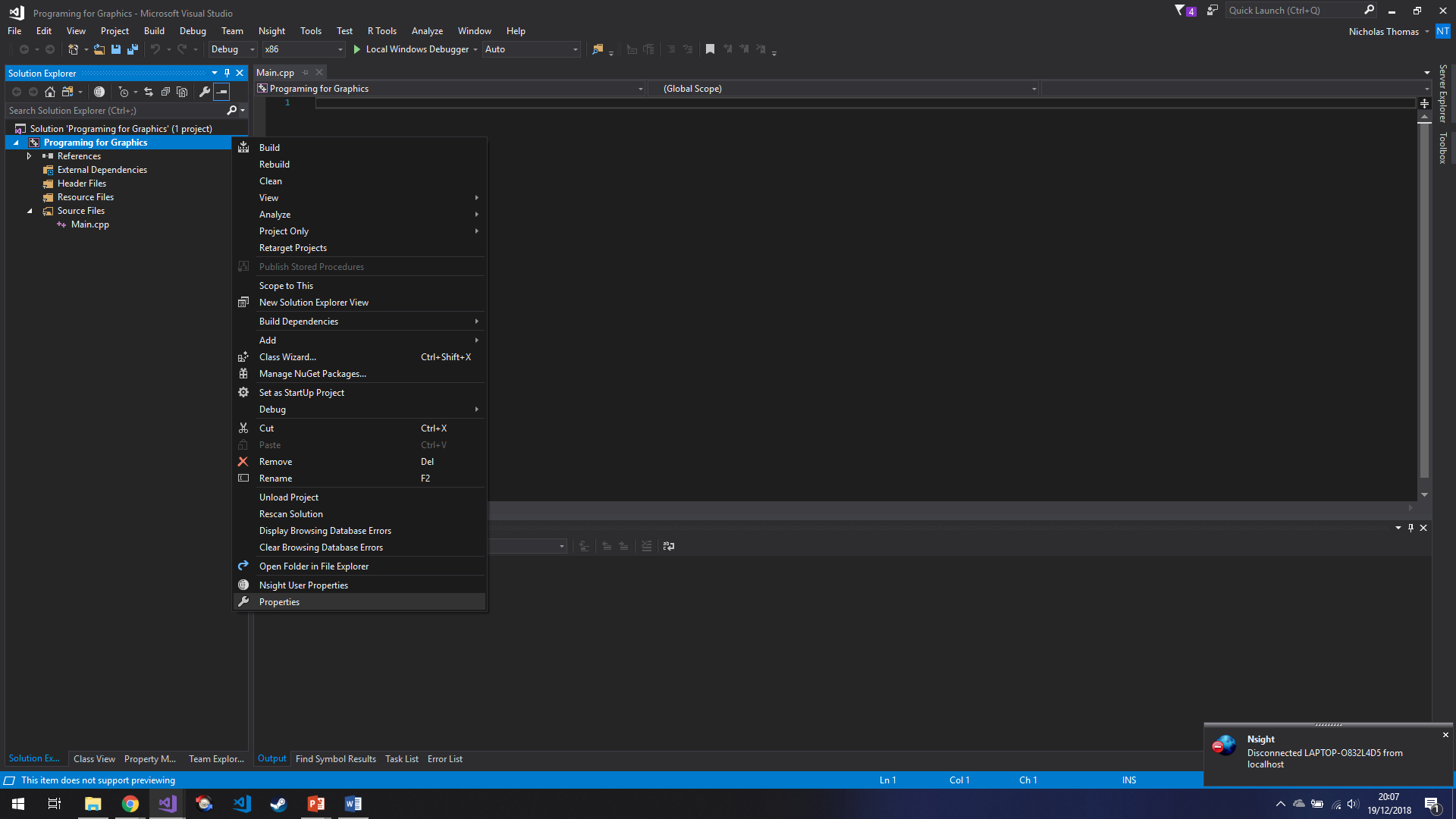


Unzip all the libraries, listed above, in to that folder.

