

Installing OpenGL and Running the Code

Our programs were all developed over the past few years on a Microsoft Visual Studio Community (MSVS) 2019 IDE running on Windows 10 on a 64-bit PC with a graphics card supporting OpenGL 4.3

Described below is how to set the environment up on our development platform of MSVS Community 2019/64-bit Windows 10. Keep in mind that, though a graphics card supporting at least OpenGL 4.3 is ideal, a lower version of OpenGL might be fine too. These instructions should be applicable to other versions of Windows and Visual Studio and to a 32-bit machine as well, possibly with minor adjustments and the occasional tweak to the code itself; our programs, again possibly with minor changes, should all run on a Linux or Mac OS environment.

Install Visual Studio: Download Microsoft Visual Studio Community 2019

Install Helper Libraries

GLFW, GLUT, SDL, SFML etc.. are libraries used to create and manage OpenGL contexts and windows. You need these libraries before you can use OpenGL. Some of these libraries provide support for cross platform input and sound stuff as well.

GLEW, GLAD, gLoadgen etc.. are OpenGL extension loaders. Since OpenGL support is implementation specific and constantly updating, these loaders query which OpenGL specs and extensions are supported by the drivers installed, or which you want to use, and provide you a way to access that OpenGL functionality.

Create a folder called OpenGLwrappers in the C: drive, so this folder is C:\OpenGLwrappers. This particular name and location is so that all our programs in ExperimenterSource run out of the box. If for some reason, e.g., not having access to the root drive, you can't place OpenGLwrappers as asked, then place it where you can. You will have to change our project properties accordingly and we'll say further on how to do this.

The version numbers below of the libraries we ask you to download may not be the latest but they are the ones we have installed in our environment. There are sometimes niggling little things that differ with versions. To be sure, we have not yet heard of a problem in upgrading to a newer version that could not be fixed without too much trouble. Still, to avoid any issues at all while you are learning we suggest you install the exact versions below.

1. FreeGLUT 32 bit: Download and unzip <http://files.transmissionzero.co.uk/software/development/GLUT/older/freeglut-MSVC-2.8.1-1.mp.zip> and save the folder freeglut-MSVC-2.8.1-1.mp in OpenGLwrappers.
2. FreeGLUT 64 bit: Download and unzip freeglut-MSVC-3.0.0-2.mp from <https://www.transmissionzero.co.uk/software/freeglut-devel/>
3. GLEW 32 bit: Download and unzip the file <https://sourceforge.net/projects/glew/files/glew/1.10.0/glew-1.10.0-win32.zip/download> and save the folder glew-1.10.0-win32 in OpenGLwrappers.
4. GLEW 64 bit: Download <https://sourceforge.net/projects/glew/> and unzip glew-2.1.0-win32 and save it under OpenGLwrappers.
5. GLM: Download and unzip the file <https://github.com/g-truc/glm/releases/download/0.9.7.5/glm-0.9.7.5.zip> and save the folder glm-0.9.7.5 under OpenGLwrappers.


6. GLFW 32 bit: Download <https://github.com/glwf/glwf/releases/download/3.2.1/glwf-3.2.1.bin.WIN32.zip> and unzip glwf-3.2.1.bin.WIN32 and save it under C:\OpenGLWrappers
7. GLFW 32 bit: Download <https://www.glwf.org/download.html> and unzip glwf-3.3.2.bin.WIN64 and save it under C:\OpenGLWrappers
8. Copy freeglut.dll from C:\OpenGLWrappers\freeglut-MSVC-2.8.1-1.mp\freeglut\bin to C:\Windows\SysWOW64.
9. Copy freeglut.dll from C:\OpenGLWrappers\freeglut\bin\x64 to C:\Windows\system32.
10. Copy glew32.dll from C:\OpenGLWrappers\glew-1.10.0-win32\glew-1.10.0\bin\ReleaseWin32 to C:\Windows\SysWOW64.
11. Copy glew32.dll from C:\OpenGLWrappers\glew-2.1.0\bin\Release\x64 to C:\Windows\system32.
12. Copy glfw3.dll from C:\OpenGLWrappers\glfw-3.2.1.bin.WIN32\lib-vc2015 to C:\Windows\SysWOW64.
13. Copy glfw3.dll from C:\OpenGLWrappers\glfw-3.3.2.bin.WIN64\lib-vc2019 to C:\Windows\system32.
14. Check if glu32.dll is already in C:\Windows\SysWOW64. Normally, it should be. If not, search resources on the web to reinstall it there. Now, if, in fact, you were able to place OpenGLWrappers in C:, then you should be all set run the book programs so go ahead and test your environment.


Create and Use an OpenGL Project Template

1. Open Visual Studio 2019 from the Start Menu to bring up the MSVS Start Page. Click New Project and in the popup dialog box select Visual C++ . For Name enter OpenGLProjectTemplate and for Location any convenient folder. Leave the boxes "place solution and project in the same folder" unchecked. Click OK.


Create a new project

Recent project templates


 CUDA 10.2 Runtime

 Python Application

Python

 Windows Desktop Wizard

C++

 Empty Project

C++

C++

[Clear all](#)

All languages

All platforms

All project types



Install Windows Universal tools for C++ development

Installation required

Tools for developing Windows Universal C++ apps are available.
Click OK to install.

C++

UWP

Windows



Empty Project

Start from scratch with C++ for Windows. Provides no starting files.

Console

C++

Windows



Console App

Run code in a Windows terminal. Prints "Hello World" by default.

Console

C++

Windows



Windows Desktop Wizard

Create your own Windows app using a wizard.

Console

C++

Desktop

Library

Windows



Windows Desktop Application

A project for an application with a graphical user interface that runs on Windows.

C++

Desktop

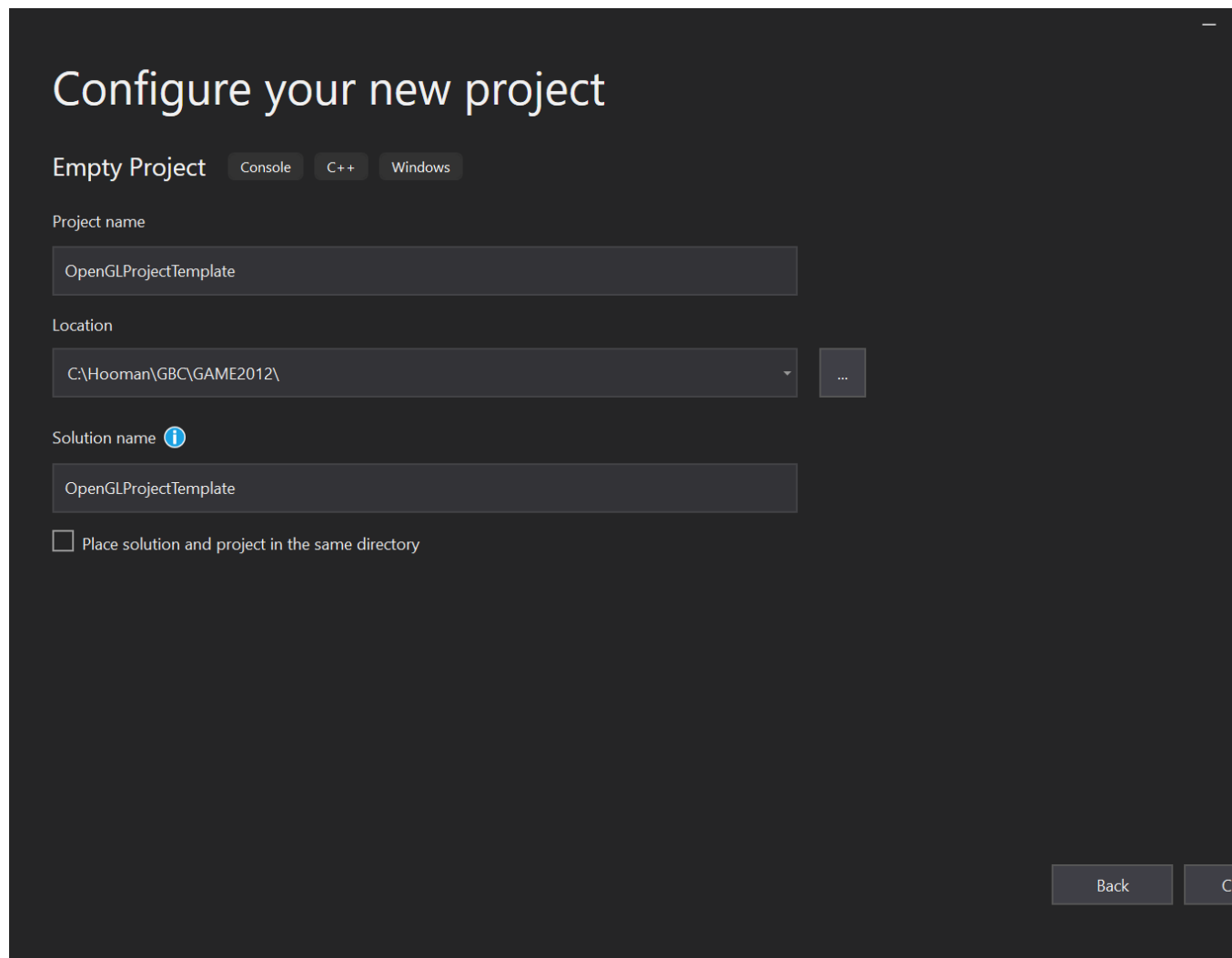
Windows



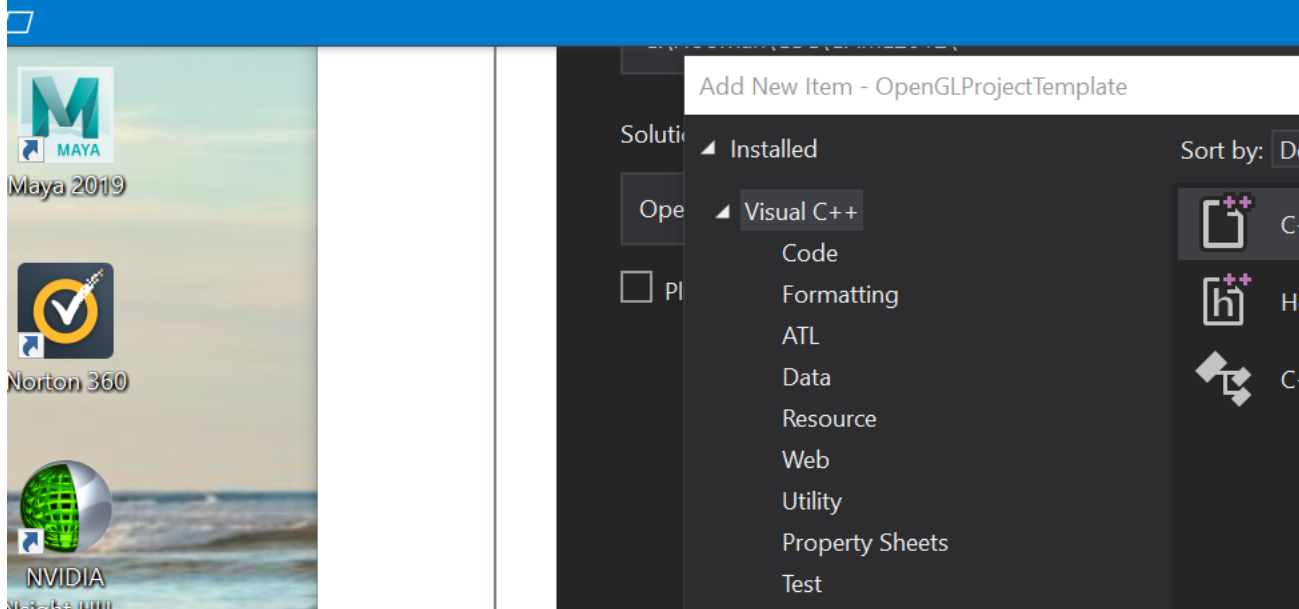
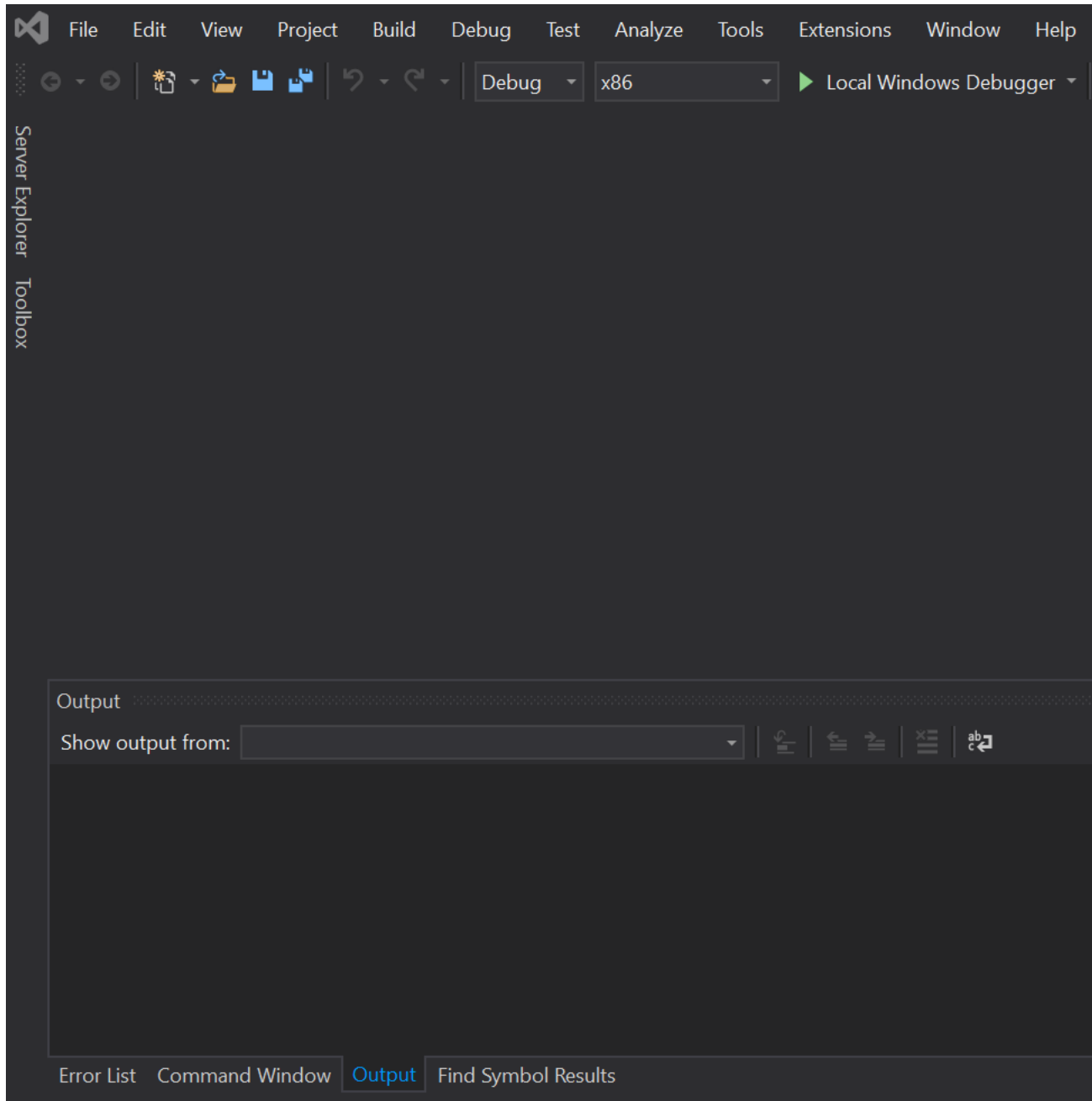
Shared Item Project