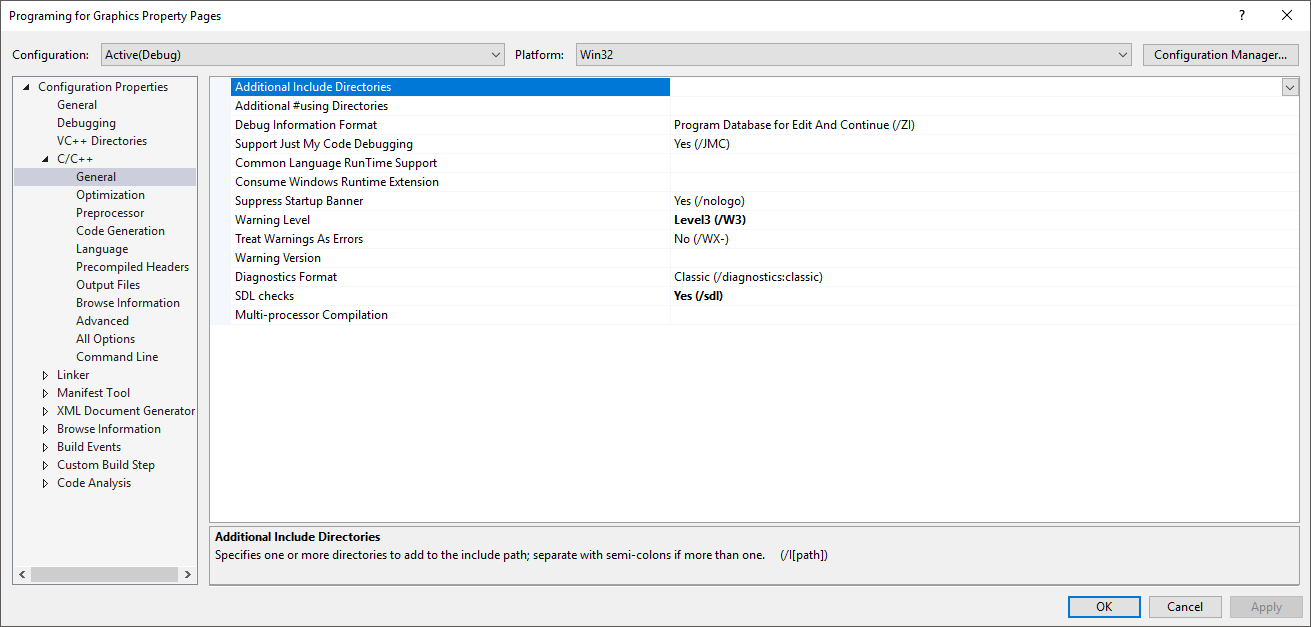


Back in to visual studio.

Right click on the project and select properties



Under C/C++, General, click on the drop-down menu and chose edit



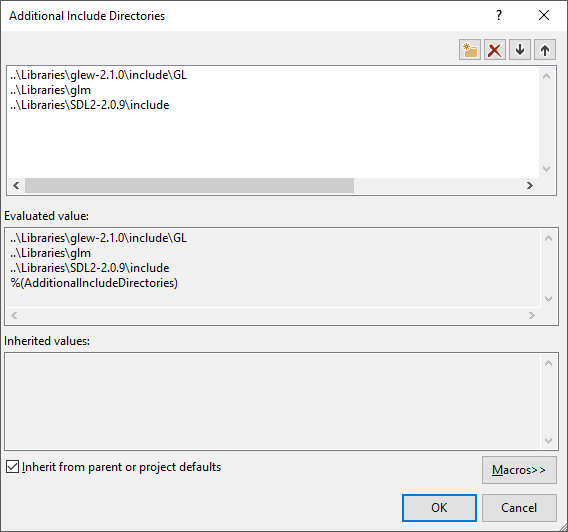
In this window we will add all of the include folders from the libraries we just downloaded.

Assuming your project looks like mine, we can use relative referencing to locate our include files. Add the following to the menu. (the “..\” is an instruction to theOS to go up one folder level before looking for the “Libraries” folder).