[Back](#)

[Next](#)



2. The MSVS page of OpenGLProjectTemplate comes up. On the tool bar click View -> Solution Explorer to open the Solution Explorer pane. Right click Source Files -> Add → New Item. Select Visual C++ -> C++ File (.cpp), name the new file test.cpp and click Add to see test.cpp in a pane.



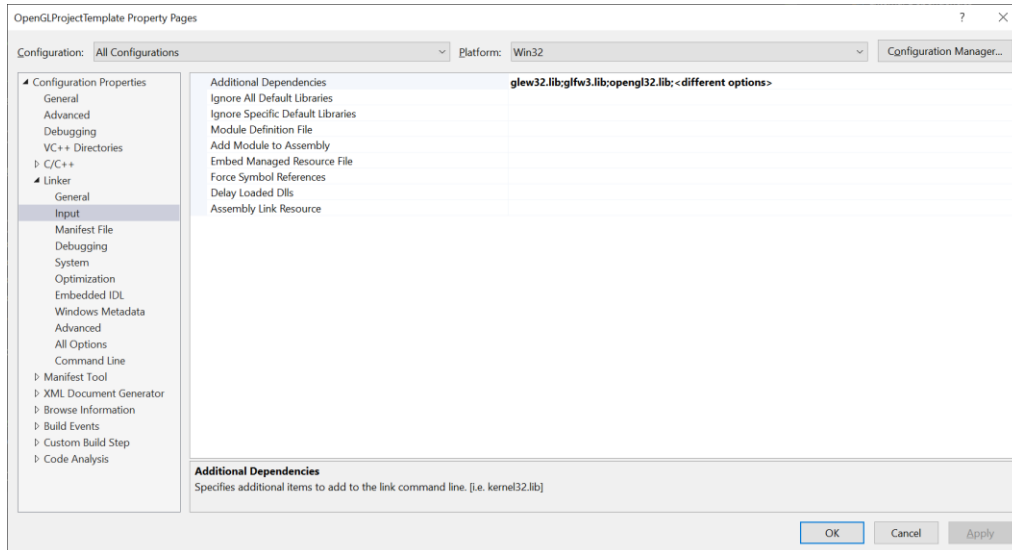
4. Open the file test.cpp in a text editor and copy its contents into test.cpp. Ignore the multiple red lines in



test.cpp

test.cpp for now.

5. On the tool bar of the MSVS page click Project -> OpenGLProjectTemplate Properties to bring up the Property Pages. Select All Configurations from the dropdown menu to the right of "Configuration:" and choose "Win32" platform



6. Expand Configuration Properties and click Configuration Properties C/C++ -> General -> Additional Include Directories -> Edit (in the drop-down menu) to open the Additional Include Directories dialog box. Click the New Line icon successively to create a new line to add each of the following folders (order doesn't matter; you can click the box with ".." at the right of the empty new line to navigate to the folder):

If OpenGLWrappers isn't in C:, then navigate to where it is.

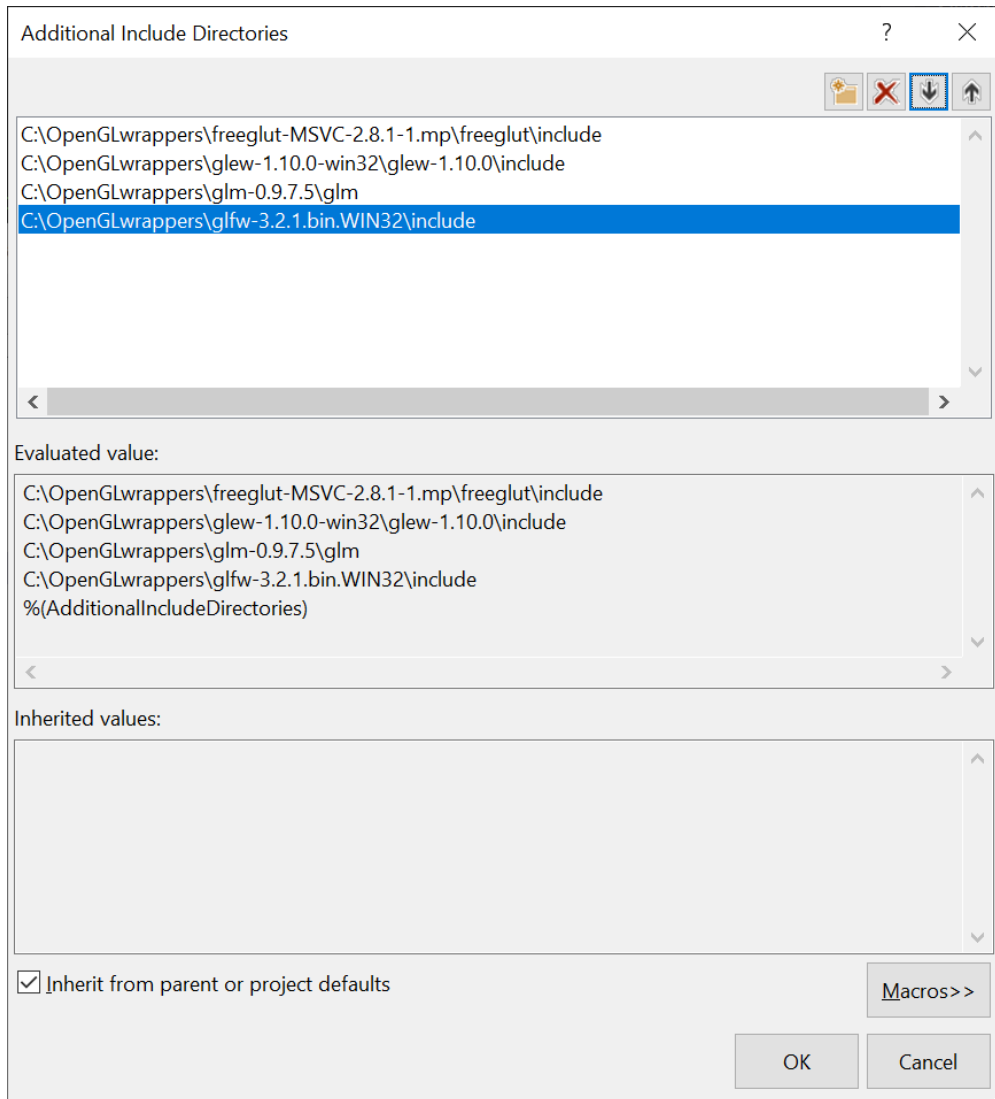
C:\OpenGLWrappers\freeglut-MSVC-2.8.1-1.mp\freeglut\include