..\Libraries\SDL2-2.0.9\include

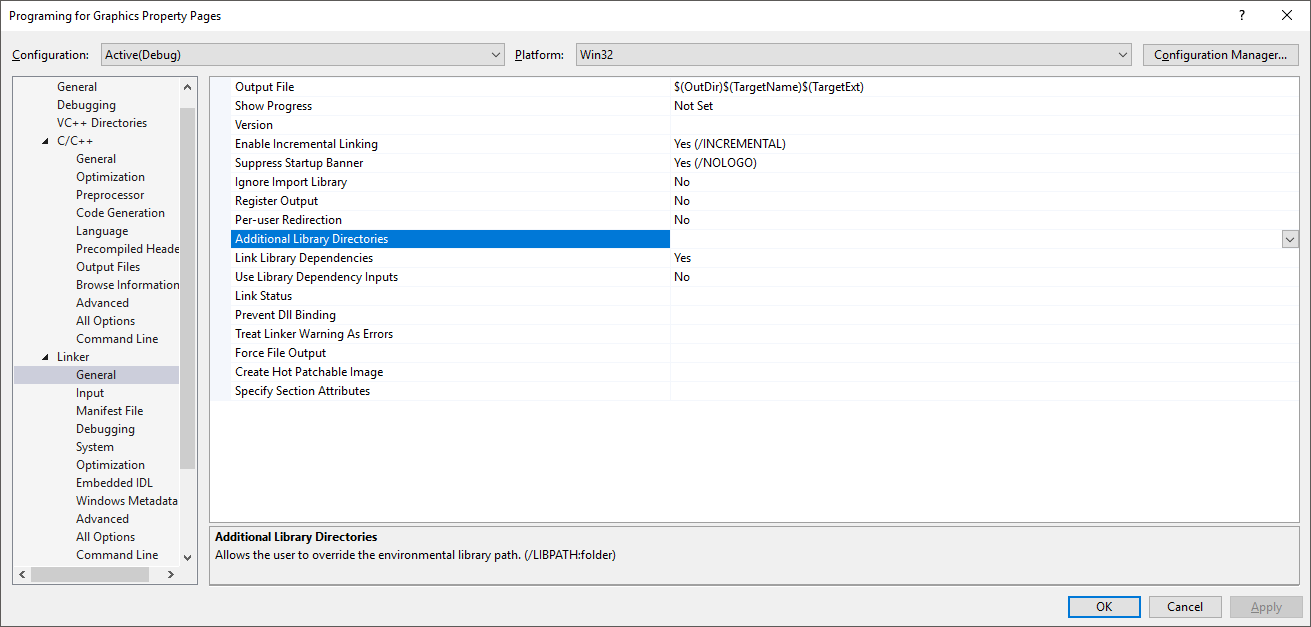
..\Libraries\glm

..\Libraries\glew-2.1.0\include\GL



Click ok

Now navigate to linker/general and select the dropdown box for “Additional Library directories”