C:\OpenGLWrappers\glew-1.10.0-win32\glew-1.10.0\include

C:\OpenGLWrappers\glm-0.9.7.5\glm

C:\OpenGLWrappers\glfw-3.2.1.bin.WIN32\include

Click OK to confirm.



7. Click Configuration Properties -> Linker -> General -> Additional Library Directories ->Edit (in the drop-down menu) to open the Additional Library Directories dialog box. Click the New Line icon successively to create a new line to add each of the following folders 3 (order doesn't matter; you can click the box with \: : ." at the right of the empty new line to navigate to the folder):

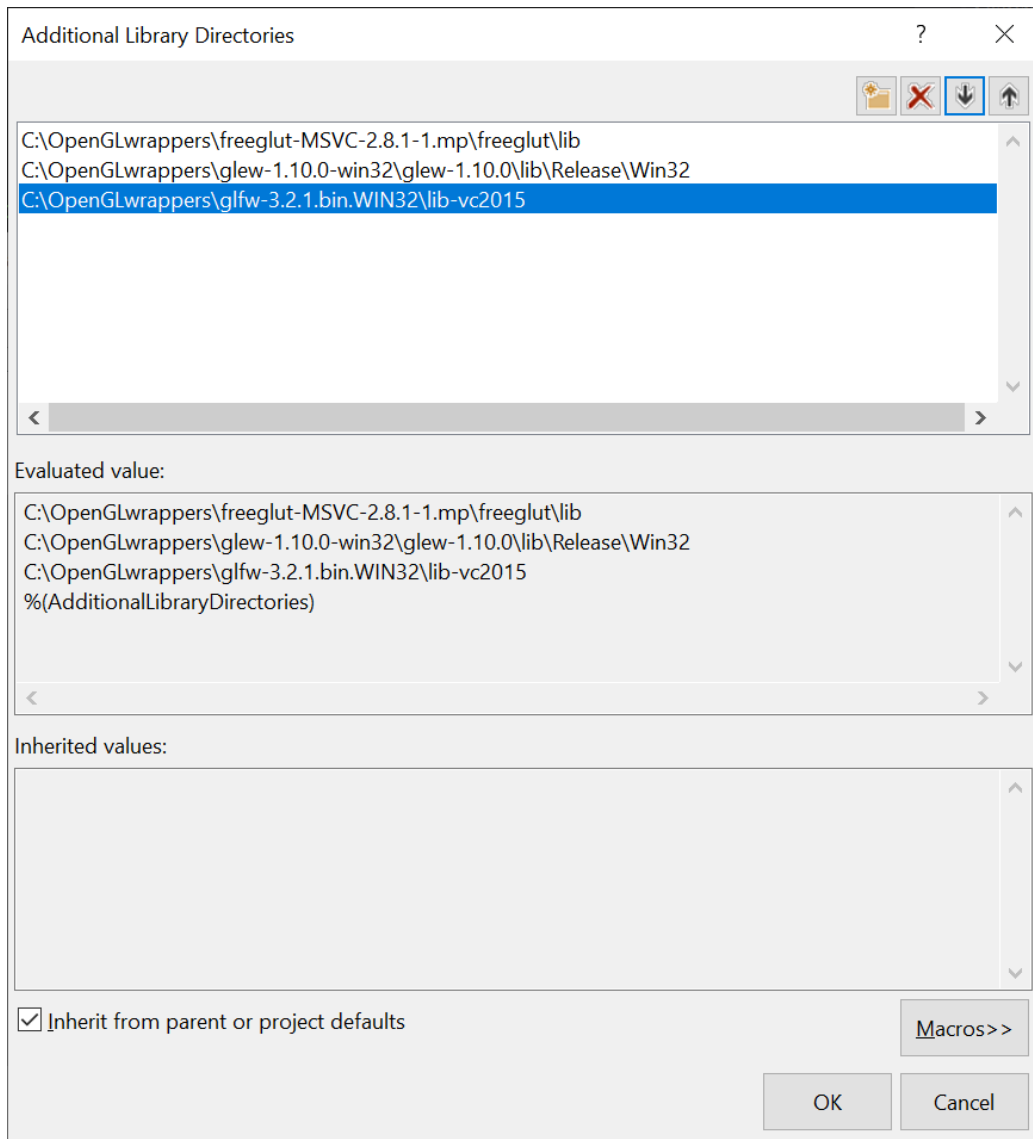
If OpenGLWrappers isn't in C:, then navigate to where it is.

C:\OpenGLWrappers\freeglut-MSVC-2.8.1-1.mp\freeglut\lib

C:\OpenGLWrappers\glew-1.10.0-win32\glew-1.10.0\lib\Release\Win32

C:\OpenGLWrappers\glfw-3.2.1.bin.WIN32\lib-vc2015

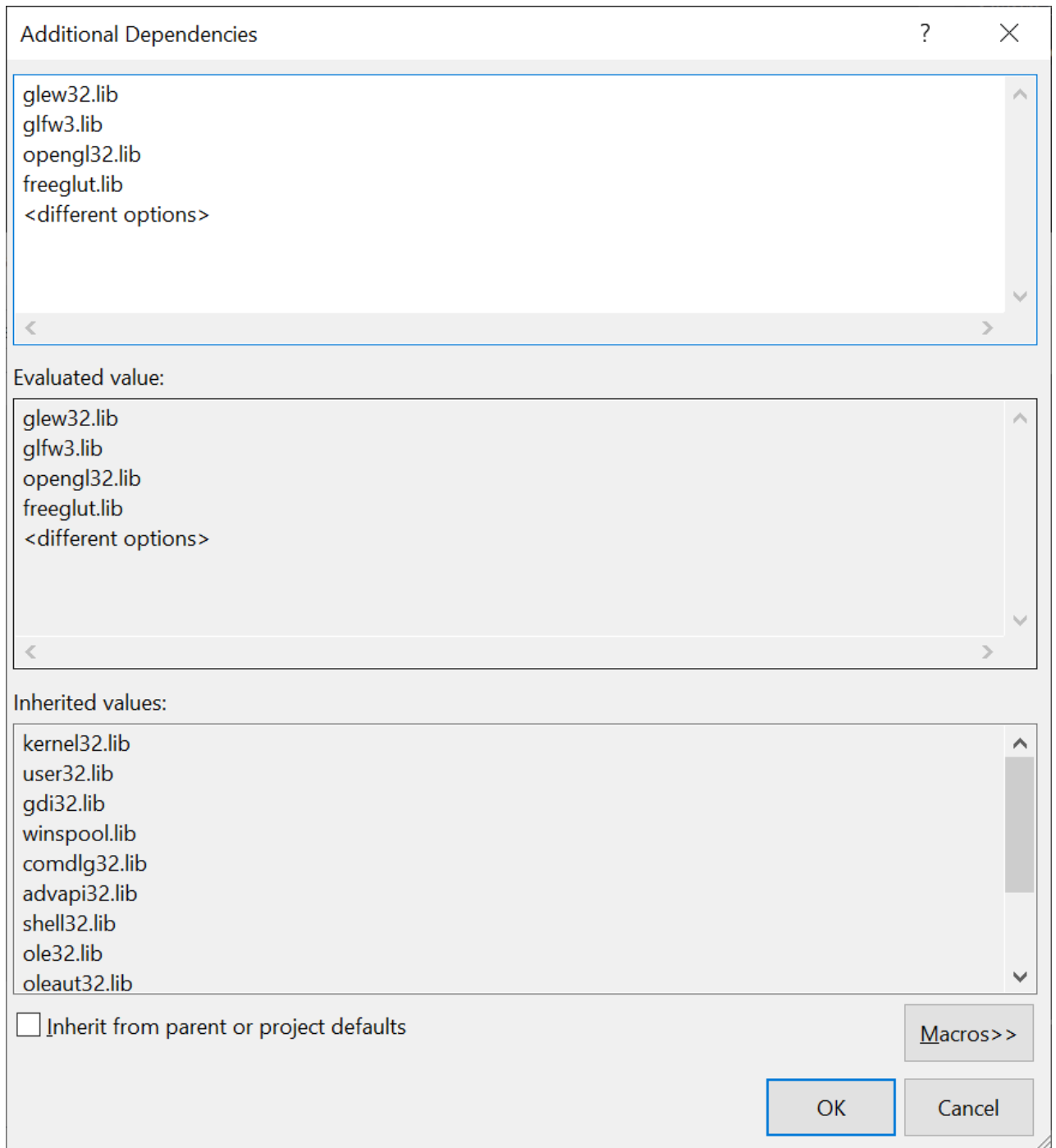
Click OK to confirm.



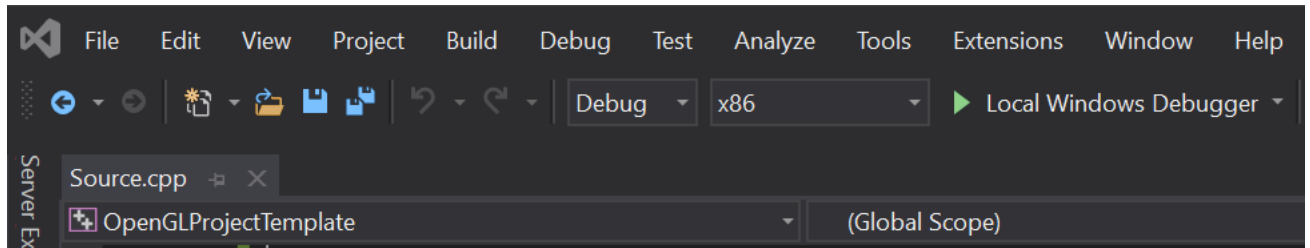
8. Click Configuration Properties -> Linker -> Input -> Additional Dependencies -> Edit (in the drop-down menu) to open the Additional Dependencies dialog box. Manually enter the following file names one per line in the window at the top (order doesn't matter):

glew32.lib
glfw3.lib
opengl32.lib
freeglut.lib

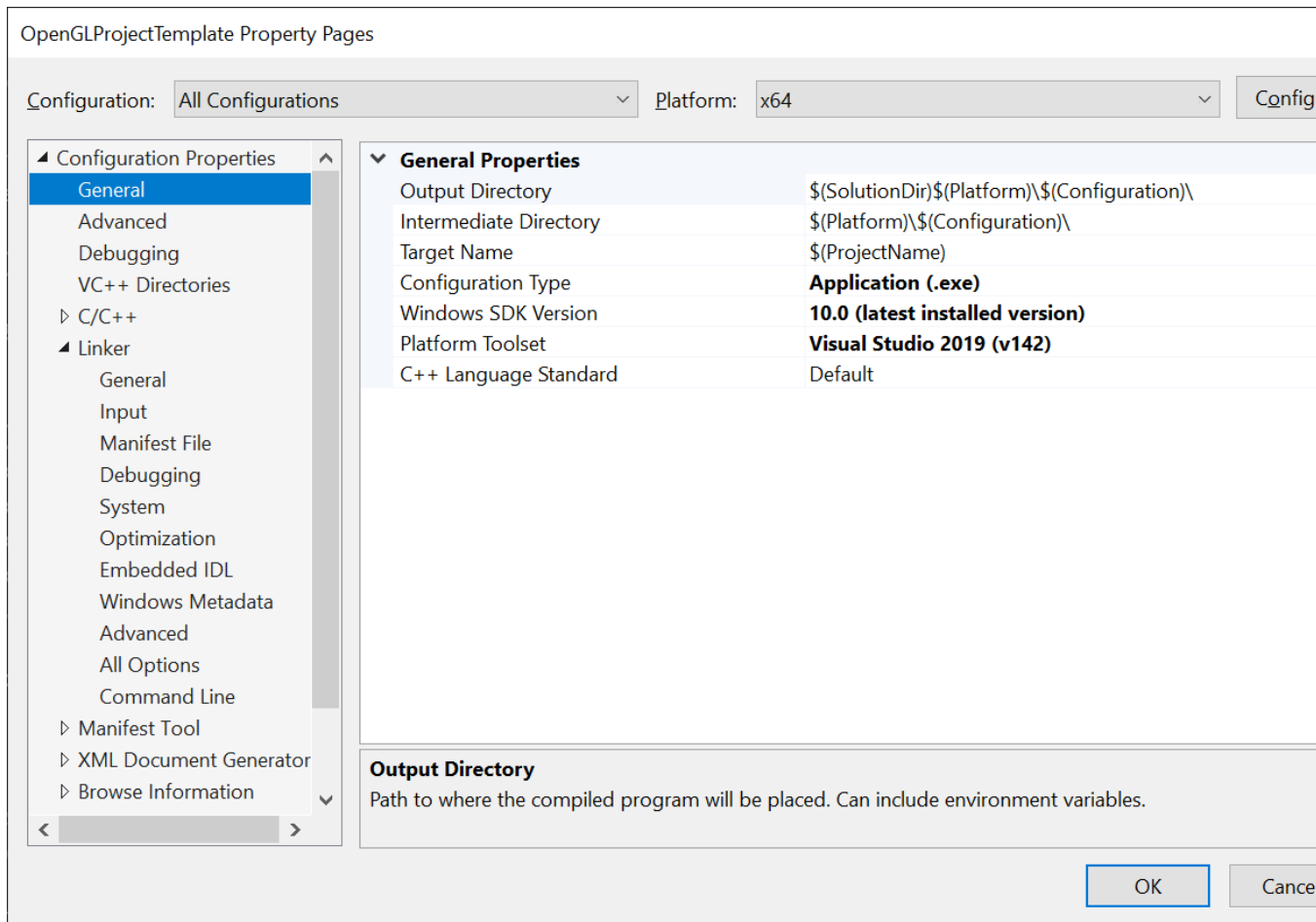
Click OK to confirm.