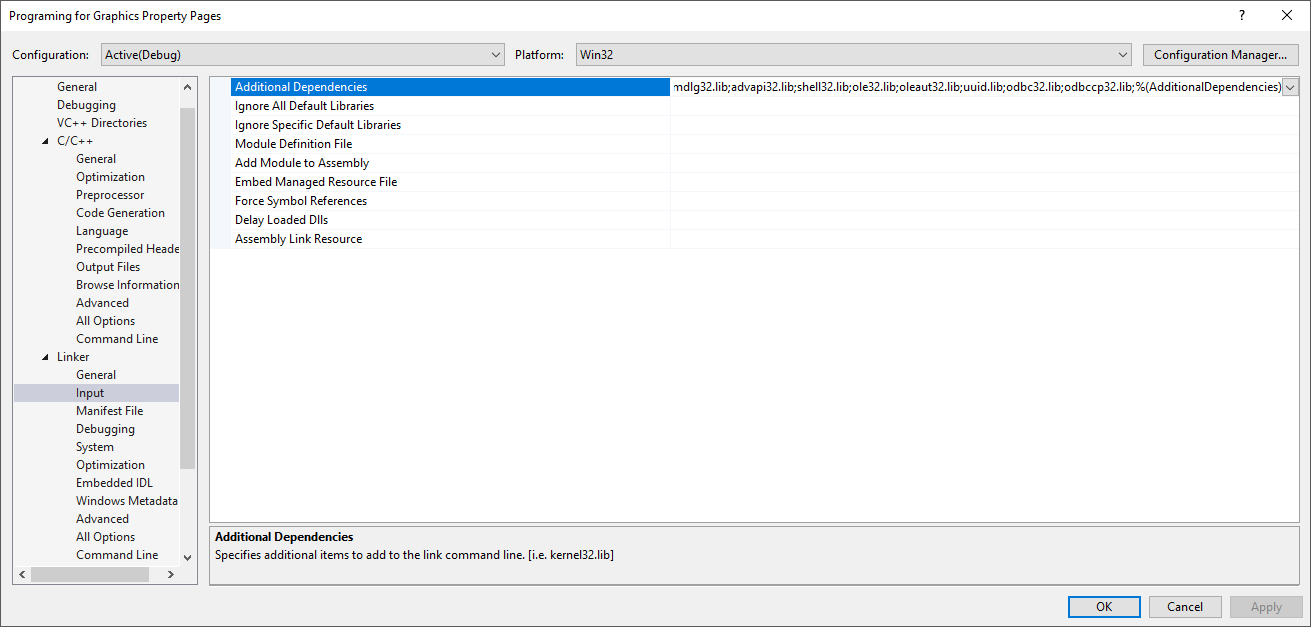


And add the following (again, assuming you project is set up link mine).

$(ProjectDir)\Libraries\SDL2-2.0.9\lib\x86

$(ProjectDir)\Libraries\glew-2.1.0\lib\Release\Win32

Now move to Linker/input and click the drop down in “additional dependencies”.



In the new window, add the following

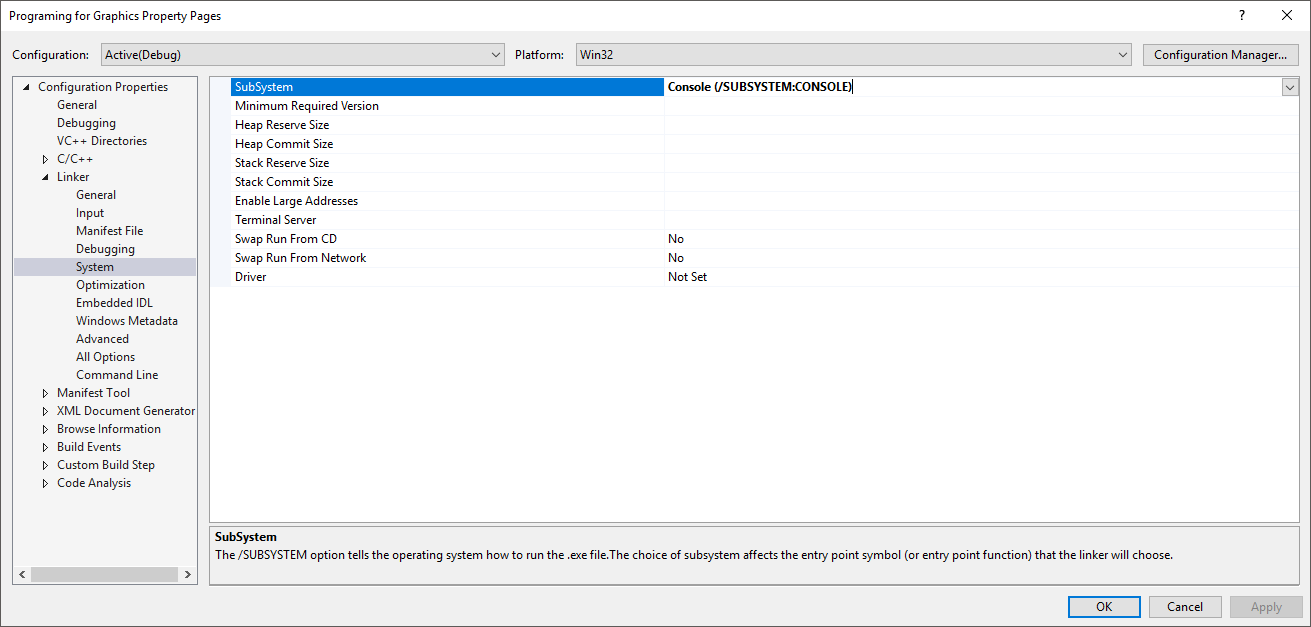
SDL2.lib

SDL2main.lib

glew32s.lib

opengl32.lib

now navigate to Linker/System/Sub system and ensure “Console (/SUBSYSTEM:CONSOLE)” is selected.