9. Now, Test the Environment using x86 platform. You should see a black C++ window and an OpenGL window.



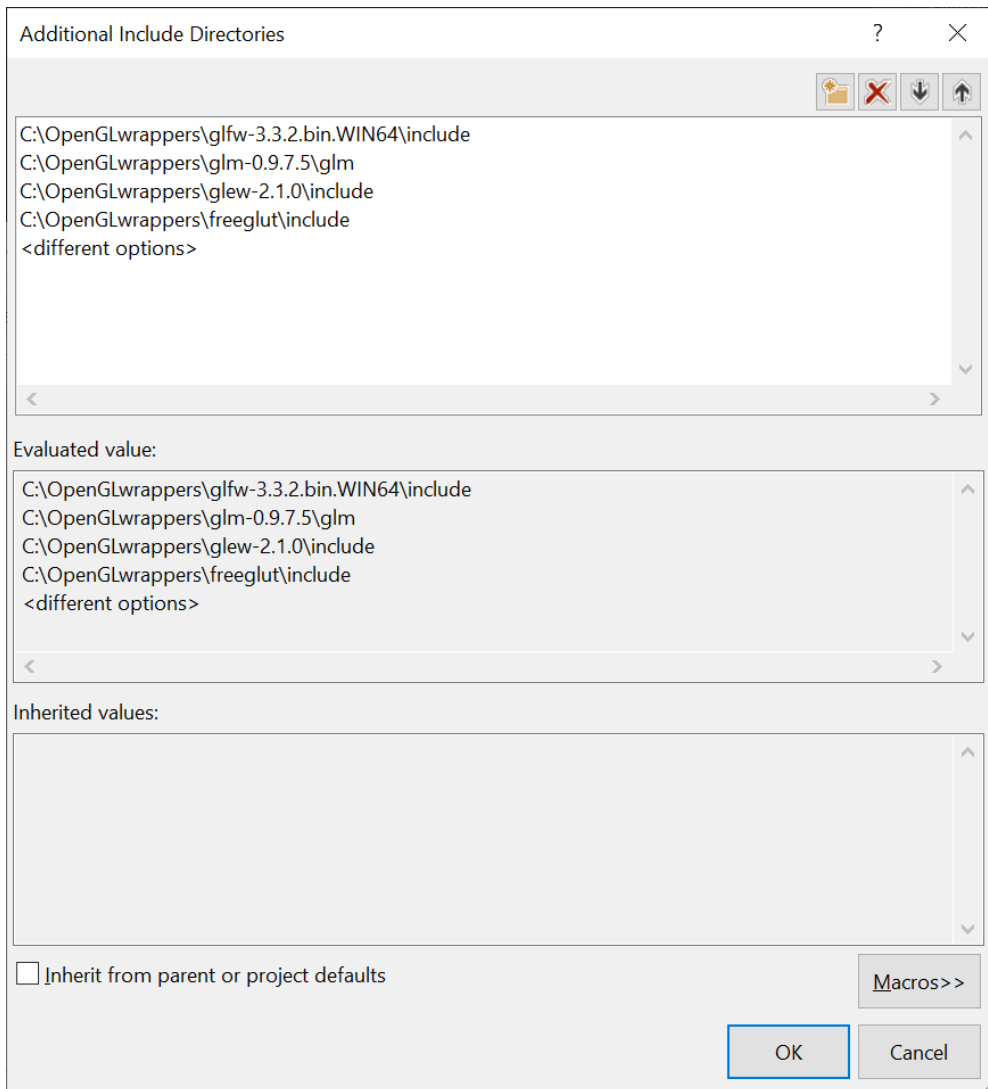
10. On the tool bar of the MSVS page click Project -> OpenGLProjectTemplate Properties to bring up the Property Pages. Select All Configurations from the dropdown menu to the right of "Configuration:" and choose "x64" platform



11. Expand Configuration Properties and click Configuration Properties C/C++ -> General -> Additional Include Directories -> Edit (in the drop-down menu) to open the Additional Include Directories dialog box. Click the New Line icon successively to create a new line to add each of the following folders (order doesn't matter; you can click the box with ".." at the right of the empty new line to navigate to the folder):