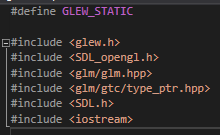


click ok

click apply and close the window.

Now we need to test our setup is correct.

We’ll start by opening main.cpp and adding the following code.



Assuming that everything is setup correctly, none of these includes should be underlines in red. If they are that mean visual studio cant fine the file. Normally this is because the path specified for the additional include, is not correct. Go back a few steps and see if you can solve problem.

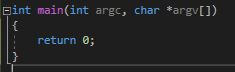
Otherwise lets move on.

The first line defines a pre-processor command, GLEW\_STATIC. This command is an instruction to GLEW that we want to use GLEW directly in our executable, no lib or DLL files will be used.

The 6 lines are just regular include commands. The only thing to note here is that you MUST include glew.h before anything else, elements of openGL depend on it and your compiler will generate an error if you try to include it after openGL. To be safe, always include GLEW first.

Don’t forget to add the “using namespace std”.

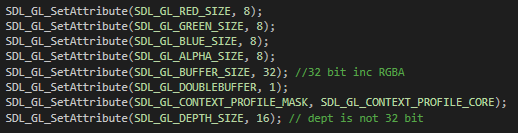
And define you main function. I’m just going to make mine return 0 for now, to keep the compiler quiet.



Now we’re going to initialize SDL and we’re going to use the SDL\_INIT\_EVERYTHING command because its easier that cherry picking the stuff we need, so we’ll just take it all and if we don’t use parts of SDL, oh well.



Next we need to tell SDL how we want to set up openGL for us



The first 3 calls to SDL\_GL\_SetAttribute tell openGL how many bits to use for our red, green and blue channel data. This give us 256 possible values for each. The 4th call sets the number of bits for the alpha channel. Last session, when I said, “we will be using RGB,” I lied we actually using RGBA.

A, Alpha is the level of transparency for a given pixel. We will be using the Alpha channel for textures in later sessions.

The 5th call tells openGL that we are using 32 bit colour values. RGBA is 4 \*8 bits, so 32 and openGL needs to allocate buffers for us, this the correct size, or we will end up with buffers that are too big (and filled with extra space we will not use) or worse is buffers that are too small (and we’ll end up writing over data of other pixels).

The 6th call simple tells openGL that we want double buffering, this will be explained in later lectures.

The 7th call tell openGL that we want the want the core functionality of modern openGL and that deprecated functions are disabled.

The 8th call tell openGL that we want a depth buffer per-pixel that is 16 bits large, this will be used later for shadows.

Now we need to create a window and a context for openGL to render too.



The first thing we do is create a window pointer and set it using the SDL\_CreateWindow function. This will do all the heavy lifting of window creation for us. The first perimeter is the name that will appear in the top of the window, the second 2 parameters define where the window will be located on start-up. The next 2 are the width and height of the window. And the last one is actually 2 bit fields, or’d together to tell the window what we want to be able to do. We want it to be resizable and we want it to work with openGL.

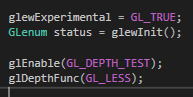
Next, we want to initialize GLEW but before we do this, we want to tell glew that we want some of its experimental features, they might come in handy in later tutorials. glewExperimental is a global in GLEW and we use GL\_TRUE to set it. Why GL\_TURE? Because openGL runs on multiple platforms and it isn’t guaranteed that TURE on one platform will be equal to true on another platform. So, some crafty developers decided to have a big list of variables, beginning with GL\_ that will be redefines at compile time, to the appropriate value for that platform.

What dose this mean for us?

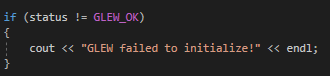
Not much, just know that GL\_TRUE is just TURE by a different name and if you see GL\_INT or GL\_FLOAT its just a platform agnostic way to define variable for openGL. If you use a float instead of a GL\_FLOAT, no Biggy, it should still work.

The next line tells open GL that we want to enable depth testing. We’ll cover this in later lectures, for now just add it and we’ll come back to it.