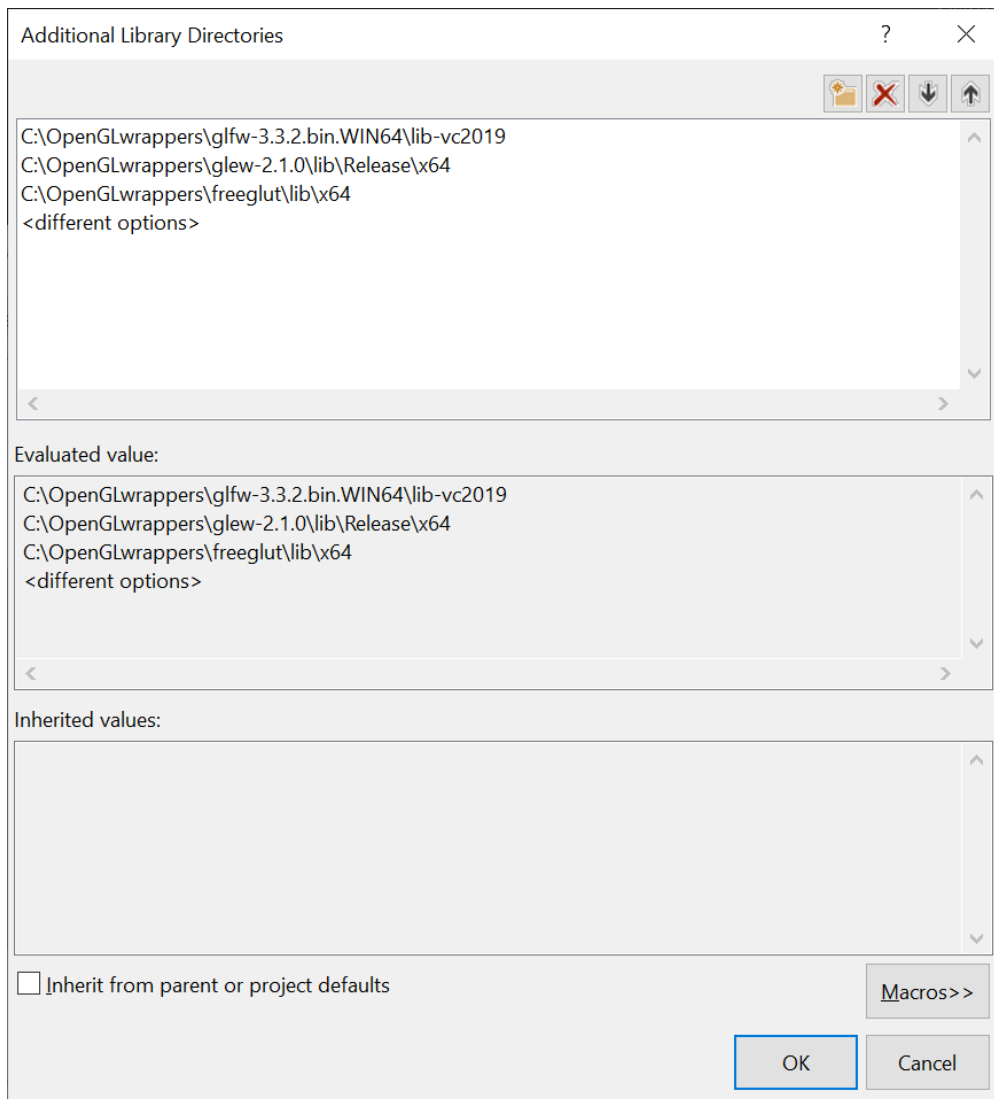
C:\OpenGLwrappers\freeglut\include
C:\OpenGLwrappers\glew-2.1.0\include
C:\OpenGLwrappers\glm-0.9.7.5\glm

C:\OpenGLWrappers\glfw-3.3.2.bin.WIN64\include



12. Click Configuration Properties -> Linker -> General -> Additional Library Directories ->Edit (in the drop-down menu) to open the Additional Library Directories dialog box. Click the New Line icon successively to create a new line to add each of the following folders 3 (order doesn't matter; you can click the box with \: : ." at the right of the empty new line to navigate to the folder):

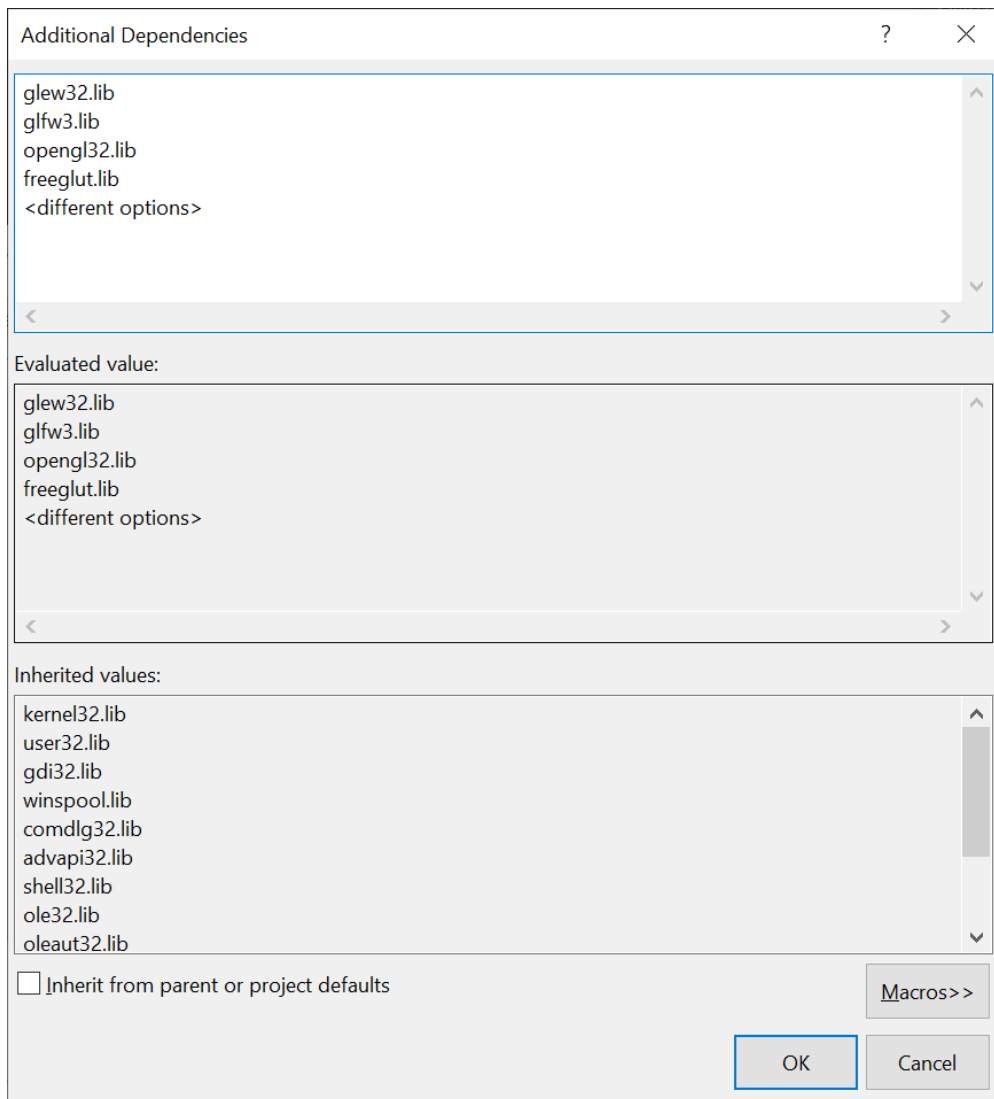
C:\OpenGLWrappers\freeglut\lib\x64
C:\OpenGLWrappers\glew-2.1.0\lib\Release\x64
C:\OpenGLWrappers\glfw-3.3.2.bin.WIN64\lib-vc2019



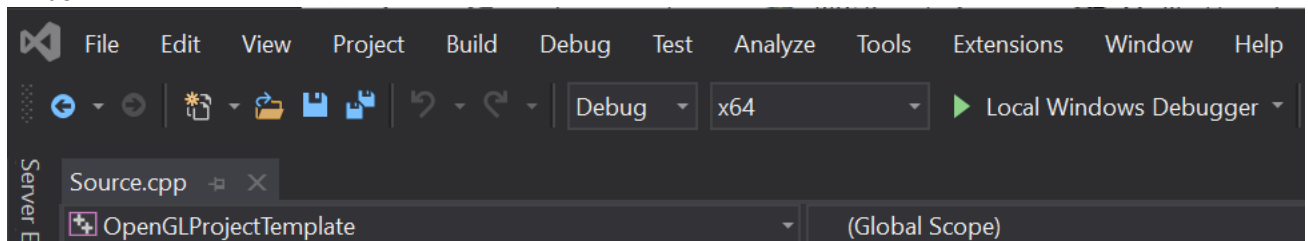
13. Click Configuration Properties -> Linker -> Input -> Additional Dependencies -> Edit (in the drop-down menu) to open the Additional Dependencies dialog box. Manually enter the following file names one per line in the window at the top (order doesn't matter):

glew32.lib
glfw3.lib
opengl32.lib
freeglut.lib

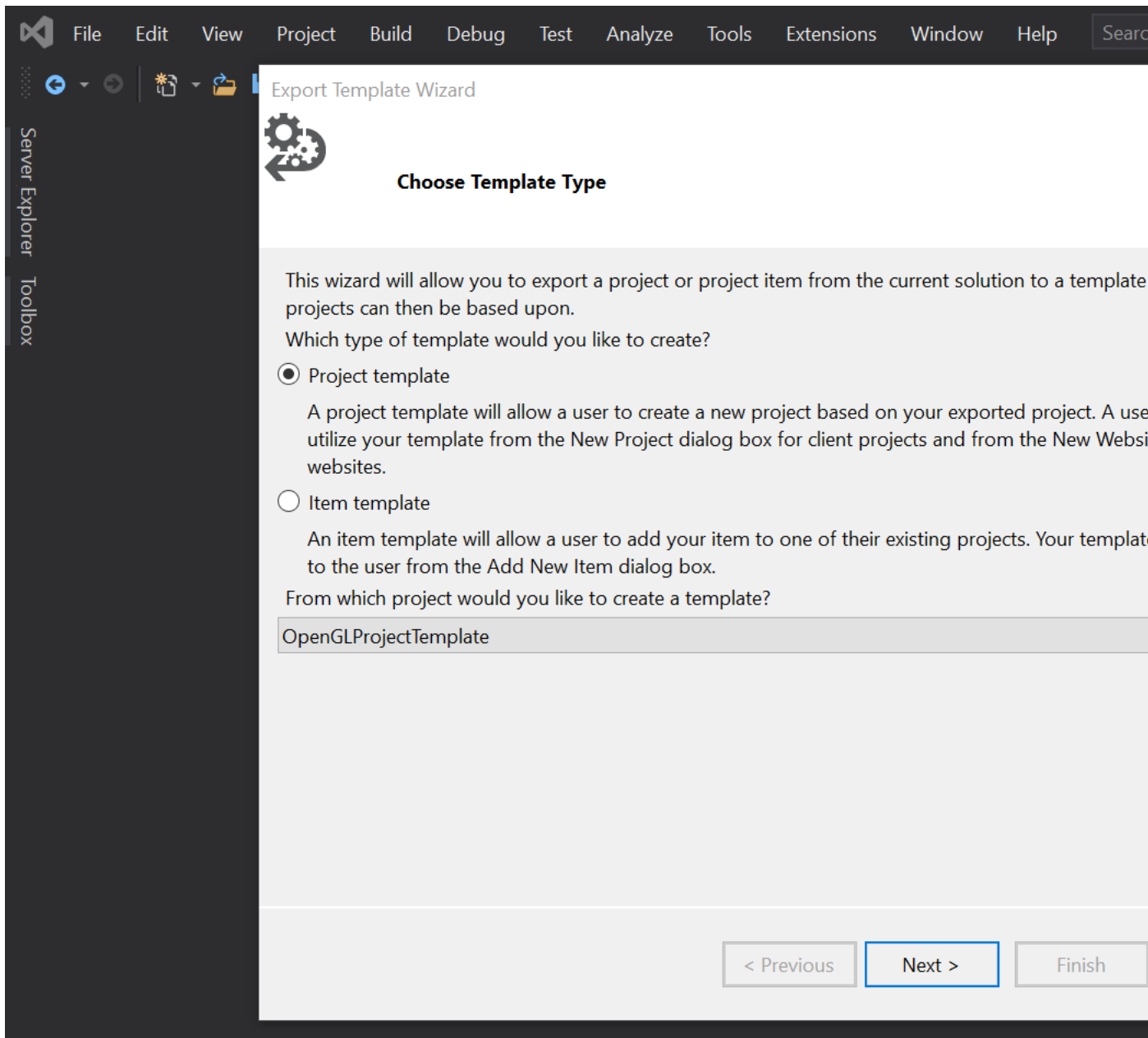
Click OK to confirm.




14. Now, Test the Environment using x86 platform. You should see a black C++ window and an OpenGL window.



15. On the MSVS toolbar click File -> Save All. Next click Project -> Export Template. In the Export Template Wizard choose the Project template radio button and click Next. Name the template OpenGLProjectTemplate and check the box "Automatically import the template into Visual Studio" and uncheck the box "Display an explorer window on the output files folder". Click Finish. The template OpenGLProjectTemplate has been created and you can, in fact, now delete the project OpenGLProjectTemplate from the folder you created it in.



Export Template Wizard ?



Select Template Options

Template name:

Template description:

Icon Image:
 Br...

Preview Image:
 Br...

Output location:

☒ Automatically import the template into Visual Studio






☐ Display an explorer window on the output files folder

< Previous Next > Finish Cancel

16. To use the template, open Visual Studio 2019 from the Start Menu to bring up the MSVS Start Page. Click New Project and search for OpenGLProjectTemplate. Name the project and an appropriate folder for its location and click OK. The new project comes up. To check if the template is ok.

Create a new project

Recent project templates

-  Empty Project C++
-  OpenGLProjectTemplate
-  Windows Desktop Wizard C++
-  CUDA 10.2 Runtime
-  Python Application Python

Search for templates (Alt+S)

All languages

All platforms



OpenGLProjectTemplate
OpenGL Project Template



Console App (.NET Core)

A project for creating a command-line application that runs on Windows, Linux and MacOS.

Console

C#

Linux

macOS

Windows



Console App (.NET Core)

A project for creating a command-line application that runs on Windows, Linux and MacOS.

Console

Linux

macOS

Visual Basic

Windows



ASP.NET Core Web Application

Project templates for creating ASP.NET Core web apps that run on Linux and macOS using .NET Core or .NET Framework. Choose Pages, MVC, or Single Page Apps (SPA) using Angular, React, or Vue.

Cloud

C#

Linux

macOS

Service

Web



Blazor App

Project templates for creating Blazor apps that run on the web or in the browser on WebAssembly. These templates create apps with rich dynamic user interfaces (UIs).