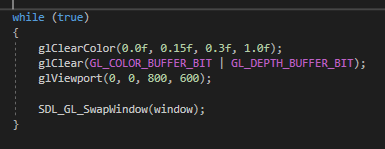
The last line just specifies how openGL will perform the depth test. Again, this will be covered later.



Next, we are going to be safe and check that GLEW was initialized correctly.



Now we’re going to add an innate while loop, after all, we don’t want to see our window for 1 microsecond and then see it disappear when the program ends

.

We will call glClearColor and pass 4 floats to it. This function sets the background of the openGL window, the 4 floats are representations for our RGBA values.

We use floats in the range 0.0 to 1.0 to represent no colour value and all the colour value.

0.0 is 0 for an 8 bit number and 1.0 is 255 for an 8 bit number (0.5 would be 128).

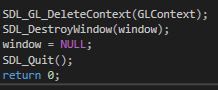
Since we can change how many bits our colour channels have, the openGL developers could not be sure how many bits this glClearColor should take per-perimeter. So, they elected to use floats instead and convert to the appropriate range with in the function.

glClear tells openGL to clear the screen now, with the colour we defined above. One parameter but we are or’ing 2 bit flags together, again, to tell openGL that we want to clear the colour information and the depth information per-pixel.

Next, we set the view port with glViewport and we pass two 0’s, these 0’s tell the window how far to offset the openGL viewport in the window. We want to offset in ether x or y so we pass 0. Then we tell openGL how big we want the viewport to be, 800 by 600 to match our window.

And finally, we call DL\_GL\_SwapWindow and pass in our pointer to window. This is and SDL function to tell openGL that we have finished rendering this frame and we are ready to present the results to the window we have specified.

For completeness, even thought we will never get here due to the infinite loop, we will clean up our objects and make sure we don’t leave any pointers dangling, eating up users memory.



Now run your program and!...

You should get an error stating that the program can’t find SDL2.dll or GLFW.dll.

This is normal. Close the program and open the project in an explorer window.

Next to the Libraries folder you should see a new folder called Debug, open it and you should see several files, one of which should be a .exe.

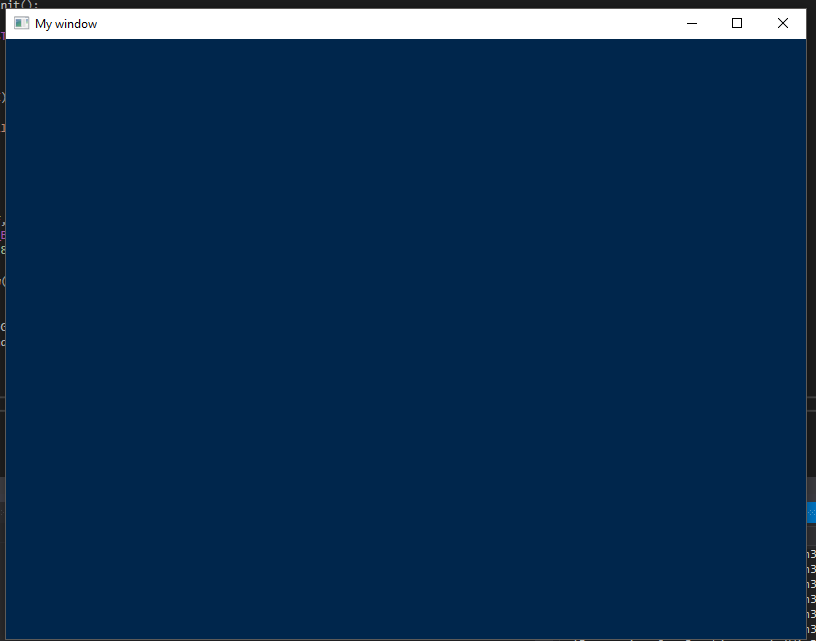
Leave the window open and in an new window, navigate to:

\Libraries\SDL2-2.0.9\lib\x86 and COPY the SDl2.dll file in to the Debug folder.

Now navigate to:

\Libraries\glew-2.1.0\bin\Release\Win32 and copy the glew32.dll into the debug folder.

Now run your code again and you should see something like the following.



Congratulations!

You have a window ready for openGL.

Extras

Using SDL\_Delay I want you to change the colour of the background so it changes from blue to read and back again.